EVEREST: Evaluation and Validation of Election-Related Equipment, Standards and Testing[*†]

Final Report
December 7, 2007

[*] This report was prepared by teams from Pennsylvania State University, the University of Pennsylvania, and WebWise Security, Inc. as part of the EVEREST voting systems analysis project initiated by the Secretary of State of Ohio in the Winter of 2007. Unless otherwise indicated, all analyses detailed in this report were carried out at the home institutions between October 1, 2007 and December 7, 2007.

[†] This report is released by Ohio Secretary of State Jennifer Brunner consistent with the Ohio Public Records Act, Ohio R.C. 149.43. The reader of this document is advised that any conduct intended to interfere with any election, including tampering with, defacing, impairing the use of , destroying, or otherwise changing a ballot, voting machine, marking device, or piece of tabulating equipment, is inconsistent with Ohio law and may result in a felony conviction under, among other sections, Ohio R.C. 3599.24 and 3599.27.

Project Personnel

Pennsylvania State University Team

Patrick McDaniel, *Principal Investigator & Team Lead*

Kevin Butler
Pennsylvania State University

William Enck
Pennsylvania State University

Harri Hursti
Independent Contractor

Steve McLaughlin
Pennsylvania State University

Patrick Traynor
Pennsylvania State University

University of Pennsylvania Team

Matt Blaze, *Team Lead*

Adam Aviv
University of Pennsylvania

Pavol Černý
University of Pennsylvania

Sandy Clark
University of Pennsylvania

Eric Cronin
University of Pennsylvania

Gaurav Shah
University of Pennsylvania

Micah Sherr
University of Pennsylvania

WebWise Security, Inc.

Giovanni Vigna, *Team Lead*

Richard Kemmerer
WebWise Security, Inc.

Davide Balzarotti
WebWise Security, Inc.

Greg Banks
WebWise Security, Inc.

Marco Cova
WebWise Security, Inc.

Viktoria Felmetsger
WebWise Security, Inc.

William Robertson
WebWise Security, Inc.

Fredrik Valeur
WebWise Security, Inc.

Policy and Document Consultants

Joseph Lorenzo Hall
University of California, Berkeley

Laura Quilter
University of California, Berkeley

CONTENTS

Part I

Project Overview

CHAPTER 1

EXECUTIVE SUMMARY

This report details the findings of one part of the Ohio Secretary of State's EVEREST: Evaluation and Validation of Election-Related Equipment, Standards and Testing initiative. The goal of this review was to assess the security of electronic voting systems used in Ohio, and to identify any procedures that may eliminate or mitigate discovered issues. The review teams were provided the source-code (computer instructions), software, and election equipment for the majority of systems used in Ohio. During the 9 week review, security researchers at three institutions studied the software and systems and identified and confirmed security issues. The evaluated systems included those designed and developed by Election Systems and Software (ES&S), Hart InterCivic (Hart) and Premier Election Solutions (Premier, formerly Diebold).

All of the studied systems possess critical security failures that render their technical controls insufficient to guarantee a trustworthy election. While each system possessed unique limitations, they shared critical failures in design and implementation that lead to this conclusion:

- **Insufficient Security** - The systems uniformly failed to adequately address important threats against election data and processes. Central among these is a failure to adequately defend an election from insiders, to prevent virally infected software from compromising entire precincts and counties, and to ensure cast votes are appropriately protected and accurately counted.

- **Improper Use or Implementation of Security Technology** - A root cause of the failures present in the studied systems is the pervasive mis-application of security technology. Failure to follow standard and well-known practices for the use of cryptography, key and password management, and security hardware seriously undermine the protections provided. In several important cases, the misapplication of commonly accepted principles renders the security technology of no use whatsoever.

- **Auditing** - All of the systems exhibited a visible lack of trustworthy auditing capability. In all systems, the logs of election practices were commonly forgeable or erasable by the principals who they were intended to be monitoring. The impact of the lack of secure auditing is that it is difficult to know when an attack occurs, or to know how to isolate or recover from it when it is detected.

- **Software Maintenance** - The software maintenance practices of the studied systems are deeply flawed. This has led to fragile software in which exploitable crashes, lockups, and failures are common in normal use. Such software instability is likely to increase over time, and may lead to highly insecure and unreliable elections.

The security failures present in the studied systems place incredible pressures on the physical election procedures. The review teams provide a number of procedures that mitigate or completely address issues throughout. However, in many cases, we could not identify any practical procedures that adequately address the limitations.

The review teams were able to subvert every voting system we were provided in ways that would often lead to undetectable manipulation of election results. We were able to develop this knowledge within a few weeks. However, most of the problems that we found could have been identified with only limited access to voting equipment. Thus, it is safe to assume that motivated attackers will quickly identify – or already have – these and many other issues in these systems. Any argument that suggests that the attacker will somehow be less capable or knowledgeable than the reviewer teams, or that they will not be able to reverse engineer the systems to expose security flaws is not grounded in fact.

The issues identified in this report are based on exploitable vulnerabilities that are practical under election conditions. As detailed throughout, the vast majority of these vulnerabilities take seconds to perform and require little access to privileged data. Thus, most are *attacks of opportunity* which can be performed in any number of ways. This suggests that care should be taken to not underestimate the attackers or the means and vulnerabilities at their disposal.

The security of the studied election systems is crippled by flaws in its design and implementation. Therefore, after an extensive analysis, the teams unanimously believe that such flaws mandate fundamental and broad reengineering before the technical protections can approach the goal of guaranteeing trustworthy elections.

The remainder of this report details the technical and observational findings of this study. Readers interested in briefly obtaining a more concrete understanding of the limitations of each system in isolation should review the executive summaries in Chapter 4 (ES&S), Chapter 10 (Premier), and Chapter 16 (Hart). Readers interested in obtaining a broader understanding of the report and its findings are referred to the next chapter.

CHAPTER 2

OVERVIEW

This report was created as part of the Ohio Secretary of State's EVEREST: Evaluation and Validation of Election-Related Equipment, Standards and Testing initiative. The goal of the initiative was to assess the reliability, accessibility, and security of electronic voting systems used in Ohio. This report outlines the methods and results of the expert source code and red-teaming efforts carried out in the study. This effort was focused principally on evaluating the security of the voting systems. The teams further identify, where possible, procedures that may mitigate any issues.

We began this work on October 1, 2007 and completed the work on December 7th, 2007 with the delivery of this report to the Ohio Secretary of State Jennifer Brunner's office.

2.1 Report Structure

This report is divided into 5 parts. Part 1 provides an executive overview of the evaluation findings (Chapter 1), a broad description of the evaluation structure, activities, and limitations (this chapter), as well as identify the security requirements of voting systems in Ohio (Chapter 3). Parts 2-4 detail the vendor-specific evaluations. Part 2 (chapters 5 through 9) describes the ES&S system evaluation and findings, Part 3 (chapters 10 through 15) describes the Hart system evaluation and findings, and Part 4 (chapters 16 through 21) describes the Premier evaluation and findings. Part 5 contains reference appendices that provide additional non-essential technical and procedural information.

2.1.1 Reading Recommendations

This report is intended for a broad audience. Non-technical readers interested in an overview of the findings should read the executive summaries provided in Chapter 1 and at the beginning of each vendor specific evaluation section of the report. Readers interested in a broader and more technical characterization of results should read the architectural and systemic issues sections of each vendor-specific part of the report.

Readers looking to acquire a comprehensive understanding should review the sections of this report in the following order:

1. This chapter (overview) and Chapter 3 (threat model).
2. Vendor-independent executive summary in chapter 1 and vendor-specific executive summaries in Chapter 4, 10, and 16.
3. Overview and introduction to each vendor system in Chapters 5, 11, and 17.
4. Vendor-specific architectural and structural issues in chapters 6, 12, and 18 (*highly recommended*)
5. Vendor-specific issue identification and attack scenario chapters. These are the most technical descriptions in the report, and may be difficult for non-technical readers to navigate.

The appendices provide background material that may useful in understanding the substance in the report. They are referenced where appropriate, but may be ignored without significant loss of detail.

2.2 Evaluation Design

We now provide a broad description of the evaluation. Three vendors participated in the evaluation; Election Systems and Software (ES&S) of Omaha, Nebraska; Hart InterCivic (Hart) of Austin, Texas; and Premier Election Solutions (Premier, formally Diebold) of North Canton, Ohio. To the degree possible under the tight time constraints and as made available to the reviewers, all aspects of the voting systems were evaluated. This includes the touchscreen voting terminals, optical scan voting terminals, and all back-end election management and vote tally systems.

Three teams participated in the review: a team of faculty, graduate students, and one consultant at Pennsylvania State University in University Park, Pennsylvania, a team of faculty and graduate students at the University of Pennsylvania in Philadelphia, Pennsylvania, and a collection of security consultants at WebWise Security, Inc. in Santa Barbara, California. The team was also supported by two experts in election procedures from the University of California Berkeley, as well as numerous election officials and staff at the Secretary of State's office in Ohio.

Prior to this report, the ES&S system had not received a detailed source code and red-teaming security review. For this reason, we focused a good portion of the evaluation team on ES&S. The University of Pennsylvania team focused on evaluating the source code associated with the ES&S systems. They studied the design and implementation of the ES&S software and firmware, and performed penetration tests using specially crafted tools.[1] The WebWise Security, Inc. team focused on performing "red-teaming" exercises on the ES&S voting equipment. They crafted special tools designed to circumvent the operation of the election. They also performed a number of physical attacks that misuse the equipment in such a way as to force it to alter or prevent an election.

The Pennsylvania State University team focused on the Hart and Premier systems, both of which have been the subject of prior security reviews. They were broadly charged with investigating the security of those systems. As part of that effort, they sought to confirm previously reported issues. The PSU team performed source code analysis and red-teaming on the voting systems, and attempted to understand the impact of discovered and confirmed issues in the light of Ohio election procedures.

The team was also instructed to assess the degree to which the security issues are mitigated by current Ohio procedures. Where procedures do not provide protections, the team was instructed to identify mitigating procedures, if any. Detailed in the technical evaluation, this required the assistance of the procedural experts and election officials in Ohio. We also comment briefly on the realities and common misconceptions about procedural mitigation in the next Chapter.

2.3 Methodology

The evaluation of security in any non-trivial software systems is a difficult task. There is no set way to approach such an effort, and there is no way to enumerate all security issues and errors in a system. Conversely, it is a practical impossibility to develop any system that is completely free of bugs. The job of the security practitioner is to identify what bugs or design errors are in the system, and to assess whether those issues can be exploited to undermine the system's function, e.g., alter an election result.

A security evaluation of any system the size of the election systems used in Ohio can take years and require many security and test engineers. Given the compressed schedule, we focused on the evaluation

[1] We would like to thank Fortify Software for providing their commercial analysis tools free of charge.

of the system's ability to function correctly in the face of a determined adversary. As in previous studies, we assess the existence and quality of safeguards preventing malicious or accidental ballot, vote, or result tampering, or the denial-of-service to voters. We make critical judgments based on demonstrable evidence that a system will behave correctly under all circumstances that may be encountered in the field.

In this evaluation, we ask the following questions about the quality of the design and implementation of the voting systems, and speak to the soundness of the security and software engineering principals to which an election system must adhere. In this, we ask the following questions:

- Does the design use accepted standard practices for engineering? Is the design likely to result in a reliable and secure system?
- Does the system have the basic safeguards against misuse? Is the code defensively written to prevent misuse of forced errors or bad input? Are the interfaces between components well designed and documented?
- Does the system have the safeguards necessary to identify and audit potential error conditions? When the system is compromised, to what degree can the damage be determined and mitigated?
- Does the system have the basic security infrastructure needed in a system of this type? Is this infrastructure properly designed, implemented, and used?
- What assumptions about trust are made? Is this trust appropriate? Does the system behave properly when a trusted system is compromised?
- Is the environment in which the system is constructed likely to result in a reliably and securely functioning system? Is the use and selection of third-party tools and components likely to introduce hidden side-effects that compromise the election integrity or introduce avenues for denial-of-service?

2.3.1 Prior Studies

Previous studies of election systems have identified numerous security issues that can affect the accuracy and operation of an election. We widely use these studies as reference material. These studies include:

- *California Top-to-Bottom Review (TTBR)* - Initiated by the Secretary of State of California in 2007, this is the most comprehensive study of voting systems to date.[2] The study evaluated the Hart, Premier, and Sequoia systems and documentation. The TTBR report was carried out by 9 academic teams similar to those in this study.
- *Florida* - These reports studied the Premier (then Diebold) system's software,[3] and were supplemented[4] with an evaluation of the vendors' subsequent software updates.

[2]California Secretary of State, *Top-To-Bottom Review*. July 2007 ⟨URL: `http://www.sos.ca.gov/elections/elections_vsr.htm`⟩.

[3]Ryan Gardner et al., *Software Review and Security Analysis of the Diebold Voting Machine Software*. Security and Assurance in Information Technology (SAIT) Laboratory, Florida State University, For the Florida Department of State, July 27, 2007 ⟨URL: `http://election.dos.state.fl.us/pdf/SAITreport.pdf`⟩.

[4]David Gainey, Michael Gerke and Alec Yasinsac, *Software Review and Security Analysis of the Diebold Voting Machine Software: Supplemental Report*. Security and Assurance in Information Technology (SAIT) Laboratory, Florida State University, For the Florida Department of State, August 10, 2007 ⟨URL: `http://election.dos.state.fl.us/pdf/DieboldSupplementalReportFinalSubmission.pdf`⟩.

- *Connecticut* - These two reports considered issues with the Premier (then Diebold) Optical Scanners[5] in 2006 and Touchscreen[6] systems in 2007.

- *Academic works* - there have been a large number of academic papers published within the security community on topics relating to the security of election systems. While some are more targeted to the broader science of information security, others are quite specific to the evaluation of vendor systems.[7]

We make use of these reports as they provide insight into the systems under evaluation. In particular, the previous state-initiated reports provided useful frameworks for identifying and reporting security issues.

2.4 Limitations

The review teams were hampered in several ways when conducting this review. The effect of these hurdles was that we were limited in the scope of the review, and in the amount of confirmation of previously reported issues that could be performed. More generally, the lack of access to important vendor materials tools slowed visibly or prohibited many review activities. The central inhibiting limitations were:

- In the initial EVEREST plan, the teams were to evaluate software updates to be provided by the vendors to the state of Ohio. While Hart and Premier had indicated they would provide the updates, they were not able to get them in on time to be reviewed. The degree to which software upgrades may address previous (or new) issues or introduce new ones is unknown.

  Note that systemic issues and modifications that require system or component redesign can not be addressed by simple software upgrades. These issues reflect the most grave concerns about the security and robustness of these systems, and any claims for issues fixed in software upgrades should be viewed with deep skepticism; all such claims must be demonstrated authoritatively. Moreover, pervasive code and design quality issues cannot be addressed via incremental upgrades, but can only be addressed through the rigorous and lengthy application of sound software engineering practices.

- The review teams were extremely limited on time. The review period was set to begin in early September 2007, but we did not receive initial reviewing materials (source code) until the 1st of October. Further, a usable complement of equipment and election materials (e.g., sample ballots) was received around the 10th of October with remaining equipment and materials arriving periodically up until early November. The vast majority of the review activities disclosed in the report were performed in the 8 week period between October 1, 2007 and November 19th, 2007.

- The received source code was missing several key items, some of which remained unresolved at the completion of the review. These missing components included boot-loaders, various user interface objects, third party libraries, and other tools and utilities. The set of documents received was largely

[5] A. Kiayias et al., *Security Assessment of the Diebold Optical Scan Voting Terminal*. UConn Voting Technology Research (VoTeR) Center, October 30, 2006 ⟨URL: `http://voter.engr.uconn.edu/voter/OS-Report_files/uconn-report-os.pdf`⟩.

[6] A. Kiayias et al., *Integrity Vulnerabilities in the Diebold TSX Voting Terminal*. UConn Voting Technology Research (VoTeR) Center, July 16, 2007 ⟨URL: `http://voter.engr.uconn.edu/voter/OS-TSX-Report_files/TSX_Voting_Terminal_Report.pdf`⟩.

[7] Tadayoshi Kohno et al., 'Analysis of an Electronic Voting System'. In Proceedings of the IEEE Symposium on Security and Privacy. May 2004; Ariel Feldman, J. Alex Halderman and Edward Felten, 'Security Analysis of the Diebold AccuVote-TS Voting Machine'. In USENIX/ACCURATE Electronic Voting Technology Workshop (EVT). 2007; Aggelos Kiayias et al., 'An Authentication and Ballot Layout Attack against an Optical Scan Voting Terminal'. In Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop. August 2007; Elliot Proebstel et al., 'An Analysis of the Hart Intercivic DAU eSlate'. In Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop. August 2007.

complete, and generally the vendors were responsive to requests for missing materials. The Ohio SoS's office was instrumental in tracking these requests.

In addition to writing their own programs, the election systems rely heavily on third party software that implement interfaces to the operating systems, local databases, and devices such as optical scanners. We were given no access to the source code of the third party libraries upon which the election systems are built. The reasoning provided by the vendors was that they were restricted by licensing agreements. Thus, this review could only observationally review the third party software–a very limited and ineffective method of evaluation. This is an area of concern, as the construction and features of this software is unknown, and may contain undisclosed vulnerabilities such trojan horses or other malware.

Hart, Premier, and ES&S provided all the documents that we asked for. We asked for all the documents listed in previous reports and on the vendor websites.

- The review teams initially had no build environment tools (compilers, linkers, and libraries to make the executable programs from source code). With help from the Secretary of State's office and using existing relationships of the home institutions, we able to cobble together a loose collection of tools that allowed us to construct most of the set of tools needed for evaluation. However, we were unable to build functional vendor systems, and thus could not confirm that the source code matches the software provided to us.

  Several of the tools are no longer commercially available in any form, the companies that created them having gone out of business many years ago. This unavailability harms the evaluator much more than the attacker: inability to re-compile the vendor software to exercise particular code paths takes away one of the most significant advantages a source code analysis has in rapidly evaluating vulnerabilities.

  Premier provided the Penn State team with an ITA (independent testing authority) computer that included all the build environment tools for the Premier system on November 16th, 2007. This equipment and software arrived as the team was transitioning from reviewing to writing the final report. Thus, it was not useful for directing the group's activities, but only served to assist in minor confirmation of results we had found during the course of our investigations.

  Hart InterCivic and ES&S did not provide a build environment.

- Both Hart InterCivic and Premier systems initially prohibited the team access to the private reports of previous studies. The reasons for not allowing the team to review the reports remain unclear. After many weeks of negotiation with the Ohio Secretary of State, Premier allowed the team to review the unredacted California TTBR and Florida review reports on November 19th, 2007, and then only under the supervision of legal council in Harrisburg, PA and with restrictions on the way in which the team could take notes. This viewing occurred essentially as the review period was over, and thus was not useful for directing the group's activities, but only served to further confirm results we had found during the course of our investigations.

  Hart InterCivic provided no access to the unredacted reports.

  The lack of access to private reports presented real barriers. For example, the inability to read the private report directly prevented us from confirming or denying issue 14 of the California TTBR Hart Source Code Report[8] (see Section 19.1.1). Further, the lack of access unnecessarily required the teams to rediscover many (often mundane) aspects of the evaluated systems. This took valuable time away from other important technical and reporting activities, and reduced the scope and depth of this report.

[8] Srinivas Inguva et al., *Source Code Review of the Hart InterCivic Voting System*. University of California, Berkeley under contract to the California Secretary of State, July 20, 2007 ⟨URL: `http://www.sos.ca.gov/elections/voting_systems/ttbr/Hart-source-public.pdf`⟩.

- Absentee ballot scanners, especially high-speed scanners, have not been reviewed in either the Hart or Premier systems. Due to the volume, absentees are special risk for integrity of election. As a point of reference, in a typical California county over 30% of the votes are cast absentee.

  High speed scanning uses different technology than normal paper scanning. Thus, the following hardware and software components were not evaluated:

  Premier: The AV-OS CC (Central Count) high speed optical scanner device has not been evaluated. The technology upon which this device is built is substantively different than the AV-OS PC, and therefore its absence limits the scope of this review.

  Hart: While we have partially evaluated the Ballot Now software, we have not been able to test the BNIP (Ballot Now Image Processor) other than with image files. For this reason, we have not been able to collect sufficient information to identify how the central count interacts with the BNIP server, scanner, and other clients within the network.

2.5 Acknowledgments

We would like to acknowledge the exceptional assistance we received in performing this review and documenting its results. We would particularly like to thank Michael Krippendorf and Terry Dick at the Ohio Secretary of State's office for the tireless efforts in marshaling the vast body of documents, source-code, and equipment that this review required. This review simply would not have been possible without their constant and seemingly inexhaustible support. Katherine Thomsen at the SOS office was instrumental in getting answers back from the counties, as well as helping us locate policies, procedures, etc.

We are grateful for the assistance of Dell in prioritizing our requests for equipment. With the help of so many individuals, especially Ken Reneau, John Daley, and Pam Deivernois, conducting our evaluation within the strict time constraints was made possible. Additionally, we would like to thank Fortify Software, Scientific Toolworks, and Microsoft Developer Network Academic Alliance for providing their commercial tools free of charge.

We would also like to thank David Richardson, John Hanold, and Alicia Bunnell at the Pennsylvania State University's Office of Sponsored Projects, Raj Acharya and Corry Bullock of the Department of Computer Science, David Wormley and John Mason of the College of Engineering, and the faculty and students of the Systems and Internet Infrastructure Security Laboratory whose support was essential to our success.

We owe a huge debt of thanks to Deborah Fisher and Mark West of the University of Pennsylvania, whose outstanding (and tireless) contracting and project management support proved instrumental in making the Penn team's participation possible. We are also grateful to our fellow faculty and students of the Penn Distributed Systems Laboratory for their generosity and patience while our normally shared space was converted into a secure facility.

Finally, we would like to thank the unwavering support of the Ohio Secretary of State Jennifer Brunner and Assistant Secretary of State Chris Nance during this process.

CHAPTER 3

THREAT MODEL

The first step in security analysis of a system is to define the *threat model*.[1] The threat model for a system describes the goals an attacker might have (e.g., to manipulate the vote count), the types of attackers that might attempt to attack the system (e.g., voters, poll workers, etc.), and the capabilities available to each type of attacker. It is equally important to describe the threats that are out of scope for the analysis.

There are many possible attacks that exploit social or physical processes that are out of scope for this report. We consider these attacks inasmuch as they interact with technical procedures, and then only where they are used to compromise the integrity of the election equipment under review.

3.1 Reference Model

In order to simplify the analysis, we assume a common reference model that distills the essential features of voting systems. The system consists of the following components. In the polling place:

- Management stations (MS)
- Electronic Pollbook (*optional*)
- Direct recording electronic (DRE) voting machines, attached to Voter Verified Paper Audit Trail (VVPAT) printers
- Paper ballot optical scanners (opscan)

At Election Central (the election headquarters in the local county):

- An election management system (EMS)
- High-speed paper ballot optical scanners (e.g., for absentee votes)

The election can be thought of as proceeding in three stages (for simplicity, we ignore early voting):

1. *Pre-voting*: Before election day, election officials use the EMS to set up the election. They generate the ballot definition(s) and record them onto media for distribution. During this stage, voting machines are also prepared and distributed to polling places. Valid voter logbooks are created as physical books or loaded onto electronic poll books.
2. *Voting*: On election day, voters arrive at the polling place, sign into a physical or electronic voter logbook, are verified as being permitted to vote, and cast their ballots.

[1] Some of the material in this chapter is derived from the threat model presented in the California TTBR reports. We appreciate the authors' permission to use that material.

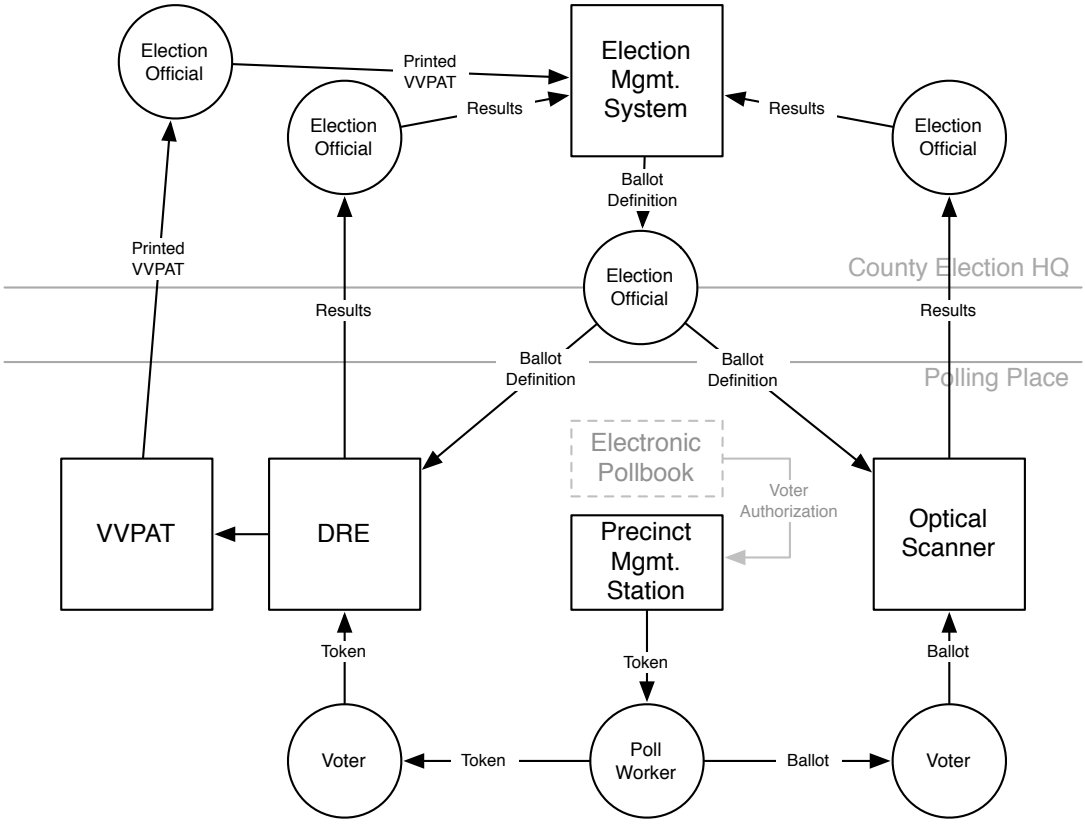

Figure 3.1: Reference architecture

3. *Post-voting*: After the polls are closed, the votes are tallied, and the results are certified. The Ohio Secretary of State's directive 2007-29[2] specifies how the official canvass must occur within E+11 to E+15 (the 11th and 15th days after the election (ORC 3505.32)).

The relationship between these components is shown in Figure 3.1.

3.1.1 Pre-Voting

In the pre-voting phase, election officials need to:

- Create the election definition.
- Print the paper ballots used for optical scan systems.
- Reset the local voting equipment and potentially load the election definitions.
- Identify the voters permitted to vote in each precinct.
- Distribute the local voting equipment and voter rolls to the polling places.

[2]Ohio Secretary of State. (2007). Directive 2007-29: Official Canvass and Report Forms for November 6 General Election. Retrieved November 10, 2007, from `http://www.sos.state.oh.us/sos/ElectionsVoter/directives/2007/Dir2007-29.pdf`

There is some variation among voting system vendors, but in general the EMS is used to create the ballot definition files. These are then loaded onto some memory card/cartridge and/or directly onto the voting machines. The vote counters in the machines are reset, the internal clocks are set to the correct time, and the machines are then shipped out to the local polling places or provided to poll workers to be hand-carried to the polling place. The ballot definition files must be protected from tampering and so memory card/cartridges are generally distributed with some physical security measures, either by sealing them into the local equipment at the central office or by distributing them in a sealed package. Seals may take the form of tamper-evident tape or of individually numbered metal or plastic loops that, once installed, can only be removed by cutting them.

3.1.2 Voting

Once the polls open on election day, voting can begin. The exact details of the voting phase differ with the local procedures, technology, and equipment manufacturer, but there is a fair amount of commonality across manufacturers within a given technology (DRE, opscan).

Optical scan machines. Opscan voting can most easily be thought of as machine-counted paper ballots. All the procedures here could be replicated by humans with appropriate audit controls.

When an opscan voter enters the polling place, and is verified as permitted to vote, he or she is given a blank paper ballot. He or she marks the ballot with a pen or pencil and the ballot is then mechanically counted with an optical scanner. This can be done either locally or at the county headquarters. In the local case, the precinct has a scanner which counts the ballots as they are inserted. In general, the voter personally inserts the ballot into the scanner. Precinct-based optical scanners can detect "overvoting" and "undervoting" and reject such ballots, giving the voter an opportunity to correct the error. At the end of the day, the scanner's electronic records are then sent back to the county for aggregation with records from other precincts. The paper ballots are also sent back, for auditing and recounts.

With central counting, untabulated ballots in their original ballot box are sent back to Election Central where they are tabulated with a high-speed scanner under the supervision of election officials. Central and precinct-based tabulation may be mixed in a variety of ways. Central tabulation is naturally used for absentee ballots and can also be used for audits and recounts of precinct-cast ballots, whether or not they were originally tabulated in the precinct.

DRE machines. DRE voting is fundamentally different from opscan voting. Instead of entering their vote on paper, the voter uses a computer-based *graphical user interface* (GUI). Once the vote has been "cast," an electronic record of the vote is stored locally, in the DRE machine (and, in the case of the Hart system, a copy is also stored in the management station). At the end of the day, these electronic records may either be extracted from the DRE machines on memory cards, or the DREs themselves may be transported to Election Central. In any case, the EMS will collect the electronic records from each precinct and will tabulate them electronically.

As with opscan voting, in DRE voting, the voter enters the polling place and first establishes his or her eligibility to vote. However, some method must be used to limit authorized voters to casting only one vote. In all DRE systems in this study, the poll worker uses an administrative device to issue the voter a token of some sort. The voter may then take this token to any voting machine and vote once. With Premier and ES&S, the token is a hardware device. With Hart, it is a four-digit "Access Code."

Once the voting machine is activated, it takes the voter through each contest and allows him or her to select candidates. DRE machines automatically forbid overvotes (too many votes cast in a contest) but optionally not undervotes (too few choices cast in a contest). The voter is then presented with an opportunity to review his ballot and then commits to it ("casts" it), at which point it is recorded to local storage.

In Ohio, the *voter-verified paper audit trail* (VVPAT) is the legal ballot. On all of the machines under study, the VVPAT takes the form of a sealed printer with a continuous spool of paper attached to the DRE. Before the voter confirms his ballot, a summary is printed out on the printer and displayed through a glass or clear plastic window. Once the voter accepts or rejects the ballot, an appropriate indication is printed on the VVPAT record. When the voter casts the ballot, the VVPAT record is marked as accepted and scrolled out of sight. The intent of the controlled printer VVPAT is to prevent misuse; the reasoning is that because the paper scroll is held behind glass, it becomes more difficult for a voter to "stuff" additional ballots into the machine or to take the record of their vote home, incorrectly, as a "receipt."

Once the election is over, the local results are transmitted to the Election Central, typically by shipping some form of removable memory device from the voting machine. The VVPAT paper rolls, perhaps still sealed in their printers, are also sent to the Election Central to be used in recounts.

Typical deployments. There are two common models for deploying this equipment in the polling place. In the DRE-only model, every polling place contains some number of DREs, most or all voters vote on the DREs, and there are no optical scan machines in the polling place. In the hybrid model, every polling place contains one optical scan machine and one or more DREs; voters may have the option whether to vote on paper ballots or using the DRE, or the DRE may be reserved for voters with disabilities and all others may vote on paper ballots.

3.1.3 Post-Voting

After the election is over the election officers need to do (at least) three things:

1. Tabulate the uncounted opscan and absentee ballots.
2. Produce combined tallies for each contest based on the records received from the individual precincts and the tallies of centrally counted ballots.
3. Perform the official canvass. This takes place in Ohio several days after the election (see previous note on Ohio statutes).

The first two tasks are relatively straightforward, though it's important to note that the second task is typically performed based on electronic records. In the most common case, the precincts send back memory cards containing the election results and those cards are added directly to the tally without any reference to the paper trail (except, of course, for centrally tabulated opscan and absentee ballots).

A manual recount compares the paper records for a given set of votes to the reported vote totals. In the case of opscan ballots, this means manually assessing each opscan ballot. In the case of DREs, it means manually assessing the votes on the VVPAT records. Note that while in principle the opscan tally may differ slightly from the paper ballots due to variation in the sensitivity of the mark/sense scanner, the VVPAT records should exactly match the DRE records.

3.2 Attacker Goals

At a high level, an attacker might wish to pursue any of the following goals or some combination thereof:

- Produce incorrect vote counts.
- Block some or all voters from voting.
- Cast doubt on the legitimacy of the election results.
- Delay the results of the election from becoming known.

- Violate the secrecy of the ballot.

An attacker might wish to pursue any of these goals either generally or selectively. For example, an attacker might target a certain subset of voters (e.g., registered Republicans, or voters who live within a certain geographic area) to attack. Also, the ability to determine how an individual voted can be used to enable vote buying or voter coercion, either individually or en masse.

3.2.1 Producing Incorrect Vote Counts

The most obvious attack on a voting system is to produce incorrect vote counts. An attacker who can cause the votes to be recorded or counted in a way that is different from how people actually voted can alter the outcome of the election. There are a number of different ways to influence vote counts, including:

- Confuse voters into voting differently than their intent.
- Alter the votes, within the computer, before they are recorded.
- Alter votes in the vote storage medium, after they were originally recorded.
- Corrupt the vote tabulation process.

Which attacks are feasible depends on the attacker capabilities.

3.2.2 Blocking Some or All Voters from Voting

Two classical techniques to influence election outcomes, regardless of the election technologies in use, are voter education / encouragement (i.e., "get out the vote") or voter suppression. If we limit the context to attacks specifically on voting systems, an attacker might attempt to suppress voting by making it very difficult for certain classes of voters to vote. For example, an attacker might:

- Arrange for some subset of machines to malfunction, possibly those in precincts in which voters of the opposite party are over-represented.
- Arrange for machines to selectively malfunction when voters attempt to vote in a certain way.
- Arrange for all the machines in an election to malfunction.
- Arrange for some or all of the machines to operate slowly. Such delays are often cumulative; a delay of only 10 or 15 seconds per voter can result in hours-long lines in crowded precincts.

The first two of these attacks are selective attacks and could be used to influence the outcome of an election. A more global attack is primarily useful for denial-of-service or to invalidate an election. These attacks would generally require tampering with the software within the voting machines, or in some cases damaging the physical voting equipment.

3.2.3 Cast Doubt on the Legitimacy of the Election Results

One of the most important requirements of an election is its ability to verify "correctness", i.e., to convincingly show that the voters' intent is accurately and completely captured in the election tallies. An adversary that can either manipulate the verification procedures, or even introduce *the perception of manipulation* will introduce concerns of illegitimacy. Generally, the attacker will use one of the following methods to introduce such doubts:

- Falsify evidence of malfeasance. For example, any attacker that can modify election data to suggest that more or less voters voted than was counted would introduce serious doubts about the validity of the election.

- Falsify seemingly legitimate, but uncounted, ballot material.

Doubt is often difficult to dispel. Lingering concerns often have a chilling effect on voters, and tend to color unrelated legitimate actives as well. Such concerns may continue for future elections.

One way to assess a system's resistance to attacks of this type is to consider a *skeptical third party*. A skeptical third party is an unbiased individual who hypothetically is going to make a judgment about the legitimacy of a given election. If an attacker can raise reasonable doubt in the mind of this party, then the attack has been successful. What constitutes "reasonable" in this case can vary from situation to situation, and thus requires some consideration by election officials and voters.

3.2.4 Delay the Results of the Election from Becoming Known

It is often sufficient for an attacker to simply frustrate the election process to meet their malicious goals. For example, while apparently not the work of intentioned attackers, the delays in the 2000 presidential elections had lingering effects on the presidency and called into question the competence of election officials and the accuracy of the election itself. Even though the election itself may have not been subverted, such results may be the intent of the attacker. There are any number of ways that such delays can be accomplished:

- Render the voting equipment temporarily unavailable or unusable, e.g., breaking the external interfaces to a DRE device, consume all the paper in a VVPAT printer.
- Delay the delivery of votes to tally devices.
- Improperly suggest election miscounts, thereby mandating often lengthy full or partial recount procedures. This is closely related to doubt-casting detailed below.

Note that procedures must speak to these issues. In particular, how the ballots (physically and recorded electronic form) are delivered is key to ensuring timely results.

3.2.5 Violating Ballot Secrecy

An attacker who cannot influence voting directly might still be able to determine how individuals or groups voted. There are two natural applications for this kind of attack:

- Vote buying / voter coercion
- Information gathering

In a vote buying or voter coercion attack, the attacker pays individual voters to vote in a specific way or threatens retribution if they do not. However, in order for such an attack to be successful, the attacker needs to be able to verify that the voter in fact voted the way he agreed to. Note that the buyer does not need to be able to determine with absolute certainty how a voter voted, but merely needs a high enough confidence that defection by bribed voters becomes unattractive. The primary benefit of traditional secret-ballot voting is that that it allows the voter to cast a vote in complete secrecy, defeating the attacker's ability to validate a voter's cast ballot and thus making vote buying or coercion unattractive.

Another possible reason to violate voter secrecy is to gather information on a large group of people. For example, an attacker might wish to determine which voters were sympathetic to a particular political party (but had not registered with it) and target them for investigation, surveillance, or even targeted mail or telephone solicitations. Alternately, an attacker might wish to publish a given voter's votes in an attempt to influence public opinion about them. In either case, ballot secrecy is required to block these attacks.

3.3 Potential Attackers

A voting system can be subject to attack by a number of different types of attackers with different capabilities. We consider the following broad classes of attackers, listed roughly in order of increasing capability.

- *Outsiders* have no special access to any of the voting equipment. To the extent that voting or tabulation equipment is connected to the Internet, modems, wireless technologies, and so forth, an attacker can mount network or software-based attacks. Outsiders may also be able to break into storage facilities and tamper with the equipment.
- *Voters* have limited and partially supervised access to voting systems during the process of casting their votes.
- *Poll workers* have extensive access to polling place equipment, including management terminals, before, during, and after voting.
- *Election officials* - have extensive access both to the back-end election management systems as well as to the voting equipment that will be sent to each precinct.
- *Vendor employees* - have access to the hardware and source code of the system during development and may also be called upon during the election process to assist poll workers and election officials. Some vendors use third-party maintenance and election day support whose employees are not tightly regulated.

Note that these categories are not intended to be mutually exclusive—a given attacker might have the capabilities of more than one category. For example, it might be possible to combine the limited physical access available to a voter with a network attack such as an outsider would mount. In addition, not all members of a given class have identical capabilities; an on-site vendor employee has a different level of access than an employee who only works with the source code. However, the purpose of this classification is to guide analysis, not to provide a complete taxonomy of attackers.

Attackers are also defined by their capabilities. Some attackers will have little sophistication and thus be limited by expertise. Others will have more sophistication and motivation and be able to commit both time and money to subvert an election. Strong adversaries such as organized crime can commit huge sums of money (for example to purchase and evaluate election equipment), seek outside expertise, and may be able to coordinate many attackers. The strongest adversaries are often *nation states*, entities who essentially have unlimited resources, expertise, and logistical support to attempt to subvert an election.

A particular focus of security analysis is *privilege escalation*. In many cases, one participant in the system is forbidden to perform actions that are normal for another participant in the system. A key feature of a secure design is enforcing such restrictions. For example, a voter should only be allowed to vote once, but poll workers are in charge of allowing people to vote and therefore must be able to authorize new voters to access the system. A voter who was able to use legitimate access to the voting terminal to acquire the ability to authorize new voters would be an example of privilege escalation. Similarly, poll workers are entrusted with maintaining the integrity of their polling place. If a poll worker was able to use authorized access to one polling place to influence or disrupt the voting equipment located in other polling places, that would be another example of privilege escalation.

3.3.1 Outsiders

An outsider to the system has no authorized access to any piece of voting equipment. They may be completely outside the system or may be physically present (perhaps as an observer) but not able to physically

touch the equipment. Such an attacker has limited capabilities in the context of this review. They might, for example, enter the polling place with guns and forcefully manipulate the voting systems (at least, until the police arrive). This sort of attack is explicitly out of our scope, although the "booth capture" problem is very much a real concern outside the U.S.

Outsiders may have the power to mount network- or malware-based attacks. Both election management systems and the development systems used by the voting vendors typically run on general purpose operating systems—Microsoft Windows in the case of all the systems in this review. If those machines are connected to the Internet or connected to machines which are occasionally connected to the Internet or the outside world in any way (laptops are a popular channel), an attacker might manage to infect the systems and thereby alter the software running on the election management systems or even the polling place systems. In that case, individuals anywhere in the world may have the opportunity to attack the voting system.

Outsiders may also have the power to physically tamper with voting equipment. In many counties, voting equipment is stored unattended at the polling place overnight before the election. While polling places may be locked overnight, most polling places are low-security locations; they may be located at a school or church or public building or a citizen's garage. Consequently, an attacker who is willing and able to break into the polling place, either by surreptitiously picking the lock or by forcibly entering, can likely obtain unsupervised physical access to the voter equipment for at least several hours. This kind of attack does require the outsider to be physically present and take on some risk of discovery.

We note that an outsider may be able to impersonate other roles in the system, such as a vendor representative or an election official. As an example, an outsider might mail CDs containing a malicious software upgrade to the election official in packaging that closely resembles the official packaging from the vendor. In another example, an outsider might be able convince a precinct leader that an attacker was legitimate simply by showing up with some forged documents and "looking honest". Note that these and other *social engineering* attacks are outside the scope of this review.

3.3.2 Voters

It is very easy to become a voter in Ohio and so it is expected that any attacker who wants to can acquire a voter's capabilities in at least one precinct. Unlike an outsider, a voter has physical access to a voting machine for a short period of time. That access is partially supervised, so that we would not expect a voter to be able to completely disassemble the voting machine. However, in order to preserve the secrecy of the ballot, it is also partially unsupervised. The details of the level of supervision vary to some extent from machine to machine and from county to county. In particular, the difference between optical scan and DRE is relevant here.

Voter based attacks are often some of the most serious, as access is necessarily given to individuals with unknown motivations. Moreover, such access can legally only be partially monitored.

Optical scan machines. In optical scan voting, the voter marks the ballot himself but the ballot is simply a specially crafted piece of paper. The voter then inserts the ballot into the optical scan equipment, typically under the supervision of the poll worker. Ordinarily, poll workers would be observing voters as they feed their ballots into the scanner. Therefore, the voter probably cannot tamper with the scanner without being detected. This does not entirely preclude voter access to open I/O ports on the scanner but does make it more difficult. The most likely avenue of voter attack is by the ballot itself. For example, a voter might attempt to have multiple ballots recorded or might mark maliciously crafted patterns on the ballot intended to subvert the scanner. Such specific patterns could also be used to trigger a dormant "Trojan horse" (i.e., activate pre-installed malicious software) to induce the machine to begin cheating. Such a compromised machine might otherwise behave correctly.

DRE machines. In DRE voting, by contrast, the voter has mostly unsupervised access to the voting terminal for a significant period of time. The front of the terminal is deliberately hidden by a privacy screen in order to protect voters' secrecy and therefore the voter has the opportunity to mount a variety of attacks. Any section of the machine that is accessible to a voter inside the privacy screen, including buttons, card slots and open I/O ports, must be assumed to be a potential point of attack. The voter also has an opportunity to input substantial amounts of data to the system via the official user interface. It may be possible to use this interface to compromise the machine.

In all the systems studied here, the DRE terminal requires some kind of token to authorize a voter to vote. In the ES&S and Premier systems this is a hardware token and in the Hart system it is a four-digit access code. Any voter who has access to these tokens is also a potential attacker and might attempt to subvert the token or substitute the original token with a new one.

It's important to note that the privacy afforded to voters by voting in a polling place is intended to be mandatory, but there are many steps a voter may take, while being bribed or coerced, to violate this privacy. This may include the use of cell-phone cameras, the placement of identifying marks on paper ballots, or the placement of unusual voting patterns or specific write-in votes on any voting system.

Because any United States citizen and Ohio resident is potentially a voter in Ohio, it must be assumed that any attack which requires only voter access is practical.

3.3.3 Poll Workers

Local poll workers have a significant capability that voters do not have: they have legitimate access to the management capabilities of the equipment. For example, the poll worker has the ability to authorize voters to vote. Note that although in principle this may give the poll worker opportunities for malfeasance, this risk may be mitigated by procedural controls. For example, a poll worker who controls the management station can in principle authorize a voter to vote an arbitrary number of times simply by issuing multiple tokens. However, polling places would normally have procedures in place to block or at least detect such attacks: because poll workers must perform their duties in public view, such malfeasance might be noticed by other poll workers or other voters; moreover, if at the end of the day there were more votes cast than registered voters who signed in, that would indicate a problem. Purely technical means may, alone, be insufficient to prevent such attacks.

Colluding poll workers represent another serious set of concerns; if a precinct has more than one attacker poll-worker, they may be able to circumvent many of procedures intended to detect problems or misuse or manipulate the election equipment. In the extreme, if all poll workers in a precinct were in fact acting maliciously together, then they would have full freedom to ignore many procedures or perform fraud with significantly decreased fear of detection.

Depending upon county practices, poll workers may also have long-term unsupervised access to voting equipment. In some counties, voting equipment is stored in the houses or cars of individual poll workers prior to the election. For example, some counties provide the chief poll worker at each polling place with DREs or opscan machines to store and deliver to the polling place. Even counties that deliver DREs and opscan machines by commercial transport may provide the chief poll worker with other equipment (smartcards, smartcard activation devices, management consoles, etc.) before the election. And even if equipment is stored in a secured polling place, dual controls may not be in place to prevent individual poll workers from accessing the area on their own. This provides a number of opportunities for tampering with equipment. Many pieces of equipment include seals to detect such tampering, but each system must be individually analyzed to determine whether these seals are effective and whether they protect all the relevant access points (see Section 3.5.2).

It must be noted that poll workers are primarily volunteers and are subject to extremely minimal security screening, if any is performed at all. In many counties the need for poll workers is so great that any registered voter who calls and offers to serve sufficiently far in advance is almost sure to be hired, and poll workers are often allowed to request to serve at a particular precinct. In practice, we must assume that any attacker who wants to be a poll worker can do so. Therefore an attack which requires poll worker access to a particular precinct is quite practical.

3.3.4 Election Officials

County election officials and county staff have three significant capabilities that poll workers do not have:

- Access to functionality of local voting equipment which may be restricted from poll workers.
- Access to large amounts of local voting equipment between elections.
- Access to the back-end election management system used for equipment management, ballot creation and tabulation.

For reasons of administrative efficiency, this access might be unsupervised or only loosely supervised, depending upon county practices. Such supervision is often more difficult because of a lack of very specialized skills. An observer who does not deeply understand the software and hardware will have difficulty determining the difference between normal and malicious behavior.

The first two capabilities imply greater ability to mount the sort of attacks that poll workers can mount. An election official with access to the warehouse where voting machines are stored might be able to compromise all the equipment in a county rather than merely all the machines in a precinct. In addition, the procedures for some equipment require that they be sealed—for example, the memory cards or results cartridges may be sealed inside the machine—before they are sent to the precincts. Because this sealing happens under the supervision of election officials, those officials might be able to bypass or subvert that process.

The third capability is wholly unavailable to the local poll worker. The back-end election management systems typically run on general purpose computers which are used by the election officials. If those systems are subverted they could be used to mismanage (compromise) polling place voting equipment, create fake or incorrect ballots, and to miscount votes. This subversion could happen in at least three ways. First, an attacker can use their legitimate access to mount an attack. For example, the software may offer an opportunity for election official to make "corrections" to vote tallies. Such "corrections" might be incorrect. Second, an election official might find a way to defeat the technical access controls in the election management software and tamper with its vote records.

Third, the official could directly subvert the computers on which the software runs. It is a truism in computer security that if an attacker has physical access to a general purpose computer he can eventually gain control of it. This may be achieved in a number of ways ranging from software attacks to directly compromising the system hardware, but in most cases is quite straightforward. The clear implication of this fact is that if election officials have unsupervised access to the election management systems, the integrity of those systems is provided purely by the integrity and honesty of those officials, not by any technical measures.

3.3.5 Vendor Employees

Finally, we consider attackers in the employ of the vendors. Such attackers fall into two categories: those involved in the production of the hardware and software, prior to the election, and those present at the polling place or Election Central warehouse during an election. An individual attacker might of course fall into both categories.

An attacker involved in the development or production of the software and hardware for the election system has ample opportunity for subverting the system. He or she might, for example, deliberately insert malicious code into the election software, insert exploitable vulnerabilities or back doors into the system, or deliberately design the hardware in such a way that it is easily tampered with. Such attacks are extremely hard to detect, especially when they can be passed off as simple mistakes, since mistakes and bugs are extremely common in large software projects and good-faith mistakes may be very difficult to distinguish from deliberate subversion. Note that for such an attack to succeed it is not necessary for the attacker to arrange for uncertified software or hardware to be accepted by election officials or poll workers. Rather, the vulnerabilities would be in the certified versions. Neither the current certification process nor this review is intended or able to detect all such vulnerabilities.

A vendor employee may also be present in the county to assist election officials or poll workers. For example, the employee might be present at Election Central during or after the election to help election officials, either answering questions or helping to fix or work around malfunctioning equipment. A vendor employee might also be present at Election Central to help install or maintain the voting system or to train county staff or poll workers. A vendor employee might even be present at the polling place or available by phone to assist poll workers or answer questions. Such an attacker would have access to equipment comparable to that of poll workers or election officials, but would also have substantial freedom of movement. Because they are being asked to correct malfunctions and install and configure software, activities which are actually intended to subvert the equipment are much less likely to be noticed. To the extent to which the systems have hidden administrative interfaces they would presumably have access to those as well. Finally, vendor employees pose a heightened risk because they may have access to multiple counties which use the vendor's equipment, and the ability of one individual to tamper with voting equipment in multiple counties increases the scope of any potential subversion.

3.4 Types of Attacks

We can categorize attacks along several dimensions:

- *Detectable vs undetectable.* Some attacks are undetectable no matter what practices are followed. Others are detectable in principle, but are unlikely to be detected by the routine practices currently in place; they might be detected by an in-depth forensic audit or a 100% recount, for example, but not by ordinary processes. Still other attacks are both detectable and likely to be detected by the practices and processes that are routinely followed.

  The potential harm caused by the former two classes of attacks surpasses what one might expect by estimating their likelihood of occurrence. The mere existence of vulnerabilities that make likely-to-be-undetected attacks possible casts doubt on the validity of an election (see Section 3.2.3). If an election system is subject to such attacks, then we can never be certain that the election results were not corrupted by undetected tampering. This opens every election up to question and undercuts the finality and perceived fairness of elections. Therefore, we consider undetectable or likely-to-be-undetected attacks to be especially severe and an especially high priority.

- *Recoverable vs unrecoverable.* In some cases, if an attack is detected, there is an easy way to recover. In contrast, other attacks can be detected, but there may be no good recovery strategy short of holding a new election. In intermediate cases, recovery may be possible but expensive. For example, recovery strategies that involve a 100% manual recount impose a heavy administrative and financial burden and will introduce delays in the finalization of the election results.

  Attacks that are detectable but not recoverable are serious. Holding a new election is an extreme remedy, often requiring contentious litigation. There are scenarios where a new election cannot fairly

be held: for example, redoing one county's part of a statewide race once the other counties' results become known would be unfair to the other counties.

In addition, unofficial election results, once announced, tend to take on certain inertia and there may be a presumption against abandoning them. When errors are detected, attempts to overturn election results can potentially lead to heated partisan disputes. Even if errors are detected and corrected, the failure again has the potential to diminish public confidence. At the same time, detectable-but-not-recoverable attacks are arguably not as serious as undetectable attacks: we can presume that most elections will not be subject to attack, and the ability to verify that any particular election was not attacked is valuable.

- *Prevention vs detection.* Often, there is a tradeoff between different strategies for dealing with attacks. One strategy is to design mechanisms to prevent the attack entirely, closing the vulnerability and rendering attack impossible. When prevention is not possible or too costly, an attractive alternative strategy is to design mechanisms to detect attacks and recover from them.

  Most election systems combine both strategies, using prevention as the first line of defense along with detection as a fallback in case the preventive barrier is breached. For example, we attempt to prevent or deter ballot box stuffing by placing the ballot box in the open where it can be observed and charge poll workers with keeping an eye on the ballot box. At the same time, we track the number of signed-in voters, account for all blank ballots, and count the number of ballots in the box at the end of the day to ensure that any ballot box stuffing that somehow escapes notice will still be detected. This combination can provide a robust defense against attack.

- *Wholesale vs retail.* One can distinguish attacks that attempt to tamper with many ballots or affect many voter's votes ("wholesale" attacks) from attacks that attempt to tamper with only a few votes ("retail" attacks). For example, attacks that affect an entire county or a large fraction of the precincts within a county are typically classified as wholesale attacks, whereas attacks that affect only one voter or one precinct are typically classified as retail attacks. This is a useful distinction because retail fraud is likely to have less impact on the outcome of the election.

- *Casual vs sophisticated.* Some attacks require little technical knowledge, sophistication, advance planning, resources, or access. For example, stealing an absentee ballot out of someone's mailbox is a classic low-tech attack: anyone can execute such an attack, and no special qualifications or skills or organization is needed. In contrast, other attacks may require deep technical knowledge, specialized skills or expertise, considerable advance planning, a great deal of time, money, or other resources, and/or insider access. This study examines both sophisticated technical attacks as well as casual low-tech attacks.

When discussing vulnerabilities and potential attacks on the vendor's voting system, where possible we identify these distinctions to enable readers to form their own judgments about the severity and impact of those attacks. Judgments about the probability of an attack or the impact on the election are outside the scope of this review.

3.5 Procedures

Election *procedures* are best practices mandated locally (within the precinct), at the county level, or at the state or federal level. Sometimes these practices are codified in statutes, others are documented in training manuals, and still others are exchanged via word of mouth. These practices are intended to ensure that the election is carried out correctly and securely. In short, procedures define how the election is to be run

by humans. Thus, from a practical standpoint, procedures are often as important as the technical security features of the election systems.

An important consequence of human involvement is that any procedure that is performed even on a less than regular basis will almost certainly not be performed every time. In many cases, where the purpose of the procedure is not apparent to the individual performing it and doing something else is more convenient, it may be true that it is not often at all. Any procedure, no matter how well crafted should be viewed at best an imperfect mitigation, and at worst a mere suggestion. In light of this, those setting procedures should carefully consider what happens when a procedure is not followed, for example, once, occasionally, or frequently.

There are many types of procedures. One useful distinction is to view them as verifiable or unverifiable. Verifiable procedures provide some evidence that the procedure was followed, e.g., a chain-of-custody form is commonly used to ensure that election equipment is never in the possession of an unauthorized party. Such verification evidence is good for detecting bad practice. Thus, reviewing procedural evidence before, during, and after and election is good policy. Verifiable procedures often have the desirable side-effect that the individuals will be more likely to follow them; people are more likely to follow the rules if they know they are being watched.

Unverifiable procedures are those for which no evidence can be generated. For example, no validatable procedure exists to ensure that the poll workers watch a ballot box; one can observe and attest to a poll worker sitting next to the box all day, but that does not ensure that the box is indeed watched. However, if the poll worker was the only one who has access to the box and they physically inserted the ballots into the box, then box access would be procedurally verifiable. Obviously, it is highly desirable to find procedures that are verifiable.

3.5.1 Ohio Procedures on Information Technology and Networking

There are several procedures mandated by the state of Ohio that speak directly to this review. The use of modems or other networking technology for any election operations is forbidden except where explicitly allowed by Ohio statute. Under ORC 3506.23 added in 2006, "A voting machine shall not be connected to the Internet." However, that says specifically, "Internet" and is silent about telephone lines, LANs, WANs, etc. Also, the definition of "voting machine" in ORC 3506.01(E) says "'Voting machines' means mechanical or electronic equipment for the direct recording and tabulation of votes.", which seems to refer directly to DREs and optical scanners but is not as clear when applied to tabulation equipment. Ohio Secretary of State directive 2005-23 provides further guidance on Internet usage and connectivity.[3] Specifically, Dir. 2005-23 identifies the following procedures and prohibitions:

- **General Internet Access and Networking**

    1. No "Election Information System" is permitted to connect to any computer network or to the Internet, including for (1) setting up elections; (2) defining ballots; (3) receiving voted data from polling places; or (4) tabulating data related to ballots or votes.
    2. Two exceptions are (1) using local networks for creating or uploading memory cards, ballot definitions, precinct results, etc. and (2) to download software upgrades, repairs or updates (board of elections has to apply for a waiver).
    3. Under no circumstances may a polling place device be connected to an outside polling place.

[3] Ohio Secretary of State. (2005). Directive 2005-23: Telecommunications: General Internet Access and Networking, Downloading Software and Modem Access. Retrieved November 10, 2007, from `http://www.sos.state.oh.us/sos/electionsvoter/directives/2005/mainDocs/Dir2005-23.pdf`

- **Downloading Software**

    1. No one may download and install software on any elections-relevant system without written authorization of the directors of the board of elections.
    2. All such software must be virus checked by a current piece of virus-detection software.

- **Modem Access**

    1. No modem transmission is allowed without a security plan approved by the Director of Elections. This plan must specify:
        (a) steps to prevent signal interception, imitation or replacement
        (b) procedures to verify the identity of anyone seeking to transmit information.
    2. A majority vote of the Board of Elections is required to authorize modem use.
    3. No modem maybe used in an "auto-answer" mode without a remote identification system, "challenge-response" dial-back or something equivalent as authorized by the Secretary of State.
    4. No VOIP connections allowed at all.
    5. Steps shall be taken to protect against accidental transmission of confidential information.

3.5.2 Mechanisms for Tamper Sealing

Virtually every election system makes extensive use of tamper seals as a part of its security design. This section presents a brief summary of how these mechanisms work and the level of sophistication an attacker must have to violate them.

Tamper resistance refers to the ability of a system to deter an attacker from gaining access to the system. This could take the form of software controls (e.g., careful limits on the protocols spoken across networks) to procedural controls (e.g., the use of strong passwords) to hardware mechanisms (e.g., strong locks). Tamper resistance generally refers to the amount of time, effort, and/or sophistication required to overcome a security mechanism.

Tamper evidence is the converse of tamper resistance, representing the extent to which an attacker's attempt to overcome a tamper-resistance mechanism leaves behind evidence of that tampering. For example, in a typical home, a burglar could easily break in by putting a brick through a window. A plate-glass window offers little tamper resistance, but strong tamper evidence (i.e., the effort that would be required by a burglar to reinstall a broken window and clean up the broken glass is quite significant). By contrast, a typical door lock is far more tamper resistant than the glass window. However, if a skilled burglar can pick the lock, there will be little or no evidence that it had been picked.

In the context of voting systems, there are a number of tamper sealing mechanisms commonly used:

Key locks To prevent access to memory cards or sensitive machine ports, many voting machines place a plastic or metal door in front of these ports, using a key lock. Assuming the keys are suitably controlled (and unauthorized duplication is prevented), attackers would be prevented from accessing the protected ports. Of course, if bypassable lock mechanisms are used, or if access to the locked compartment can be gained without opening the lock, then the locks will offer neither tamper resistance nor tamper evidence as has been observed with both Diebold[4] and Nedap/Groenendaal[5] voting systems.

[4] Ariel J. Feldman et al., 'Security analysis of the Diebold AccuVote-TS voting machine'. In Proceedings of the 2007 Usenix/ACCURATE Electronic Voting Technology Workshop. Boston, MA, August 2007.

[5] Rop Gonggrijp and Willem-Jan Hengeveld, 'Studying the Nedap/Groenendaal ES3B voting computer: A computer security perspective'. In Proceedings of the 2007 Usenix/ACCURATE Electronic Voting Technology Workshop. Boston, MA, August 2007.

Wire loops Many voting machines have adopted a mechanism commonly used with traditional ballot boxes—the use of holes through which metal or plastic wires loops may be fitted. These seals have much in common with standard "tie wraps;" once fitted, the wire loop cannot be loosened; it can only be physically cut off. Like key locks, the loops, when sealed, lock a physical door in place. In common election practice, these seals are stamped or printed with individual serial numbers. Those numbers are then logged when the seals are installed and again when they are cut to detect the substitution of an alternate seal. An attacker with simple tools may be able to clone the serial numbers from old wire loops to new ones without detection.[6]

Tamper-evident tape Adhesive tape can be printed with numbered labels in the same fashion as wire loops. Typically, two different adhesives are used, such that if/when the tape is removed, part of the label will remain stuck to the lower surface while part of the label will be removed with the tape. Technology of this sort is commonly used for automobile registration and inspection stickers. Anecdotal evidence suggests that it may be possible to peel back the tape and replace it without this being easily observable.[7]

A recent study of tamper seals considered 244 different seal designs and found that "the majority could be defeated—removed and replaced without evidence—by one person working alone within about two minutes and all of these devices could be thwarted within about 30 minutes".[8] Needless to say, such seals cannot be counted on, alone, to provide significant security protections for electronic voting systems.

Of course, these mechanisms can be augmented through other procedural means, including requiring multiple people to be present when machines are handled or maintaining video cameras and other locks on the storage areas of the elections warehouse. Johnston also recommends that officials have genuine seals in their hands to compare against the seals being inspected.[9]

The use of tamper-evident or tamper-resistant technologies, as such, must be evaluated in the broader context of procedures and policies used to manage an election. Weaknesses in these procedures cannot be overcome by the application of tamper-resistant / tamper-evident seals. Also, the attacker's motivation must also be considered. Perhaps the attacker does not care if an attack is evident, so long as it cannot be recovered from.

3.5.3 Chain-of-Custody Logs

It is common practice for every component (e.g., DRE, optical scan, paper ballots) used in an election system to have an associated Chain-of-Custody log. The log enumerates whenever the component changes hands, indicating the time and parties involved. It serves as an important forensics tool to define a list of suspects after tampering has been detected. However, its use is limited to forensics and serves limited utility if there is no reason to suspect tampering in the first place. As such, enforcing the use of the log is also questionable. The interpretation of a "change in custody" is often up to human judgment. Does it include every time a device is handed to another party while still in plain view of the original party? or maybe allowing another party to borrow the device for the evening is acceptable. Without complete video surveillance there proving custody log accuracy is difficult.

[6] Michael Shamos, Oral testimony before the Technical Guidelines Development Committee (TGDC), 2004.

[7] Aviel D. Rubin, *My day at the polls-Maryland primary '06*. `http://avi-rubin.blogspot.com/2006/09/my-day-at-polls-maryland-primary-06.html`, September 2006.

[8] Roger G. Johnston, 'Tamper-indicating seals'. American Scientist, 94 November-Decemeber 2006.

[9] Johnston (as in n. 8).

3.5.4 Recounting Optical Scan Ballots

The optical scan voting technology is often viewed as more secure because it leaves behind a hard copy record of voter choices. However, even these records are subject to the limitations of procedures. First and foremost, correcting foul play requires the recount of *all* physical ballots (an immense and often unpractical undertaking). If election tampering is detected at one polling place, there is a greater probability that it occurred elsewhere. Additionally, ballots returned from polling places are subject to tampering while in transit, or insecure protections could allow additional ballots to be placed in ballot boxes while at the polling place. Finally, if the election officials at the polling place itself are corrupt, there is no practical way of determining if the returned ballots are correct.

3.6 Software Engineering and Maintenance

One important contributor to the security of any system is the way in which the software is designed and developed. Standards for *software engineering* developed over the last 50 years require that a system undergo a rigorous process of requirements definition, structured design and review, and careful programming and testing. Like proper engineering leads to cars of higher quality, so too does better software engineering lead to more secure, robust software computer systems. Systems that are designed without this kind of careful design and implementation are almost certain to have flaws and security issues.

Developing a system is only the beginning of software quality. Managing the process of fixing bugs and adding functionality over time is equally important to maintaining security and reliability. This *software maintenance* process is again key to sustaining a robust and secure system. Failure to adequately manage these processes is most often fatal; each buggy or poorly thought out modification has a multiplicative negative effect on the quality of software. Furthermore, even if well designed, implemented, and maintained, software has a natural life-cycle over which the software can reasonably be maintained. Software that is maintained past its lifetime becomes unsafe, buggy, and often performs poorly or not at all.

Part II

Analysis of the Election Systems & Software, Inc. Voting Systems

CHAPTER 4

ES&S EXECUTIVE SUMMARY

This part of the EVEREST report evaluates the ability of the ES&S Unity EMS, iVotronic DRE, and M100/M650-based optical scan voting systems to conduct trustworthy elections. The review team was provided access to the ES&S source code and election equipment. The reviewers studied these materials in order to identify security issues that might be exploited to affect an election. As part of that analysis, the reviewers were asked to identify, where possible, practices that limit or neutralize the impact of discovered issues.

Our analysis suggests that the ES&S Unity EMS, iVotronic DRE and M100 optical scan systems lack the fundamental technical controls necessary to guarantee a trustworthy election under operational conditions. Exploitable vulnerabilities allow even persons with limited access – voters and precinct poll workers – to compromise voting machines and precinct results, and, in some cases, to inject and spread software viruses into the central election management system. Such compromises render the election result subject to subtle manipulations – potentially across election cycles. These vulnerabilities arise from several pervasive, critical failures of the ES&S system:

- *Failure to protect election data and software* – The firmware and configuration of the ES&S precinct hardware can be easily tampered with in the field. Virtually every piece of critical data at a precinct – including precinct vote tallies, equipment configuration and equipment firmware – can be compromised through exposed interfaces, without knowledge of passwords and without the use of any specialized proprietary hardware.

- *Failure to effectively control access to election operations* – Access to administrative and voter functions are protected with ineffective security mechanisms. For example undocumented "quality assurance" hardware tokens that bypass password checks are easily forged using inexpensive commodity devices such as palmtop computers. Central back-end election management software is vulnerable to attacks that exploit coding and design errors and that can be triggered from data sent from the field.

- *Failure to correctly implement security mechanisms* – Many of the most serious vulnerabilities in the ES&S system arise from the incorrect use of security technologies such as cryptography. This effectively neutralizes several basic security features, exposing the system and its data to misuse or manipulation.

- *Failure to follow standard software and security engineering practices* – A root cause of the security and reliability issues present in the system is the visible lack of sound software and security engineering practices. Examples of poor or unsafe coding practices, unclear or undefined security goals, technology misuse, and poor maintenance are pervasive. This general lack of quality leads to a buggy, unstable, and exploitable system.

We believe the issues reported in this study represent practical threats to ES&S-based elections as they are conducted in Ohio. It may in some cases be possible to construct procedural safeguards that partially mitigate some of the individual vulnerabilities reported here. However, taken as a whole, the security failures in the ES&S system are of a magnitude and depth that, absent a substantial re-engineering of the software itself, renders procedural changes alone unlikely to meaningfully improve security.

Nevertheless, we attempted to identify practical procedural safeguards that might substantially increase the security of the ES&S system in practice. We regret that we ultimately failed to find any such procedures that we could recommend with any degree of confidence.

A particular challenge in securing the iVotronic DRE terminal is the large number of precinct-based attack vectors whose exploitation must be prevented. Effective procedures that accomplish this, even if they existed, would be arduous indeed, and would likely substantially hamper poll workers in their duties, reduce the ability to serve voters efficiently, and greatly increase the logistical challenges of running an election.

The security failings of the ES&S system are severe and pervasive. There are exploitable weaknesses in virtually every election device and software module, and we found practical attacks that can be mounted by almost any participant in an election. For this reason, the team feels strongly that any prudent approach to securing ES&S-based elections must include a substantial re-engineering of the software and firmware architecture to make it "secure by design."

It may be possible to deploy a reduced subset of the ES&S hardware and software that excludes components that present the greatest risks. For example, a system that uses only centrally-counted optical scan hardware eliminates many of the threats of precinct-based attacks. We defer to the expertise of the Ohio election officials to determine whether it is possible to use a version of the ES&S system with reduced functionality in a way that presents an acceptable level of risk to the integrity of their elections.

CHAPTER 5

ES&S SYSTEM OVERVIEW

The following chapters detail the ES&S portion of the EVEREST review. Readers are directed to Part I of this report for a discussion of the review's goals and methodologies. Those readers not experienced in information security or Ohio election practices are also encouraged to read the threat model in Chapter 3. The content in that chapter is instrumental in gaining a detailed understanding of the substance and impact of the issues identified throughout.

The ES&S source code analysis and penetration testing exercises reported here were conducted in Philadelphia, PA by the University of Pennsylvania team and in Santa Barbara, CA by the WebWise Security, Inc. team. Team personnel are identified as authors in the front matter of this report. We frequently consulted with the project's procedural and document consultants and with the Pennsylvania State University team.

The issues and commentary described in this section of the report are *reflective of our analysis of the ES&S system only*. None of the included material should be considered to apply to either the Hart or Premier systems, nor should any statement be construed to be making any quantitative or qualitative comparisons of the different systems. Such comparisons are expressly outside the scope of the review, and we have not developed any opinions on the relative merits of any one system with respect to the others. Nothing in this review should be considered as a positive or negative commentary on electronic voting in general.

The remainder of this part of this report is structured as follows. We begin in the next section by giving a brief overview of the ES&S system and its use in Ohio elections. Chapter 6 broadly characterizes the security and reliability of the ES&S system by highlighting the most serious architectural and systemic issues encountered in the review. Chapter 7 catalogs various specific software vulnerabilities that we discovered. Chapter 8 outlines software quality and software engineering risks arising from the system's design and implementation. Chapter 9 describes a number of attack scenarios that demonstrate how the identified issues may be used in concert to subvert an election.

5.1 Architectural Overview

The election management system from Election Systems & Software (ES&S) prepares, collects, and tabulates elections using either paper ballots or touchscreen terminals (or a combination of both). It contains software for managing all stages of an election, as well as special hardware interfaces for configuring and retrieving results from the election devices. The high level architecture of the ES&S system is shown in Figure 5.1.

Unity is the election management software suite. It is comprised of a number of individual programs which interact with one another through shared data files (collectively referred to as the "*database*"). **Election Data Manager** is used to initialize the database with jurisdiction, voter, and candidate information. **ES&S Image Manager** and **iVotronic Image Manager** are used to design the appearance of paper and touchscreen ballots. The **Election Reporting Manager** is used to collect and tally election results, and

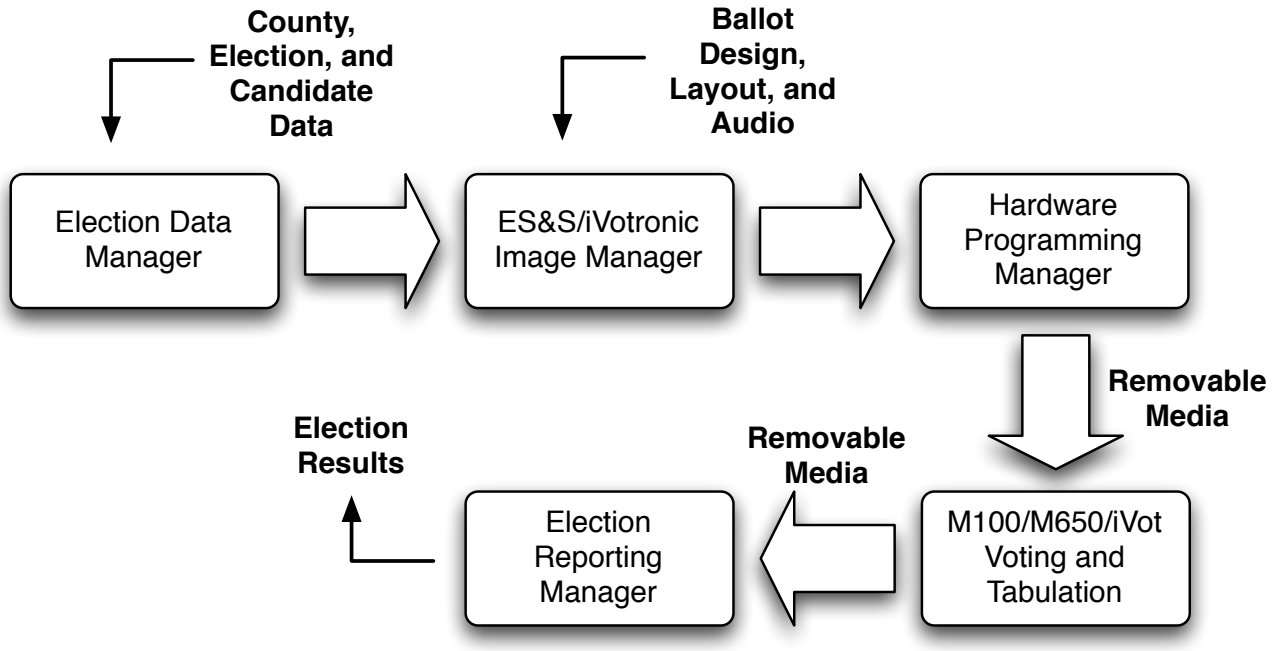

Figure 5.1: The high level overview of the ES&S Unity voting system with user input to each stage of the election.

Audit Manager is used to verify election results using audit data. All interaction between Unity and voting hardware is controlled by the **Hardware Programming Manager** program.

The **iVotronic** is a touchscreen direct recording electronic voting terminal (DRE). There are two distinct types of iVotronic terminals, distinguished by colored inserts along the sides: red *supervisor terminals* and blue *voter terminals*. Both types of iVotronic terminals are activated using special hardware tokens called **Personalized Electronic Ballots (PEBs)**, which are also used to store ballot definitions and election results. PEBs are typically programmed via a supervisor terminal at the start of an election, and read using either a supervisor terminal or a dedicated **PEB Reader** connected to the machine running the Election Reporting Manager at the end of an election. A PEB can be used in multiple iVotronics as long as they are qualified for the same election and polling place.

The voter iVotronics also use **Compact Flash** cards to store large ballots, audio ballots, and election result audit files. Connected to the iVotronic in Ohio is a printer which provides a voter-verified paper audit trail, known as the **Real-Time Audit Log** printer in ES&S documentation and source code. A separate **Communication Pack** is connected to iVotronic terminals at the start and end of elections to print zero count tallies and precinct results on a separate printer with removable paper.

The **Model 100** is a machine for scanning and validating/tallying paper ballots at a polling location. The Model 100 uses **PCMCIA Memory Cards** to hold ballot definitions and tallies.

The **Model 650** is a machine for batch scanning and tallying paper ballots at a central election office. The Model 650 uses **Zip Disks** to hold ballot definitions and election tallies.

The **AutoMARK** Voter Assist Terminal is used by disabled voters to fill out a paper ballot without additional assistance. The AutoMARK uses Compact Flash cards to load ballot definitions.

5.2 System Components

In this section we describe the different components of the ES&S voting system in greater detail. This background on the architecture and implementation will aid in understanding the vulnerabilities and attacks in subsequent chapters.

The ES&S system is made up of a number of custom and off the shelf general purpose computers. On top of these computers run several hundred thousand lines of source code in various pieces of software. It is important to keep in mind that, although they do not appear the same as your typical desktop or laptop computer, all the components of the ES&S system are fully programmable computers capable of running arbitrary software stored in easily modifiable memory. Therefore use of the term "firmware" to refer to the software controlling the hardware components of the ES&S system is somewhat misleading. The code running on the iVotronic or Model 100 is in no way less susceptible to bugs, tampering, or co-option than any other part of the Unity system. When discussing the code comprising the ES&S system we will use the term "software" to refer to code running both on desktop PCs and the specialized voting hardware.

5.2.1 Unity

Unity is the Windows-based software suite for managing elections. It contains tools for creating and managing election databases (Election Data Manager), designing the appearance of ballots (ES&S and iVotronic Image Managers, AutoMARK Information Management System), tabulating and reporting results (Election Reporting Manager). Additionally, there is a tool to audit the use of the other components of Unity (Audit Manager), and a tool for abstracting programming and communicating with the various hardware components used by several Unity components (Hardware Programming Manager). The various components of Unity communicate with each other indirectly through common files stored on the Windows filesystem.

Election Data Manager

The Unity Election Data Manager (EDM) is a Windows XP application which is used for creating and updating the election database used by the other software components of Unity. The database for each county is stored in a single file in the Windows filesystem known as the "Ballot Data File" (BDF).

Each BDF is logically made up of two different databases: the "County Database" which contains tables of data which does not change from election to election, and the "Election Database" which contains the data defining a single election. Reusing the County Database from election to election is encouraged through an import feature which copies from a prior election's BDF to a new one, and the initial County Database can be filled in from another county's BDF as well. The remaining data entry is done either through the Windows GUI, or by importing text files in a number of formats.

In addition to specifying the races, candidates, proposals and other data that will appear on the ballot for an election, EDM is also used to select the equipment which will be used to cast votes in the election and specify any policies particular to the jurisdiction which influence ballot design (e.g. candidate position rotation). Finally, centralized configuration of the iVotronic passwords (see below) is performed from within EDM.

Other modules of Unity read and parse the BDF directly, as well as using "Intermediate Interface Files" (IFF) and "Ballot Set Collection" (BSC) files produced by EDM in a process referred to as "merging the election database". While the ballot data file is updated automatically when any changes are made in EDM, the other files must be re-generated by merging the election again.

ES&S Image Manager

ES&S Image Manager (ESSIM) is a C++ Windows page-layout tool used to design optical scan paper ballots. It is also sometimes referred to by an older name, Ballot Image Manager (BIM), in documentation and source code. ESSIM uses the IFF and BSC files exported from EDM to obtain the information to be displayed on ballots, including both information for the voter (candidates, races, proposals, etc.) and information for the back-end optical scanners (ballot codes, etc.). The layout specific data is stored in a "Ballot Layout File". When ballot design is finalized, ESSIM exports another IFF file (with a different file extension) to be used in the Hardware Programming Manager, as well as PDF ballots to be sent to a printing company.

iVotronic Image Manager

The iVotronic Image Manager (iVIM) is a Java Windows application for designing text, graphical, and audio ballots for the iVotronic DRM. iVIM relies on a MySQL database server to store its settings, which can either be installed on the same machine as iVIM or accessed remotely. The main input to iVIM is an XML file exported by EDM, along with graphical templates bundled with iVIM. Once layout is complete, iVIM exports the ballots as an iVotronic Election Definition file and a folder containing bitmap images and file hierarchy for the iVotronic Compact Flash card. Audio ballots for ADA voting are not managed within iVIM. Instead, an HTML document listing the necessary recordings and filenames to save them as is produced along with the other output files.

Hardware Programming Manager

Hardware Programming Manager (HPM) is used to transfer ballot configurations from files produced by ESSIM and iVIM to the removable media used in the M100, M650, and iVotronic. It consists of a Windows application written in COBOL and a number of helper applications and shared libraries (DLLs) written in C and C++. The COBOL code focuses on overall program logic and the creation of a new election database, while the C and C++ code focuses on communicating with the PEB and memory card readers and performing specialized operations (e.g. cryptography). Additionally, HPM serves as the bridge between the Election Data Manager (where elections are defined) and the Election Reporting Manager (where elections are tallied and reported), since the latter is also a COBOL based application.

The election configuration in HPM is imported from the IFC file produced by ESSIM. Adjustments to the election database can be made within HPM prior to writing the removable media to be used in the iVotronic or ballot scanner. These changes are only seen within HPM and the Election Reporting Manager, and not reflected in EDM. If changes are made in EDM they must be propagated through ESSIM to HPM and be imported again. Finally, device-specific settings for the M100 and M650 optical scanners can also be made from within HPM.

HPM copies files from ESSIM directly to the PCMCIA SRAM cards and Zip disks for the M100 and M650 scanners. For the iVotronic, the Compact Flash folder produced by iVIM is copied using a helper application. Before the PEB can be programmed, a binary ballot file is created using another helper application. This file can either be encrypted or unencrypted depending on options set by the user. The PEB is then programmed using another command-line tool which copies this binary ballot file along with the EQC block to a unused Supervisor PEB (see Section 5.2.4). A red Supervisor iVotronic is connected to the computer running HPM over a serial port and used to communicate with the PEB for this programming.

Election Reporting Manager

The Election Reporting Manager (ERM) is used to collect and report results from an election. Like HPM, it is a COBOL Windows program which relies on several helper applications and libraries written in other programming languages to access the removable media containing votes. It uses the same directory of data files generated by HPM upon importing an IFC file, and creates a new election database file used to store results.

Election results are collected from M100 and M650 scanners directly using an external PCMCIA card reader and Zip drive. iVotronic results can be collected either from the Compact Flash card of each machine, or from the Master PEB used to close several iVotronics. The CF card is read directly using a USB card reader, while the Master PEB can be read using either a Supervisor iVotronic or PEB Reader connected to the serial port of the machine running ERM using the same command-line tool used by HPM.

Once retrieved from memory devices, the COBOL code processes the results files and records the official election results. ERM has features for producing election reports per contest, per precinct or summary results. Additional details that can be reported include the totals for each type of ES&S tabulation device.

Audit Manager

The Election Data Manager and ES&S Image Manager (as well as Audit Manager itself) contain a mechanism for logging use and modification of the Unity election database files to a Microsoft Access database. Audit Manager provides a way of viewing these logs.

Other components of Unity developed in languages other than Microsoft Access (iVIM, HPM, ERM) have no support for recording actions in the Audit Manager log. Additionally, Audit Manager relies on EDM and ESSIM to dutifully update the Access database; Audit Manager is incapable of detecting changes to critical files performed outside of Unity tools.

AutoMARK Information Management System

The AutoMARK Information Management System (AIMS) is a standalone Windows application used to configure the AutoMARK Voter Assist Terminal to read and mark optical scan ballots produced in Unity with EDM and ESSIM. Although bundled as part of the Unity suite, the AutoMARK and AIMS were not developed by ES&S, and are not tightly coupled with any of the other Unity programs or files.

AIMS contains an entire election database of its own, which can either be imported from EDM's BDF or re-entered using the AIMS interface. Documentation recommends the later approach be taken to avoid errors in the conversion. Once the election database has been created, visual, audio, and translated ballots are designed for each type of printed ballot. The final configuration is copied to a Compact Flash card to be inserted in the AutoMARK device.

5.2.2 iVotronic

The iVotronic is a touchscreen DRE based around an Intel 386 processor with 1 MB of SRAM. There are four internal flash memory devices. One of these holds the iVotronic firmware, and is memory-mapped directly into the address space of the CPU to avoid using the limited amount of RAM to hold instruction code. The other three flash devices provide redundant storage for ballot and vote data. At power-up, and periodically during operation, the contents of the three memories are compared to one another to detect any corruption.

The iVotronic does not run anything resembling a modern operating system. There is a single process started at boot which runs in an event (interrupt) driven loop. Multiple threads, memory protection, dynamic memory allocation, exception handlers and similar constructs are unavailable.

The primary user input device to the iVotronic is an LCD touchscreen which is wired to the CPU as a serial device. There are two other serial ports on the iVotronic, one connected to a standard DB9 serial port at the top of the iVotronic, and the other to an infrared transceiver. The external serial port is used to connect to the RTAL printer (see Section 5.2.3) or a Communications Pack (see Section 5.2.7) in the field, and to a computer running Unity HPM or ERM in the central election office.

The infrared serial port is used to communicate with the Personalized Electronic Ballot hardware tokens (see Section 5.2.4). The left side of the iVotronic case has a molded socket to hold a PEB allowing IR communication as well as activation of the iVotronic power switch through a magnetic reed switch. There are two IR transceivers, one on the inside of the PEB socket, and a second on the outer edge of the iVotronic case. These two transceivers are multiplexed onto the single serial port, and only one can be used at a time.

In addition to the internal solid-state flash memory, a Compact Flash slot is located next to the printer serial port. The inserted CF card is accessed using a third-party library which implements the necessary filesystem code.

There are two types of iVotronic terminals used in elections: red "Supervisor iVotronics" are used by poll workers or elections officials to administer the election, and blue "Voter iVotronics" are used by voters to cast their votes. Every iVotronic has a serial number which is programmed into a PIC microcontroller during manufacturing. Except for this serial number and the outward appearance, all iVotronics are functionally identical. The same firmware is used on both Supervisor and Voter iVotronics, with a serial number check at boot to determine which mode is used. In the next two subsections the unique aspects of each type are introduced.

iVotronic Supervisor Terminal

The red Supervisor Terminal is used to manage PEBs and the contents of their flash memory before, during and after elections. It plays a far more significant role in the Voter Activated Voting mode (which is not used in Ohio elections), where at least one Supervisor Terminal must be at every polling place.

In the Poll Worker Activated Voting mode used in Ohio, a Master PEB for each polling location is created from HPM using a Supervisor Terminal connected via null modem cable. Afterwards, each Master PEB is then cloned several times using a Supervisor Terminal (standalone from Unity) in order to produce the Supervisor PEBs needed while the polls are open (see Section 5.3). At this point, the supervisor terminal is no longer required for opening, closing, or tallying the election.

iVotronic Voter Terminal

The blue Voter Terminal is the iVotronic used by voters to cast their ballot. When a qualified Supervisor PEB is inserted the iVotronic prompts the poll worker to select the correct ballot to be voted on. The poll worker then removes the PEB and leaves the voter to make their decisions. Once the voter casts their ballot, the Voter Terminal goes into a low-power state and waits for the Supervisor PEB to be inserted again to cast another ballot.

If the poll worker inserts the Supervisor PEB while holding the "Vote" button above the touchscreen, a service menu appears. This menu allows various settings of the terminal to be adjusted, and also provides the interface for opening and closing the polls. While in the service menu, actions performed are logged to the RTAL printer.

5.2.3 Real-Time Audit Log Printer

The Real-Time Audit Log Printer (RTAL) is a continuous feed thermal printer manufactured by FutureLogic, Inc. specifically for use in DRE voting systems. It performs the function of VVPAT on iVotronic machines.

It is connected to the iVotronic by a standard 9-pin RS232 serial cable, and mounted behind a plexiglass window next to the iVotronic. The RTAL supports both ASCII text using several built in fonts and encodings and a bitmap mode used to print barcodes or other non-text content.

Unlike the printer in the Communications Pack (see Section 5.2.7), the RTAL collects the output internally. The RTAL only has a motor on the collection spool, making it impossible to rewind the audit log without jamming. A movable carrier holds the paper mechanism independently of the print head and display window, allowing for the appearance of a small backwards movement (approximately .5") and the ability for the print head to "back up" a few lines.

5.2.4 Personalized Electronic Ballot

The Personalized Electronic Ballot (PEB) is a palm-sized device containing a PIC microcontroller, 2MB of flash storage, a bi-directional infrared (IR) transceiver, and battery. The PEB is activated by a magnetic reed switch, and contains a magnet to activate the corresponding reed switch in the iVotronic PEB socket.

In addition to the flash storage, the PIC microcontroller contains a small amount of non-volatile storage which is "burned" to the PIC during the PEB manufacturing process. This area contains the PEB firmware, PEB firmware version number, PEB hardware revision number, PEB serial number, and 'PEB Kind' variable.

The microcontroller firmware implements the passive half of a very simple command/response protocol between the PEB and host over the PEB IR port using IrDA SIR (Serial Infrared). The host sends the PEB one of several commands along with an address and fixed mount of data depending on the command, and the PEB performs the command and returns a response and command-determined amount of data. The primary operation is reading and writing 128 byte blocks of the PEB's flash memory, or verifying the integrity of blocks using a cyclic redundancy check (CRC) stored with each block. In addition, the serial number, PEB kind, battery voltage can all be retrieved but not modified by the host device.

While all PEBs are internally identical in construction, they are discernible from one another by the read-only information burned in the PIC: their serial number, and more importantly by their PEB Kind (both read-only values once the PIC inside is burned). The two documented PEB Kinds are *supervisor* and *voter*, which are visually differentiated by a red and blue band in the casing. In Ohio, only supervisor PEBs are used in the documented election procedures. There is a third (undocumented) PEB Kind recognized in the iVotronic source code.

The first block of a Supervisor PEB further identifies it as having been configured as a regular Supervisor PEB or as a special-purpose Supervisor PEB which can perform a single iVotronic administrative function without needing to go through multiple menus. These special-purpose PEBs are commonly referred to by the name of the function they access (e.g. a "Clear and Test PEB" jumps immediately to the Clear and Test menu of the iVotronic it is inserted in).

In a regular Supervisor PEB, the remaining blocks hold the ballot (identical to the `.bin` or `.ebn` produced in HPM), followed by a fixed size "vote results structure" (the `.vot` file read by ERM), and finally the remainder of the PEB (except for the last block) holds write-in ballots. The last block of the PEB is known as the Election Qualification Code block, and serves to authenticate the PEB to the iVotronic.

5.2.5 PEB Reader

The PEB Reader is a standalone cradle used by devices with standard RS232 serial ports to communicate with PEBs. It acts as a media converter between the RS232 serial connection and the IrDA connection to the PEB. The PEB Reader does not contain any functionality in hardware other than this media conversion; all protocol implementation for communicating with a PEB must be done in software on the device connected

to the PEB Reader. Despite its name, there is nothing preventing a PEB Reader from being used to issue commands to write to or erase a PEB if controlled by suitable software.

5.2.6 Compact Flash Cards

Standard Type I Compact Flash cards are used by Unity and the iVotronic to hold files too large to fit in the PEB flash storage as well as audio ballots and audit and results data. Cards contain a standard FAT16 Windows filesystem and are accessed through a dedicated slot in the iVotronic and an external USB reader on the Unity PC. In Windows, these are mounted to the desktop and accessible to any Windows application with no special libraries.

The ballot data is accessed by the iVotronic on demand, but the presence of the CF card is checked periodically and the iVotronic will not boot without its presence. That same CF card must be present to close the polls. An audit log can be saved to the card when the polls are closed. It is a raw dump of the internal flash memory, compressed and encrypted before saving.

5.2.7 Communication Pack

The Communication Pack is a briefcase containing a thermal printer, a modem (not used in Ohio), space to store PEBs for transport, and a serial cable to connect to an iVotronic. A switch is used to toggle between off, printer, and modem modes, and the case contains batteries for when a wall outlet is unavailable.

The communication pack is used by disconnecting the RTAL (if present) from an iVotronic (supervisor or voter) and connecting the communication pack's serial cable when prompted by the iVotronic. The iVotronic then controls the printer or modem in the communication pack and prompts the user when to disconnect the communications pack and reconnect the RTAL.

5.2.8 Model 100

The Model 100 (M100) is a voter-operated optical scan ballot counter intended for use at the polling location. It is mounted on top of a secure ballot box which holds the accepted (and counted) ballots and provides physical security for the M100. The M100 we were provided with used one set of identically keyed locks for physical protection of the M100 and ballots, and a second differently keyed lock for selecting the mode of the M100.

The M100 contains an Intel 286 microprocessor with 2MB of RAM, and runs the QNX embedded operating system from an internal 512KB flash memory. A thermal receipt printer and 40x4 character LCD text display are built-in, and a standard parallel printer port is available for connecting an external printer (when a parallel port connection is detected the M100 automatically switches all printed output to use it instead of the built-in thermal printer). In addition to the optical scanner, the inputs to the M100 are 4 programmable buttons positioned below the LCD display for user input, an RS232 serial port, and two PCMCIA Type slots for loading ballot images and firmware updates, and storing counted ballots.

The M100 performs a self test of its operating system on power-up, and checks for a properly formatted flash card in the PCMCIA slot. The PCMCIA card is mapped directly into memory, and portions of it are copied into RAM for faster access. For the remainder of the data on the PCMCIA card, the M100 builds an index in memory to data structures on the card. The card is written to after each ballot is scanned, recording either the votes cast for a valid ballot or the error for an invalid ballot.

There are two modes for the M100, selected using a key: Open/Close Polls and Vote. When set to Open/Close, a number of administrative and diagnostic options are available which print summaries of the contents of the PCMCIA card. Once done, a poll worker selects the "open polls" or "reopen polls" menu option, and turns the key to Vote when prompted. In Vote mode, the M100 operates in an automated fashion,

only needing user interaction on invalid ballots. At any point the M100 can be switched back to Open/Close mode using the key.

5.2.9 PCMCIA Memory Cards

Unity (via the HPM) and the M100 use specially formatted PCMCIA SRAM flash storage cards for all communications. The cards are formatted so that a small header can be loaded into the M100's RAM which contains pointers into the SRAM for ballot definitions and results counters.

The PCMCIA memory card is also used to upgrade the firmware of a M100. When a correctly formatted card is present at power-up, the M100 will prompt the user to copy a new firmware image to the internal flash memory.

Unlike the Compact Flash cards and Zip disks used by other equipment, the PCMCIA card does not contain a recognizable filesystem and cannot be accessed directly through Windows. Instead, a library for the OmniDrive USB reader is required to read or write data.

5.2.10 Model 650

The Model 650 (M650) is a centralized high-speed optical ballot counter intended for use at a central elections office. It scans batches of ballots, possibly from multiple precincts, and tabulates results to be transferred to Unity.

The M650 contains an Intel Pentium processor and runs the QNX real-time operating system off of an internal solid-state hard drive. There are two standard parallel printer ports which must be connected to printers for operation to begin. Additionally, an Iomega Zip 100 drive is used to transfer ballot definitions and perform firmware updates to the M650, and carry results from the M650 to Unity. The M650 has a control panel on its front with buttons to start and stop scanning as well as set the mode of operation. An LED display provides feedback to the operator.

Inside the chassis, the motherboard and daughter cards are accessible behind a locked panel. There are two RS232 serial ports and a PS2 keyboard/mouse port located on daughter cards which are not accessible from outside the locked chassis.

5.2.11 Zip Disks

The M650 uses FAT32 formatted 100MB Zip disks to load ballot configurations and store tallies of counted ballots. The files on the disk are copied by the Hardware Programming Manager. In Windows, these disks are mounted to the desktop and accessible to any Windows application with no special libraries.

5.2.12 AutoMARK Voter Assist Terminal

The AutoMARK Voter Assist Terminal (VAT) is a combination scanner/printer used by the voter to mark and verify optical scan ballots (to be tabulated by the M100 or M650). It is intended to be used by disabled voters as required by the Help America Vote Act. The VAT runs Windows CE (now called Windows Mobile) with a custom application developed in .Net for the interface.

The VAT is operated by inserting a paper ballot which is then scanned and matched to a pre-defined ballot style configured during election setup. Translations into various languages as well as audio ballots may be configured for a particular ballot style which the user then uses to complete or verify their ballot. Once the user confirms their choices, an inkjet print head contained within the VAT marks the inserted ballot (if it was initially blank) and returns it to the voter. There is a single Compact Flash slot used to program ballot styles, and several jacks for audio headsets and input devices. The AutoMARK does not store or tabulate votes itself, and no data is read from the AutoMARK into Unity.

5.3 Election Procedures

This section describes how the ES&S components discussed in this overview are used in an actual election. We draw our information from the ES&S documentation, and assume that elections follow these general procedures, though some counties may differ on specific procedures.

5.3.1 Preparation

Preparation for an election begins by setting up complete county and election databases using the EDM. This is an involved and complicated process requiring an expert user. The EDM stores all county, precinct, jurisdiction, ballot styles, contests, registered voter totals, and whatever other information is necessary for the election. The set up of this database is so complicated that many counties rely on a service from ES&S to create the database for them. In addition to the election database, ballots must be designed using either ESSIM for paper ballots or the iVIM for iVotronic electronic ballots. Per-election passwords for iVotronic DREs are also set at this point.

Once the election database has been created, and ballot images prepared, an election administrator uses the HPM to program all the media required by the different ES&S voting hardware.

5.3.2 Touchscreen (DRE) Voting

The election process for DRE voting requires the HPM, a Supervisor iVotronic terminal, Compact Flash cards, and PEBs.

Preparation

At each new election an Election Qualification Trail must be started. The Supervisor terminal displays the Election Qualification Code (EQC) and then permits the qualification of PEBs to be used in the DREs. Each qualified PEB is programmed with the EQC and new system passwords (if changed) are ready to be loaded with the ballot for the current election. A PEB is programmed using a supervisor terminal which is connected through a serial port to a machine running the HPM. One PEB for each precinct is chosen as the master PEB, the others are referred to as supervisor PEBs.

A Compact Flash card (one for each iVotronic used in the election) must be programmed. CF cards are programmed through the HPM using a standard commodity CF card reader/writer.

At some point, after the PEBs and CF cards have been programmed, the blue voter terminals must be cleared and tested using a correctly programmed supervisor PEB. This erases the votes and audit data from the previous election, and loads the EQC and any custom passwords. The Election Test menu options are only available after a terminal has been cleared and tested.

The iVotronic allows the following election tests:

- Logic and Accuracy - Tests whether votes are recorded accurately.
- Automated Multi Vote Test - tests multivote and write-in option.
- Vote Selected Ballot Test - for multiple ballot images
- Manual Vote Test

Election Day

On election day poll workers unpack and set up the blue iVotronic terminals, plug in the RTAL printer and power cables. A properly programmed CF card must already be installed in the terminal, (this is usually done at Election Central and a tamper-evident security seal usually placed over the CF slot), before powering the terminal on. A master PEB is required to open each terminal for voting, and only that same master PEB can be used to close the terminal after the polls have closed. At this time a zero tape showing no votes cast may be printed, but this is optional.

For each voter, a supervisor PEB containing the ballot images must be inserted into the iVotronic. Insertion of the PEB turns on the iVotronic, checks the EQC, and initializes (loads in) the ballot. The poll worker removes the supervisor PEB, the voter votes, the RTAL printer prints the results and the electronic ballot is stored internally in the iVotronic until the terminal is closed.

Vote Tallying

To close a terminal, the master PEB is inserted, which collects and stores the tabulated data, copies of the "images" of the ballots cast and time and date information. At closing the iVotronic firmware automatically uploads Audit Data onto the CF card. At this time a results tape can be printed using a special external printer.

The results tape, zero tapes, Compact Flash cards, and Master PEB are then returned to Election Central. At Election Central the results can be imported into ERM using either the Master PEBs or the Audit Image on the CF cards.

5.3.3 Precinct-Based Optical Scan Voting

The M100 is an optical scan machine used to process individual paper ballots. It can be operated by the voter or by a poll worker. The election process for an M100 machine requires the HPM, a commodity PCMCIA reader/writer and two physical keys, the scanner key and the ballot box key.

Preparation

The M100 reads the proper election definition from a prepared PCMCIA card. This card is programmed at Election Central using the HPM and a locally connected PCMCIA reader/writer. The PCMCIA card can then be inserted into the M100 and the slot secured with a tamper-evident seal.

Election Day

A poll worker unpacks and sets up the M100. Usual procedures include using the ballot box key to open the emergency bin and compartment side doors to verify that the bins are empty. The compartments are then re-locked and the key secured. The M100 requires a key (scanner key) to power on. A poll worker inserts this key into the power on switch, and turns the switch to the Open/Close Polls position. The scanner loads the election definition from the PCMCIA card into its operating system. The scanner will display "S-MODE" in the upper left corner of the LCD screen and the message "ELECTION CARD INSERTED OPEN POLLS NOW?"

The poll worker then turns the scanner key to the VOTE position. After initializing, the scanner automatically prints an Initial State Report plus any other reports it was programmed to print. This may include a report showing no votes on the scanner for each of the races and/or questions as well as a certification message. The poll worker then secures the scanner key.

After a voter has marked their ballot, the ballot is scanned by inserting it into the ballot entry slot in any direction. The ballot count on the display increases whenever the scanner successfully scans a ballot. If there are no issues with the ballot such as over or under voting the operator can press accept to have the ballot deposited into the ballot box. If there are concerns with the ballot (such as over or under voting) there is an option to press reject and retrieve the ballot. The ballot will then be spoiled and the voter is usually issued a new ballot and directed to a voting booth.

Vote Tallying

The M100 keeps a running tally of votes internally and on the PCMCIA card.

To close the polls the scanner key is inserted and turned to the OPEN/CLOSE POLL position. The scanner will automatically print reports that may include a Status report, Poll or Precinct report, Certification report, and/or an Audit Log report. The POLLS CLOSED menu will appear after printing and results is complete.

The PCMCIA card is removed, the ballot boxes removed from the cabinet and the boxes, cards and the printed reports are returned to Election Central.

5.3.4 Centrally Counted Optical Scan Voting

The M650 is an optical scan machine for processing batches of ballots. It is usually kept at Election Central. The election process for an M650 requires the HPM and a commodity Zip Disk reader/writer.

Preparation

The M650 reads the proper election definition and firmware from a Zip disk. The Zip disk with the election definition is placed in the Zip drive in the M650 and the scanner is powered on. This message will appear once the scanner has initialized: “Press Stop to Keep XXXXX Press Start to Load YYYYY (Zip)”, where XXXXX is the name of the election currently in the machine, and YYYYY is the name of the election on the Zip disk.

If start is pressed, the M650 displays a menu option for zeroing out previous election totals (this is optional) and prints out a readiness report. The machine is now ready for ballot tabulation.

Election Day Vote Tallying

Ballots are transported to Election Central from the various precincts in secured transfer cases. On arrival the cases are opened and an election worker processes the ballots into the M650. To tally ballots an election worker places them in batches into the M650 input hopper and presses start. Counted ballots are stored in an output hopper and must be removed by hand before the next batch of ballots is tabulated. The result totals are stored internally and the election worker needs to periodically press the Save button on the M650 to store the results to the Zip disk for transfer to the Unity ERM. The manual recommends saving and transferring the results on the Zip disk to the machine running the ERM every time the hopper is emptied. It is possible to network the M650 and to have the results transferred by network instead of by Zip disk.

5.4 Software Versions

The source code analysis and red teams were provided with the following versions of the Unity environment and source code by the vendor:

Component	Application Version
Unity	3.0.1.1
Audit Manager	7.3.0.0
Election Definition Manager	7.4.4.0
Election Reporting Manager	7.1.2.1
Hardware Program Manager	5.2.4.0
Data Acquisition Manager	6.0.0.0
ESS Image Manager	7.4.2.0
iVotronic Image Manager	2.0.1.0
iVotronic Firmware	9.1.6.2 9.1.6.4
M100 Firmware	5.2.1.0
M650 Firmware	2.1.0.0
RMCOBOL RT	7.5.01
COBOL WOW RT	3.12

Source Component	Version		Source Component	Version		Source Component	Version
4 Key Sound Pic	1.0.0.0		Events	9.1.1.0		REGUTIL	1.0.0.0
AuditManager	7.3.0.0		EXITWIN	1.0.0.0		SCNDAT30	3.0.1.0
BXP_Gen	1.0.0.0		GENPARMS	1.3.0.0		Serve650	1.0.0.0
CB_650	1.0.0.0		GetAuditData	9.1.0.0		SHELL	1.0.0.0
CB_EMP	1.0.1.0		Images	9.1.2.0		SHELLSETUP	1.0.0.0
CB_M100	1.2.0.0		Init650	2.1.0.0		TGEN30	3.0.1.0
CB_Duplicator	9.1.0.0		iVOTIM	2.0.1.0		Undrvote	9.1.2.0
CF_Utility	9.1.0.0		iVotronic	9.1.6.2 9.1.6.4		Viodialog	9.1.2.0
CRCDLL	1.4.0.0		Loader ILD	9.0.0.0		VIOWIN	9.0.0.0
EDM	7.4.4.0		M100	5.2.1.0		Volume Control	1.0.0.0
ERM	7.1.2.1		M650 Display	2.1.0.0		WDAM	6.0.0.0
ERSGEN30	4.1.0.0		M650 Firmware	2.1.0.0		WHPM	5.2.4.0
ESSCRYPT1	1.0.0.0		M650 Support Scripts	2.1.0.0			
ESSCRYPT	1.8.0.0		MAKEIBIN	9.1.0.0			
ESSDECPT	1.8.0.0		MPRBOOT	2.6.0.0			
ESSIM	7.4.2.0		MYDLL	1.0.0.0			
ESSM100	1.3.1.0		PCCARD30	3.0.0.0			
ESSPCMIO	1.0.0.0		PEB PIC	1.7c			
ESSUTIL	1.0.0.0		PGENAZ80	1.2.0.0			

5.5 Unity Acronyms and File Extensions

`.ais`	Ballot layout file extension for ESSIM
BDF	Ballot Data File: EDM database file
`.bdf`	BDF file extension
BIM	Ballot Image Manager: see ESSIM
`.bin`	Unencrypted iVotronic Ballot file extension; created by HPM helper application; used by HPM PEB writer helper application
BSC	Ballot Set Collection: created by EDM; used by ESSIM
`.bsc`	BSC file extension
`.ebn`	Encrypted iVotronic Ballot file extension; created by by HPM helper application; used by HPM PEB writer helper application
EDM	Election Data Manager: Unity elections database program
ERM	Election Reporting Manager: Unity tabulation and results program; alternatively called Election Report Manager and Election Results Manager in source code and documentation
ESSIM	ES&S Image Manager (also known as BIM): Unity optical scan ballot page layout program
`.ifc`	IFF file extension for HPM
IFF	Intermediate Interface File: ballot data file; created by EDM; used by ESSIM and HPM
`.iff`	IFF file extension for ESSIM
iVIM	iVotronic Image Manager: Unity touchscreen DRE ballot layout program
PEB	Personalized Electronic Ballot: Hardware token used to program and vote on iVotronic DRM; alternatively called Personal Electronic Ballot and Portable Electronic Ballot in source code and documentation
`.pxt` and `.px1`	Votronic Election Definition file extension; created by iVIM; used by HPM for PEB
RTAL	Real-Time Audit Log (see VVPAT); Also used to refer to iVotronic with RTAL attached
`.vot`	iVotronic Vote Results file extension; created by ERM PEB reader helper application; used by ERM
VVPAT	Voter Verified Paper Audit Trail: Printed record of votes cast on a DRE
`.xml`	Ballot Foundry Election Definition file extension; created by EDM; used by iVIM

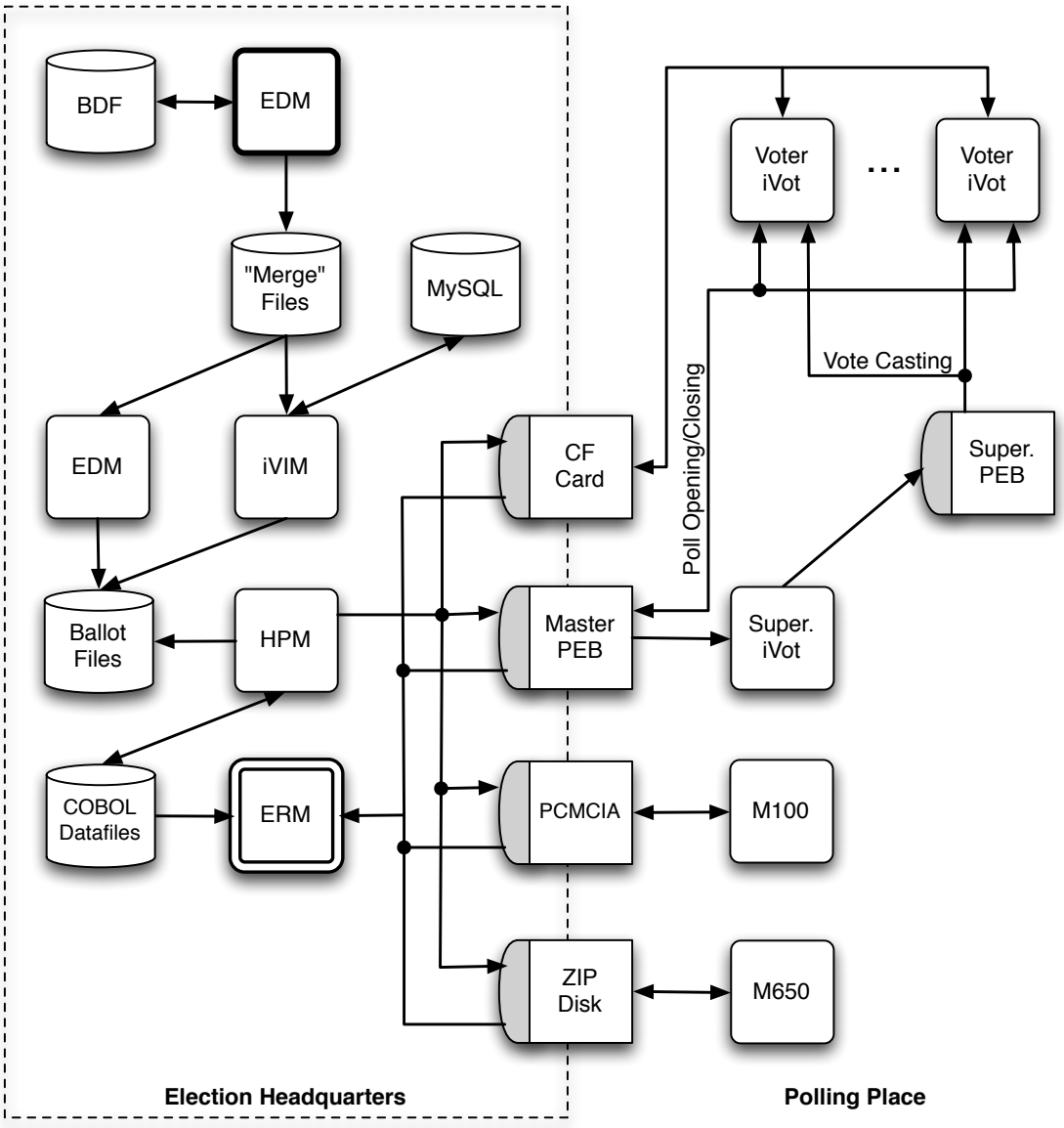

Figure 5.2: The major components and the flow of election data for counties using the ES&S Unity voting system. Arrows indicate direction of data flow. An election begins with configuration using the Election Data Manager (EDM), and ends once results are tallied and reported in the Election Reporting Manager (ERM).

Figure 5.3: A blue Voter iVotronic with a Supervisor PEB inserted

Figure 5.4: A red Supervisor PEB

Figure 5.5: A Compact Flash card used by the Voter iVotronic and AutoMARK

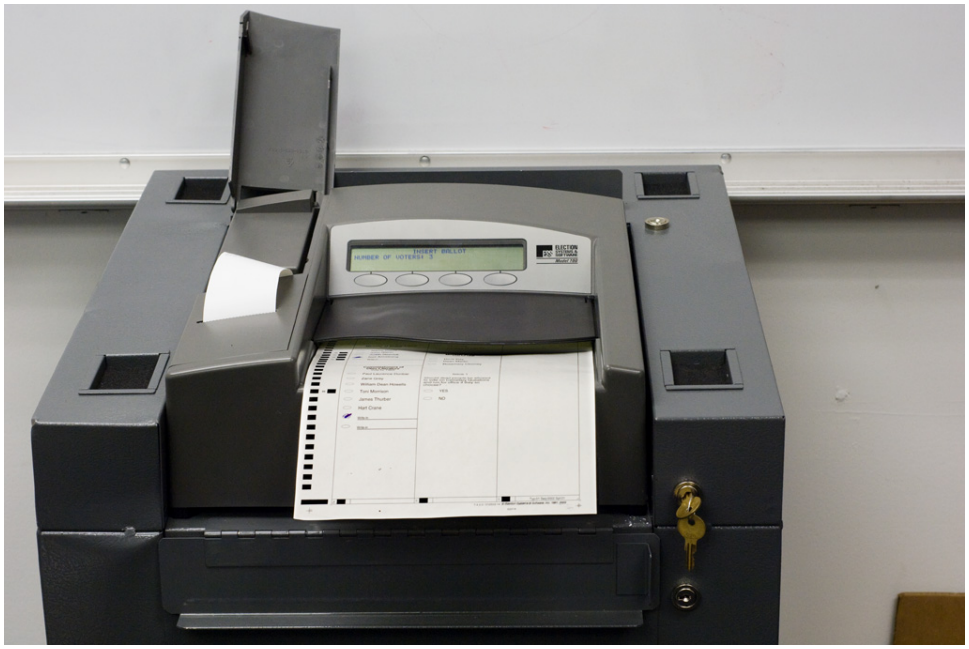

Figure 5.6: The ES&S M100 precinct optical scan ballot counter

Figure 5.7: A PCMCIA SRAM flash card

Figure 5.8: The ES&S M650 centralized optical scan ballot counter

CHAPTER 6

ES&S SYSTEMIC AND ARCHITECTURAL ISSUES

We found fundamental security deficiencies throughout the ES&S Unity EMS, iVotronic DRE and M100 optical scanner software and hardware. Virtually every mechanism for assuring the integrity of precinct results and for protecting the back-end tallying system can be circumvented. Election results can be tampered with in the ES&S system by exploiting any of a number of different vulnerabilities we discovered. The normal access provided to individual precinct poll workers (and in some cases to voters themselves) is sufficient to conduct attacks that alter county-wide election results and that, in some cases, cannot be detected or recovered from through audits or recounts.

Perhaps most seriously, there is a strong potential for practical attacks that propagate "virally" from the field back to the county election management system. That is, a single circumvented piece of precinct hardware (such as a memory card returned from a precinct for vote tallying) can effectively "take over" the county-wide back-end tally system, alter county-wide results reported in the current election, and then corrupt the installed firmware of additional precinct hardware in subsequent elections. The broad scope of such attacks provides great leverage to the adversary and can be extraordinarily difficult to detect, trace, or recover from.

Both the DRE (iVotronic) and the precinct-based optical scan (M100) systems are subject to many exploitable vulnerabilities. The DRE system provides more vectors for attacks that cannot be recovered from through manual recounts, however.

While there are many specific errors and weaknesses in various parts of the ES&S software (and which are detailed in Chapter 7), there are also systemic weaknesses throughout the system's overall design and implementation. These weaknesses render the system as a whole especially difficult to secure in practice. We identify here four fundamental, pervasive deficiencies that give rise to the most serious vulnerabilities we found:

- Ineffective access control
- Critical errors in input processing
- Ineffective protection of firmware and software
- Ineffective cryptography and data authentication

6.1 Ineffective Access Control

The firmware and configuration of the ES&S precinct hardware can be easily tampered with in the field. Virtually every piece of critical data at a precinct – including precinct vote tallies, equipment configuration and

equipment firmware – can be compromised through exposed interfaces, without knowledge of passwords and without the use of any specialized proprietary hardware.

6.1.1 iVotronic passwords and PEB-based access controls

Access to the iVotronic DRE configuration is protected by several hardware and password mechanisms, all of which can be defeated through apparently routine poll worker (and in some cases voter) access.

The primary mechanisms for preparing iVotronic DREs for deployment at precincts and for managing them throughout the election day (e.g., enabling them for each voter) employ the *Personalized Electronic Ballot (PEB)* interface. As discussed in Chapter 5, a PEB is a small module that communicates with the iVotronic via a magnetically switched infrared (IrDA) bidirectional data interface on the face (voter side) of the terminal. PEBs are used for several different kinds of functions. Some of these functions are intended to be performed at the county headquarters (e.g., loading ballot definitions and basic configurations), while others are performed by poll workers (e.g., opening the terminals at the beginning of the day, enabling a voter to use a particular ballot, closing the terminal and collecting vote totals). In the mode used in Ohio, the PEB slot is empty whenever a voter is voting.

PEBs are used as external memory devices that communicate through a simple protocol that allows the iVotronic to read and write memory blocks stored in the PEB. Access to PEB memory is not protected by encryption or passwords, although some of the data stored on them is encrypted (see Section 6.4). PEBs themselves are proprietary devices (and are apparently not commercially available except through ES&S). However, they employ a widely-used infrared communication standard (called *IrDA).*

In spite of the proprietary nature of the "official" PEB, we found it to be relatively simple to emulate a PEB to an iVotronic or to read or alter the contents of a PEB using only inexpensive and commercially available IrDA-based computing devices (such as Palm Pilot PDAs and various mobile telephones).

Most of the administrative and poll worker functions of the iVotronic (e.g., pre-election ballot loading, enabling voting, etc) require the insertion of a properly configured "supervisor" PEB and, in some cases, the entry of a password on the terminal touchscreen. However, we found it to be possible to defeat both of these security mechanisms This makes practical several possible attacks at polling stations.

Unauthorized screen calibration and configuration

One of the simplest, and yet most important, configuration parameters of the iVotronic DRE is the calibration of its touchscreen input sensors. Calibration (which can be performed in the field through the screen itself) affects how voters' tactile input "maps" to different locations on the screen. If the procedure is performed incorrectly (or has been deliberately altered), voter choices might not be correctly recorded.

It is easy to surreptitiously re-calibrate the screen of an iVotronic terminal in a way that allows most input to behave normally but that denies access to specific screen regions (e.g., those corresponding to certain candidate selections).

Access to the screen calibration function of the iVotronic terminal requires the use of a supervisor PEB during power-up (e.g., after voting or at idle times). No password is required. Any supervisor PEB is sufficient for this purpose, even one not specifically configured for the correct precinct or obtained from some other jurisdiction (e.g., through secondary markets such as eBay). A home-made PEB emulator (e.g., a specially programmed Palm Pilot and a small magnet) is also sufficient. The procedure requires about one minute and is, from a distance, largely indistinguishable from normal voter behavior.

A terminal can be maliciously re-calibrated (by a voter or poll worker) to prevent voting for certain candidates or to cause voter input for one candidate to be recorded for another. The terminal will remain in this state until the problem is detected (e.g., through voter complaints) and the terminal correctly re-calibrated by poll workers (which may require consultation with the central county office). Voters may or

may not recognize that their votes are not correctly recorded, depending on voter training and other factors.

While a maliciously calibrated terminal may be noticed by voters and can, in principle, be corrected in the field, the attack is extremely simple for a poll worker (or other person with access to a PEB) to carry out and practical even without a PEB, and so may represent a serious practical threat. We note that iVotronic behavior consistent with such attacks has been reported in various jurisdictions during actual elections.

Undocumented PEB features can be used to bypass password checks

Many of the more sensitive iVotronic administrative functions (closing the polls, clearing the terminal, etc) require the entry of passwords in addition to the insertion of a supervisor PEB. However, there is a special *Quality Assurance (QA)* PEB type recognized by the iVotronic firmware that behaves essentially as a supervisor PEB but that, when used, does not require the entry of any passwords. This PEB type does not appear to have been described or documented in any of the ES&S manuals or training materials provided to our review.

This undocumented PEB feature can be used to neutralize the security of any iVotronic administration features that depend on passwords, no matter how carefully passwords are managed by a county. Anyone with such a PEB – whether it was supplied by ES&S, stolen, or emulated with a palmtop computer – effectively has a "back door" that bypasses this basic security check. As noted above, a simple Palm Pilot-type device can be programmed to emulate a PEB. QA PEBs are no more difficult to emulate than regular supervisor PEBs; they are similar to supervisor PEBs but with a single character changed in the communication protocol.

Note that while the QA PEB bypasses password checks, there is another iVotronic security feature required for access to some (but not all) administrative functions, For these functions, a PEB must be configured with the correct *Election Qualification Code (EQC)* (a 32 bit random number assigned for each election). However, as noted in the next section, precinct poll workers (and others with brief access to the poll worker equipment) can easily extract this code from the precinct's supervisor PEB using a palmtop computer.

Note that even without the EQC, however, an attacker (who needs no more access than that provided to a normal voter) with a QA PEB (or an emulated QA PEB) can do a great deal of harm to an iVotronic terminal. For example, the EQC is not required on QA PEBs used to invoke the "clear" function on an iVotronic terminal, a which erases all stored votes and renders the terminal useless for the rest of the election day.

Unauthorized PEB copying and alteration

Anyone with physical access to polling station PEBs can easily extract or alter their memory. This requires only a small magnet and a conventional IrDA-based palmtop computer (exactly the same kind of readily-available hardware that can be used to emulate a PEB to an iVotronic terminal). Because PEBs themselves enforce no passwords or access control features, physical contact with a PEB (or sufficient proximity to activate its magnetic switch and IR window) is sufficient to allow reading or writing of its memory.

The ease of reading and altering PEB memory facilitates a number of powerful attacks against a precinct's results and even against county-wide results. An attacker who extracts the correct EQC, cryptographic key, and ballot definition can perform any election function on a corresponding iVotronic terminal, including enabling voting, closing the terminal, loading firmware, and so on. An attacker who has access to a precinct's main PEB when the polls are being closed can alter the precinct's reported vote tallies, and, as noted in Section 6.3, can inject code that takes control over the county-wide back-end system (and that thus affects the results reported for all of a county's precincts).

Individual precinct poll workers have many duties that involve handling PEBs throughout the election day (whenever a voter votes, for example), and so are in a natural position to carry out attacks that involve altering or reading PEB memory without engaging in suspicious activity.

6.1.2 Physical security, locks and seals

Many aspects of the ES&S system's security as a whole depend on the integrity of the interfaces and removable media associated with precinct equipment. Some of these interfaces and media are protected by software security (e.g., access passwords, encryption, etc); potential attacks against such mechanisms are discussed in other sections of this report. Many interfaces and media are also protected (partly or entirely) by physical mechanisms: locks, seals, and procedures.

Although this study did not aim to conduct an exhaustive analysis of the physical security of the ES&S equipment, we found many of the basic physical security features that protect precinct hardware to be ineffective or easily defeated.

iVotronic

Several features of the iVotronic's physical security were especially problematic:

- A primary mechanism for logging events (including those potentially associated with an attack) on the iVotronic terminal is the RTAL printer. However, the cable connecting the printer is readily accessible to the voter and can be removed easily and without tools or overtly suspicious activity. It is possible to for an attacker to suppress logging simply by unplugging the cable. It is also easy for an attacker to print arbitrary messages on the printer (including ballot choices) by connecting a small handheld computer to the printer cable.

- The PEB interface on the iVotronic terminal is exposed and readily accessible to the user during voting. As noted above, this facilitates several important attacks.

Locks and seals

The mechanical locks supplied with all of the ES&S precinct equipment sent to the source code review team were uniformly of very low-security designs that can be easily picked or otherwise bypassed.[1] Many locks use keys that are apparently identical in equipment shipped to different customers, and so would provide little security even if the locks were improved. In almost every case, it was not actually necessary to operate the locks, since the equipment cases could generally be opened by removing a few screws and the locks bypassed altogether.

Other physical security mechanisms depend on tamper-evident seals. As noted in Chapter 9, we were not provided with samples of the seals used in every county, and so cannot conclusively comment on the security of the particular seals used in Ohio. However, we note that all but the most sophisticated commercially-available tamper seals are often surprisingly easily to defeat.[2]

Even if effective at revealing tampering, seals are inherently limited in the protection they provide. As noted in Chapter 3 and in other reports,[3] seals do not *prevent* tampering; at best they can *detect* it. But in an election, even reliable detection of tampering may be unsatisfying, since if a seal is found to be broken once

[1] For the first weeks of the project, we did not have the correct keys for much of the equipment; we frequently had to pick the locks in order to conduct our analysis.

[2] Roger G. Johnston, 'Tamper-indicating seals'. American Scientist, 94 November-Decemeber 2006.

[3] Matt Blaze et al., *Source Code Review of the Sequoia Voting System*. University of California, Berkeley under contract to the California Secretary of State, July 20, 2007 ⟨URL: `http://www.sos.ca.gov/elections/voting_systems/ttbr/sequoia-source-public-jul26.pdf`⟩.

polling has started, it is unclear what should be done. If the compromised equipment is used, fraudulent votes may be counted. If it is not used, previously cast legitimate votes may be lost (making breaking a seal a simple way for an attacker to destroy votes).

6.2 Critical Errors in Input Processing

At least two critical components of the ES&S system suffer from exploitable errors in functions that process input over their external interfaces. Both the Unity tallying system and the iVotronic terminal have *buffer overflow* software bugs that allow an attacker who can provide input (e.g., on a PEB or memory card) to effectively take control over the system. A buffer overflow in input processing is common type of programming error, one that has been responsible for many security failures in modern computing. Avoiding buffer overflows in input processing is regarded as one of the most basic defenses a system must have.

We found numerous buffer overflows throughout the ES&S system. Several of these buffer overflows – in the Unity tallying software and in the iVotronic terminal firmware – have extremely serious practical security implications. An attacker who can present input to any these systems (on an iVotronic PEB or on an M100 memory card from a precinct) can exercise complete control over the results reported by the entire county election system.

Most seriously, the nature of these vulnerabilities means that there are few barriers to obtaining the access required to exploit them. In the case of the iVotronic system, voter access to the terminal is sufficient. In the case of the Unity system, brief access to any iVotronic or M100 optical scan results media returned back to the county for processing is sufficient. As discussed in the Section 6.3, it is also possible to carry out the attacks against the Unity system by tampering with the firmware of precinct equipment.

6.2.1 Unity

The Unity election management system processes all precinct results and produces the tally reports that, in most cases, constitute the official tallies in races. After polls are closed, precinct-counted ballot results are received into Unity through several different media, including iVotronic PEBs, iVotronic CF cards, and M100 PCMCIA memory cards.

While Unity appears to correctly process properly-formatted results from such media, buffer overflows in Unity allow a maliciously altered iVotronic or M100 tally from a precinct to execute arbitrary software on the computer on which Unity runs, to replace or alter the Unity software, and to make arbitrary changes to the tally database and other election records. There may be no indication to the operator that this is occurring, and a system thus corrupted may continue to appear to operate normally when it is actually running software controlled by an attacker.

Because these attacks are carried out entirely through media routinely brought in to the county headquarters from precincts on election night, an attacker need not have any physical access to the secure county facility in which Unity is located. It is entirely sufficient for the attacker to have access to media (such as PEBs or M100 memory cards) returned to the county at the end of the election, or to equipment (such as iVotronics and M100s) that write to such media. Poll workers handle such media in the normal course of their duties, and may have unsupervised access at various times of the day. And as noted in the next section, a voter using an iVotronic DRE may be able to circumvent the iVotronic terminal in a way that causes it to automatically produce such media when the polls close at the end of the day.

Note that because these vulnerabilities affect the central counting system, a corrupted media attack conducted from *any single* precinct can corrupt results for the entire county.

We have successfully implemented PEB-based attacks against Unity (at the University of Pennsylvania and at WebWise) and have confirmed that such attacks represent a readily-exploitable threat in both iVotronic

and M100-based systems.

6.2.2 iVotronic

The iVotronic terminal firmware has several exploitable buffer overflow errors in its PEB input processing functions. These buffer overflows allow a PEB containing carefully-structured data (or an emulated PEB based on a palmtop computer) to take control over the terminal. The implications of attacks against iVotronics are discussed in Section 6.3.

We found it to be straightforward to exploit the iVotronic buffer overflows in several different ways (by emulation of a QA or supervisor PEB with a palmtop computer or by writing data to a precinct's supervisor PEB) at various times (while opening polls and during the polling day), and with various degrees of access (as a poll worker or as a voter). The exposed nature of the PEB port and the many different scenarios under which it can be exploited make attacks against the iVotronic very difficult to effectively guard against under operational election conditions.

6.3 Ineffectively Protected Software and Firmware

The integrity of election results depends heavily on the integrity of the software and firmware that runs the central election management system and the precinct hardware. The consequences of any attack that alters, replaces or otherwise compromises this software or firmware are sweeping and often impossible to recover from. The security features that protect election software and firmware from unauthorized tampering are therefore among the most critically important safeguards in the system as a whole.

We found exploitable vulnerabilities that allow an attacker to replace or alter the firmware and software of virtually every component of the ES&S system, either by circumventing access controls or by triggering software errors.

6.3.1 iVotronic firmware

The iVotronic terminal is based on an Intel 80386 embedded computer processor controlled by firmware stored on an internal flash memory chip. The firmware is designed to be field-updated through an administrative menu function, with new firmware loaded though the terminal's CF card interface. Four security mechanisms are intended to protect against unauthorized firmware loading:

- Access to the firmware update menu function requires a supervisor (or QA) PEB.
- A 6-8 character password is required to enable firmware update.
- The firmware is loaded through the CF card interface, which can be protected by a sealed sliding cover.
- The firmware update function is disabled while the polls are open.

Unfortunately, these mechanisms are ineffective. We found several practical ways for an attacker to bypass each of these security mechanisms and successfully replace or alter the iVotronic firmware, without knowledge of any passwords or secret election parameters, possession of a PEB, or breaking any seals. We found ways to carry out these attacks even when the polls are open. It is possible, for example, for a voter (with no inside assistance) to load new firmware into an iVotronic after he or she is finished voting.

We found at least three different vectors that an attacker could exploit to load unauthorized iVotronic firmware under various circumstances.

Via direct replacement of the internal flash chip: The iVotronic terminal housing can be disassembled easily without breaking the seal that protects the CF slot. Disassembly requires only the use of a readily available Torx security screwdriver. Once the housing has been removed, the internal flash chip can be removed from its socket, reprogrammed with a standard flash writer, and replaced. Note that while surreptitious terminal disassembly is unlikely to be possible in an active polling place, it may be an attractive option for an attacker who enjoys unsupervised access to stored terminals (e.g., the night before an election).

Via the firmware update menu: This is the most direct attack against firmware. As discussed above, a palmtop computer and a magnet can be used to emulate a QA PEB and bypass the password check. If the polls are open, they can be closed by using a an emulated QA PEB to clear the terminal first. Note that with this approach, the firmware must be loaded though the external CF card interface, which might be protected with a tamper-evident seal (although that seal can be bypassed by removing the housing).

Via the PEB interface, during the polling day: This is perhaps the most serious practical threat to the iVotronic firmware. As discussed in Section 6.2, errors in the iVotronic's PEB input processing code allow anyone with access to the PEB slot on the face of the terminal (including a voter) to load malicious software that takes complete control over the iVotronic's processor. Once loaded, this software can alter the terminal firmware, change recorded votes, mis-record future votes, and so on throughout the election day and in future elections.

Any attack that compromises iVotronic firmware is extremely serious; it can be very difficult to detect whether such firmware has been used in a live election or meaningfully recover once it has. The firmware controls every aspect of the ballot presented to voters, the recorded votes, and the interface to the tally system. Because the RTAL printer is under the control of the firmware, compromised firmware can easily print misleading choices that evade the notice of voters or that cancels the printed ballot (replacing it with other choices) after the voter has left. The discovery of compromised firmware at a terminal casts doubt upon every vote cast at that machine (and, because of additional bugs in the Unity back-end, on the integrity of the results reported countywide as well).

Compounding the problem is the fact that there are apparently no tools available to counties in the ES&S system that reliably extract or audit the actual firmware present in any given terminal. The version number is displayed at boot time, but that is not a reliable indication of whether the firmware has been compromised, since the message is part of the firmware itself. Compromised firmware can display any version number that it wishes to impersonate.

The iVotronic firmware code includes a number of internal consistency checks intended to detect corrupted firmware. While these checks may be able to detect accidental memory errors, they are ineffective against maliciously installed firmware, which can simply bypass or omit the integrity check functions.

6.3.2 Unity software

No single component of the ES&S system is more important to the integrity of election results than the central Unity election management system. Unity is a complex software suite, consisting of many components that share a common database. Securing a county's Unity system therefore depends on each of its components and on the computing platforms on which it runs.

Because Unity (at least as used in Ohio) apparently runs only in a single, secure location in each county, with presumably only trusted staff permitted access to the computers, attack vectors involving unauthorized direct physical access by poll workers, voters or others are a less significant threat here than in precinct equipment sent to the field. However, because the Unity system processes electronic data received from

precincts, it is subject to a number of indirect – yet devastating – attacks that can originate with poll workers or voters, even if they cannot themselves physically touch the Unity computers.

Attacks via input from precincts

As discussed in Section 6.2, malicious input carried on iVotronic and M100 results media can take over the Unity system when it is loaded for counting. This enables many of the most serious and comprehensive attacks we discovered.

Windows environment vulnerabilities

The Unity software runs on an off-the-shelf version of the Microsoft Windows operating system. This platform is very heavily dependent on many aspects of the local computing environment for its security. When used in a networked environment, even behind a network firewall, there are many potential vulnerabilities that can be mitigated only through careful, expert system management.

Unfortunately, the precise requirements for using Unity in a networked Windows environment are not specified by ES&S, and appear to be left to individual counties to manage without specific guidance. An analysis of these issues is beyond the scope of this report, but we caution that there are many potential problems that could arise from the various ways in which the Unity system's Windows platforms might be managed, some of them quite subtle.

6.3.3 M100 firmware

Firmware can be loaded into the M100 optical scan ballot counter by placing a specially structured file on its PCMCIA card (which is also used during polling for ballot definitions and other precinct parameters). If new firmware is present on the card when the M100 is turned on, there is a brief screen prompt and the new firmware is loaded. No password is required.

Any poll worker (or other person) with access to the PCMCIA card slot can thus easily load new firmware. This slot may be protected by a tamper seal in some jurisdictions, but the seal may be able to be bypassed because of the design of the cover mechanism.

Because the firmware is loaded from the same PCMCIA cards used to load ballot definitions, corrupt firmware can also be loaded into the M100 by a corrupted Unity system when an election is provisioned.

M100 firmware controls how ballot definitions are interpreted, the counting and recording of votes, the format of data returned to Unity, and the acceptance and rejection of ballots. The consequences of corrupt M100 firmware are serious, especially given the vulnerabilities in Unity results processing. However, since the paper ballots remain available, they can be recounted if an attack might have occurred.

Unfortunately, as with the iVotronic, there is no mechanism for reliably determining or auditing the actual firmware installed in an M100, so attacks on these devices may be difficult to detect or confirm.

6.3.4 Viral propagation

The software or firmware of almost every major component of the ES&S system can be altered or replaced by input from the other components with which it communicates. In particular, note that, by design or software flaw:

- The Unity system software can be modified by election results media originating from iVotronics and M100s (due to Unity buffer overflows)
- The iVotronic firmware can be modified by configuration media originating from the Unity system (due to iVotronic buffer overflows).

- The M100 firmware can be modified by configuration media originating from the Unity system (due to the design of the M100 firmware management functions).

This confluence of vulnerabilities creates a "closed loop" for viral propagation into every part of the ES&S system through the compromise of a single system component.

For example, a voter can compromise an iVotronic terminal though its PEB slot. The iVotronic, then, may be programmed to create results media (at the end of the election day) that, in turn, corrupts the software of the central Unity system. The compromised Unity system, in turn, may be programmed to load corrupted firmware into all M100s and iVotronics in the county when provisioning a subsequent election. At this point, every major component of the system is running compromised code, which originated with a single attacker with only voter access in a single precinct. Needless to say, such an attack represents a grave threat to the integrity of the elections of any jurisdiction to which this happens.

6.4 Ineffective Cryptography and Data Authentication

Much of the critical election data in the ES&S system – ballot definitions, precinct vote tallies, and so on – are communicated between the central county headquarters and precincts through small removable storage media. In iVotronic DRE-based systems, the primary media are PEBs and, in some cases, CF memory cards. In M100-based precinct counted optical scan systems, the primary media are PCMCIA memory cards.

These media share two important characteristics that make them attractive targets for attack: they have no intrinsic security properties of their own and they may pass through many hands on the way to polling places, during the polling day, and back from polling places. That is, it is simple to read or alter data on these media, and many people may have the opportunity to do so during an election. For example, iVotronic PEBs are handled by poll workers all through an election day, with memory that can be read or written with a standard palmtop computer and a small magnet. PCMCIA and CF cards, similarly, can be readily read or altered with standard laptop computers.

The usual approach (indeed, generally the only practical approach) for securing data stored on such media is the use of cryptographic techniques that prevent meaningful access to data without knowledge of the correct key.

Unfortunately, the ES&S system does not employ cryptography at all in the M100-based optical scan system. The iVotronic DRE system does use cryptography, but errors in its implementation render the protection completely ineffective.

The lack of effective cryptographic protection enables a large fraction of the exploitable vulnerabilities discussed in this report.

6.4.1 Unauthenticated M100 data

M100 PCMCIA cards are used to load ballot definitions and firmware into the M100 and to report tallies back to the Unity system. The data for each of these functions are not cryptographically protected; an attacker with access to an M100 PCMCIA card can easily forge or modify these data. A linear cyclic redundancy code (CRC) is included with the PCMCIA data, but an attacker can easily calculate this; CRC codes are not keyed and are not designed to provide security against deliberate data modification.

6.4.2 Ineffective iVotronic cryptography

The iVotronic DRE uses cryptography to protect data stored on the PEB and in the CF card. The *Blowfish* cipher is used.[4] Unfortunately, the manner in which the encrypted data is stored on the PEBs ef-

[4] Blowfish is a popular public-domain cipher algorithm from the 1990's.

fectively neutralizes the cryptographic protection. The PEB contains an EQC, encoded using an unkeyed (non-cryptographic) algorithm. The EQC is used to encrypt the Blowfish key, which is used to the encrypt the rest of the data on the PEB. That is, although much of data on the PEB is encrypted, there is unencrypted information stored along with it that allows an attacker to easily discover the key.

6.4.3 Poor data validation of precinct results in Unity

The obvious attacks enabled by the lack of cryptographic protection of precinct media include alteration or forgery of data, unauthorized loading of firmware, as discussed in the rest of this report.

Additional vulnerabilities are introduced by Unity's poor validation of various reported precinct data. In particular, the precinct results reported on an incoming M100 PCMCIA card are not checked against the precincts for which the card was originally provisioned. This allows anyone with access to a card to add tally results for extra precincts, which will be added to (or supplant, depending on the mode the Unity operator is using) the true precinct results when read into the database.

6.5 Procedural Mitigations

We believe the issues reported in this study represent practical threats to ES&S-based elections as they are conducted in Ohio. It may in some cases be possible to construct procedural safeguards that partially mitigate some of the individual vulnerabilities. However, taken as a whole, the security failures in the ES&S system are of a magnitude and depth that, absent a substantial re-engineering of the software itself, renders procedural changes alone unlikely to meaningfully improve security.

Nevertheless, we attempted to identify practical procedural safeguards that might substantially increase the security of the ES&S system in practice. We regret that we ultimately failed to find any such procedures that we could recommend with any degree of confidence.

A particular challenge in securing systems that use the iVotronic DRE terminal is the large number of precinct-based attack vectors whose exploitation must be prevented. Effective procedures that accomplish this, even if they existed, would be arduous indeed, and would likely substantially hamper poll workers in their duties, reduce the ability to serve voters efficiently, and greatly increase the logistical challenges of running an election.

It may be possible to deploy a reduced subset of the ES&S hardware and software that excludes components that present the greatest risks. For example, a system that uses only centrally-counted optical scan hardware eliminates many of the threats of precinct-based attacks. We defer to the expertise of the Ohio election officials to determine whether it is possible to use a version of the ES&S system with reduced functionality in a way that presents an acceptable level of risk to the integrity of their elections.

CHAPTER 7

ES&S SPECIFIC WEAKNESSES AND THEIR IMPLICATIONS

In this chapter, we describe specific weaknesses found in the reviewed ES&S systems. For each discovered vulnerability, we detail the prerequisites necessary to compromise security and/or reliability as well as the impact of such attacks. Since we made no attempt to subject the various ES&S software and hardware systems to equal scrutiny, the number of findings among the perspective ES&S components should not be used to infer any conclusions regarding their relative security. We did, however, concentrate our efforts on areas we deemed most apropos to security and reliability.

It is worth emphasizing that we do not assert that the vulnerabilities described in this chapter constitute a complete and exhaustive listing of the security weaknesses exhibited by the reviewed ES&S systems. The reviewed components consist of nearly 670,000 lines of code, encompassing twelve programming languages and five hardware platforms. Given the extraordinary scope and complexity of the ES&S voting system, it is infeasible that any study could comprehensively analyze all hardware components and software modules in a reasonable amount of time. Consequently, it would be improper to conclude that a particular system can be "fixed" merely by repairing the vulnerabilities listed in this chapter, particularly given the overarching engineering issues identified in Chapter 8.

Throughout this chapter, vulnerability details containing proprietary information are provided in a separate confidential annex. References to such proprietary details are denoted using footnotes.

7.1 Unity

7.1.1 Unity ERM buffer overflow when reading a Master PEB

Description: The Election Reporting Manager (ERM) component of Unity is used to compile results from the precincts. The most common method of delivering the results is on a Master PEB. There is a stack-based buffer overflow in the ERM which can be exploited when election, pre-election, or testing results are processed. To exploit this vulnerability, an attacker can create a specially-crafted PEB that will allow arbitrary code execution. As a result, an attacker has the ability to gain full control of the machine running the Unity server.

The EQC on the results PEBs is not checked, so an attacker does not need to know a valid EQC to create a malicious PEB.

Prerequisites: This attack requires access to a PEB and the facilities to write data to the PEB, such as our pebserial tool. The PEB must be read using the PEB reader.

Impact: Since the attack enables arbitrary code execution on the Unity server, an attacker has the ability to gain full control of the machine running the server. Therefore, the attacker could perform activities such as installing a virus or trojan that could then be used to change the election results and spread the virus to other ES&S components (e.g., the iVotronic or M100 machines). Scenario unity.2 in Section 9.3.2 demonstrates an attack using this vulnerability.

7.1.2 Data from M100 can cause a buffer overflow in Unity

Description: Data from the M100 optical scanner can cause a buffer overflow in Unity. The data, which includes election results and audit data, are stored and transferred on a PCMCIA card.

The ERM module of Unity expects a fixed maximum number of precincts to be reported on a given PCMCIA card. This number is hardcoded in the source code of the program. However, the actual number of precincts is read from the PCMCIA card, and the attacker can thus set it arbitrarily high (within limits of the integer datatype used).[1] To exploit this vulnerability, an attacker can create a specially prepared PCMCIA card that will allow the execution of arbitrary code. The attacker can gain full control of the Unity server in this manner.

Prerequisites: If an attacker is to exploit this vulnerability, he or she must have either a) compromised the M100 optical scanner or b) have access to the PCMCIA card that is used to transfer the data to the Unity back-end system.

If the attacker wants to exploit this vulnerability using an exploit of an M100 optical scanner, he or she must gain enough control over the M100 to be able to make it output arbitrary data. This can be accomplished by loading modified firmware to the M100. For possibilities of loading new firmware to the M100, see Section 7.3.1.

If the attacker has unmonitored access to the PCMCIA card when it has election results, he or she can modify the data on this card, since the integrity of the data that M100 saves on the PCMCIA card is not sufficiently protected (see Section 7.1.4).

A poll worker can carry out this attack in either of the two ways described above.

We have not fully implemented the attack, thus we have not experimentally confirmed that the vulnerability is exploitable. We have performed extensive code analysis, as well as initial experiments towards performing the attack, and everything we found indicates that this vulnerability is exploitable.

Impact: This attack allows the attacker to run arbitrary code on the Unity server. Therefore an attacker has the ability to gain full control of the machine running the server and could alter election results or compromise the Unity software. He or she can then use the compromised software to spread the attack virally to other election equipment (iVotronic, M100, M650).

7.1.3 Unity accepts memory media with unauthorized precinct results

Description: Unity (via the Hardware Programming Manager module) passes election definition and other data to an M100 optical scanner via a PCMCIA card. This card contains information identifying those precincts for which the M100 will accept ballots. The election results are stored on the PCMCIA card and are transferred back to Unity on the same card for processing.

[1] See Section 22.7.1.2 in the confidential Annex.

However, Unity does not check whether only the results for which the card, and thus the M100 scanner, were configured are actually returned. A poll worker can thus take the card with election results and insert results for a precinct for which the card was not configured.

A malicious poll worker can therefore not only modify the results for his or her own precinct, he or she can influence the results for other precincts as well. We have verified experimentally that this vulnerability is exploitable.

When a PCMCIA card with results is entered into Unity, Unity indicates how many precincts have been reported on the card. Thus an attentive operator using Unity can catch this attack, in case he or she has a list of cards (their serial numbers) and the number of precincts that should be reported for each card.

This attack raises a question whether a similar one can be carried via PEBs that contain results from iVotronic. This appears to be possible, because, firstly, it is possible to report results for multiple precincts on a PEB and secondly, analysis of the source code of Unity indicates that Unity does not check whether the returned PEB contains only results for precincts for which it was configured. The attack via a PEB (as opposed to one via a PCMCIA card) was not experimentally verified.

Prerequisites: To exploit this vulnerability, an attacker must have either compromised the M100 optical scanner or have access to the PCMCIA card that is used to transfer the data to the Unity back-end system. For more details on prerequisites, see Section 7.1.2. The attacker can also return an independently prepared PCMCIA card for results processing.

Impact: An attacker (such as a malicious poll worker) can influence the results of an election by reporting fictitious results for different precincts. The attacker will not only have complete control over the results being reported from the M100 from his or her precinct, but can also report extra results for other precincts.

7.1.4 Integrity of data on a PCMCIA card is not protected

Description: A PCMCIA card is used to transfer data from the M100 optical scanner to an iVotronic. The integrity of the data on the card is protected only by CRC checks. The protection is not cryptographically strong, and CRC checks were not designed to protect against deliberate data modification. This protection thus does not prevent an attacker to change the data on the card in such a way that the data pass the CRC checks. An attacker can therefore easily modify, delete or add data to the correct results that are stored on the card. This greatly facilitates the attacks described in Sections 7.1.2 and 7.1.3.

Prerequisites: An attacker needs to have access to the card when the data (election results and audit data) are being transferred to Unity.

Impact: If the attacker has access to the card when it is transferred to Unity, he or she can arbitrarily modify the data on the card. Thus the attacker will have complete control over the results coming from the precinct(s) on the card. He or she can also attempt to overtake the Unity back-end system (see Section 7.1.2) or change the vote totals for other precincts (see Section 7.1.3).

7.1.5 Processing audit data can cause a buffer overflow of a global variable

Description: Audit data uploaded to Unity can cause a buffer overflow of a global variable. The ERM module uses a buffer with a fixed amount of allocated memory to input a string with a variable length.[2] The

[2] See Section 22.7.1.5 in the confidential Annex.

attacker can prepare the input to be longer than the pre-allocated buffer. The audit data enter the system via a PEB or a CF card prepared by the iVotronic.

The buffer in question is a global variable (thus is not allocated on the stack) and the number of bytes by which the attacker can overflow the buffer is limited. Therefore, the attacker might not be able to use this vulnerability to execute arbitrary code on the server running Unity.

Prerequisites: If an attacker is to exploit this vulnerability, he or she must have either compromised the device that put the data to the PEB or the CF card (the iVotronic) or have access to the PEB or the CF card that is used to transfer the data to the Unity back-end system.

Impact: The attacker can corrupt the memory of the ERM module, i.e. change the value of some data or influence the control flow. In the event that the memory layout is favorable to the attacker, he or she might be able to realize more severe attacks. As explained above, this attack involves a global variable, therefore the attacker is not guaranteed to be able to overtake the machine running Unity.

7.1.6 Unity decrypts a PEB using the EQC from the PEB

Description: Unity uses the Election Qualification Code stored on the PEB to decrypt the encryption key stored on the PEB.

If Unity used the original EQC that was used for qualifying the other election equipment and compared it with the EQC on the incoming PEB, it would ensure that the data is coming from a properly qualified iVotronic. However, this is not the case.

Unity reads the EQC from a PEB, then verifies the EQC and key data with a CRC, and finally uses the EQC value to decrypt the encryption key, which is subsequently used to decrypt the data on the PEB. The only protection is thus the CRC check, and the attacker can choose data to be such that the CRC check passes.[3]

Prerequisites: The attacker must be able to obtain a PEB - possibly a new one or one used in previous elections and must be able to return it with the PEBs that are to be processed by Unity.

Impact: An attacker can take advantage of this vulnerability to perform the buffer overflow attack on Unity (see Section 7.1.1) or to return incorrect results for the precinct(s) for which the PEB was configured. Note that the attacker does not need access to a PEB used in an election that is in progress and does not need to know the correct EQC.

7.1.7 Unity contains large pieces of duplicated code

Description: There are three copies of a large segment of code in C++ performing the same task, namely processing audit data.[4] The three copies might have been obtained by cut-and-paste, and slightly modified in subsequent versions. This is an unsafe programming practice. The result is that the three copies have slightly different security properties. For instance, there are security issues in one of the three copies - see Sections 7.1.8 and 7.1.5.

This does not create a vulnerability by itself, therefore we do not list prerequisites and impact here. The purpose of this section is to point out an instance of an unsafe programming style that may lead to security problems.

[3] See Section 22.7.1.6 in the confidential Annex.

[4] See Section 22.7.1.7 in the confidential Annex.

7.1.8 Unity contains many small buffer overflows

Description: There are numerous occurrences of unsafe memory accesses throughout the Unity code. This is indicative of fragile programming style, which might lead to errors that lead to unpredictable behavior of the program and/or open vectors of attack.

In parts of code that process incoming election results and audit data, there are numerous instances of small buffer overflows (1 or 2 bytes). There are also 'hardcoded' buffer overflows, i.e. memory write accesses that are determined by a constant fixed in the code, and that are such that the access is guaranteed to be beyond allocated memory.[5]

While these problems will not result in the attacker running arbitrary code, they cause memory corruption. In case of 'hardcoded' buffer overflows, the corruption of memory will occur regardless of activities of an attacker.

This does not create an exploitable vulnerability by itself, therefore we do not list prerequisites and impact here. The purpose of this section is to point out an instance of an unsafe programming style that may lead to security problems.

7.1.9 SQL injection to bypass authentication in EDM, ESSIM, and Audit Manager

Description: The authentication process for the Election Data Manager (EDM), Ballot Image Manager (ESSIM), and Audit Manager components of the Unity election management system can be bypassed with a simple SQL attack.[6]

The authentication code used by both EDM and ESSIM is considered to be a part of the Audit Manager's functionality. The user manual for EDM explicitly states that the Audit Manager provides security for election and should always be used.

Scenario unity.1 in Section 9.3 demonstrates an attack using this vulnerability.

Prerequisites: The attacker needs access to the Unity election management system. The attacker also needs to know the name of the table that contains users' information. This can either be guessed or found in the Unity documentation; for example, it is in the functional specification for EDM.

Impact: This injection works by bypassing authentication for EDM, ESSIM, and Audit Manager. As a result, if user accounts are not enforced at the OS level, anybody could login to any of the three applications. Once the attacker is logged on, he/she could view in clear text or change the iVotronic passwords (which do not seem be available in clear text outside of EDM), or he/she could change logs for EDM and ESSIM (using the Audit Manager).

The same SQL injection does not work for other Unity modules, since they use different logging and authentication mechanisms, or they do not use any authentication mechanism at all.

7.1.10 An M100 PCMCIA card can be read multiple times without warning

Description: In the ERM module of Unity, if the operator chooses the "add-to-mode" settings when pro-

[5] See Section 22.7.1.8 in the confidential Annex.
[6] See Section 22.7.1.9 in the confidential Annex.

cessing a M100 card, the same card can be read in multiple times without warning the operator that such a thing has occurred. There is graphical notification that could alert the operator about which results have been read in and from what precinct they came, but there are no warnings that there were multiple reads of the same card. It would also not be obvious that this has occurred.

However, there are multiple modes for processing M100 cards, "add-to-mode" and "replace mode." The difference between the two modes is that in "replace mode" if a precinct has already been read into the ERM, then a warning is presented asking the operator if he or she really wants to replace the results for that precinct. In "add-to-mode", this is not the case, and all results are added together.

In large scale elections, with multiple polling devices per precinct, it is not clear whether the "replace mode" can be used, because it would imply that there is only one data device for returning results per precinct. With multiple polling devices per precinct this is not the case.

According to the documentation,[7] there appears to be no way to remove duplicated results once read into the ERM. The only way to correct the mistake would be to zero the results and start from the beginning.

Impact: Reading in results multiple times from a PCMCIA card would skew vote tallies and change the outcome of an election.

7.1.11 Assumptions on the environment for Unity are unspecified

Description: The assumptions for the environment in which Unity is running are left unspecified. This includes the configuration of the operating system (Microsoft Windows), networking environment, services running on the computer, etc. Thus it is not clear what are the security guarantees that Unity expects from the host system.

It is possible to assume that Unity will be run in a relatively isolated environment in county election headquarters and is used only by trusted staff. However, even in this case, many issues need to be carefully considered, including network setup, possible interference with other processes running on the machine, updates and patches to the operating system and removable media that are read by the machine. A concrete example is that "autorun" should be disabled for all removable media drives. If it is not the case, the program run by "autorun" may compromise the host machine.

The assumptions on the security of the host system are not specified by the vendor. Instructions for the operators for setting up and managing the server on which Unity is running are also not provided.

Impact: Many potential platform security issues may be present in any county Unity installation. Enumerating all issues caused by the host system and resulting attacks is beyond the scope of this study.

7.1.12 iVotronic Image Manager bundled with vulnerable Java Runtime Environment

Description: The Windows installer for the iVotronic Image Manager includes an install of Sun Java Runtime Environment 1.4.2_8. This version of Java has a known, exploitable vulnerability in its GIF image display code.[8] iVotronic Image Manager is written in Java and supports importing GIF graphics for political parties. These graphics are displayed by the iVotronic Image Manager when previewing ballots. Anyone

[7] *The Unity Election Reporting Manager Users Guide*

[8] US-CERT, *Vulnerability Note VU 388289: Sun Microsystems Java GIF image processing buffer overflow*. `http://www.kb.cert.org/vuls/id/388289`, January 2007.

who provides a GIF image to be imported into iVIM (or any GIF images downloaded from, e.g., Google) could potentially exercise this exploit.

We have not experimentally confirmed that this vulnerability is exploitable in iVotronic Image Manager.

Impact: A malicious GIF file could lead to arbitrary code executing as the user running iVotronic Image Manager, and writing to any files allowed for that user.

7.2 iVotronic

7.2.1 Election encryption/decryption key is recoverable from a PEB

Description: The per-election encryption key used to protect vote and audit data is trivially recoverable from a qualified PEB. The election key is itself encrypted using the election qualification code (EQC) and is stored as ciphertext on the PEB. However, the EQC is written to the PEB in unencrypted form. Hence, access to the PEB reveals the EQC which can then be used to decrypt the encrypted election key. This is the electronic equivalent to storing a secret in an impenetrable safe and then painting the safe's combination on its door.

Prerequisites: Recovering the encryption key requires physical access to a qualified PEB and the ability to communicate to it via an infrared (IR) link.

Impact: The encryption key is used by Unity and iVotronics to protect election definitions, vote information, and audit logs. Knowledge of the encryption key and physical access to a PEB permits the following actions:

- Arbitrary modification of data on the PEB, including election results (if the PEB has been used to collect votes from an iVotronic) and the election definition.
- Decryption of audit logs stored on the PEB or a Compact Flash card.
- The ability to emulate PEBs (e.g., using a Palm Pilot with an IR port) that are accepted by iVotronics (see Section 7.2.2).

7.2.2 PEBs may be emulated using infrared-capable devices

Description: The iVotronic communicates with PEBs using a proprietary IR protocol. Any device with IR capability can be used to emulate a PEB. Since PEBs and the iVotronic contain magnets and use magnetic switches to detect each other's presence (see Section 5.2.4), communicating with the iVotronic requires close proximity to the iVotronic's voter-facing PEB interface.

As is illustrated in Figure 7.1, we were able to emulate PEBs using a commodity Palm Pilot and a small magnet (the construction of smaller PEB emulators is certainly possible). Due to the PEB slot's location on the front of the iVotronic, it is difficult for poll workers to monitor for potential PEB emulators without sacrificing voter privacy.

Prerequisites: PEB emulators must be capable of communicating using ES&S' proprietary IR protocol. This protocol is simple (consisting primarily of *send-block* and *receive-block* commands) and can be easily understood by probing a legitimate PEB.

Figure 7.1: A PEB emulator running on a Palm Pilot simulates an initialization PEB during an open election, resetting all terminal passwords to "EVEREST"

Impact: The ability to emulate a PEB enables a wide range of serious poll worker and voter attacks. Attacks involving PEB emulation are described in Sections 7.2.4, 7.2.5, 7.2.8, 7.2.9, 7.2.10, 7.2.11, 7.2.12, and 7.2.13.

7.2.3 PEB Emulators can read and/or write arbitrary data on a PEB

Description: A PEB emulator can read and/or write arbitrary data on a PEB placed in close proximity.

Prerequisites: Since PEBs are activated by detecting the presence of a magnet, the PEB emulator must be placed in close proximity when reading or writing to the PEB. Malicious voters may have sufficient access to the PEB when signing in if the poll worker leaves the PEB on the voter sign-in table.

Impact: As described in Section 7.2.1, the EQC, election encryption key, election definitions, and any stored election results are easily discernible by reading the PEB. Additionally, by writing specially crafted blocks to the PEB, an attacker can surreptitiously load exploit code that can compromise the backend Unity system when results are read from the altered PEB (see Section 7.1.1).

7.2.4 Emulated PEBs permit multiple votes

Description: A voter who can emulate a PEB (by combining the attacks described in Sections 7.2.1 and

7.2.2) can use the PEB emulator (e.g., a Palm Pilot) to authorize multiple voter sessions.

Prerequisites: To authorize a voter session, the voter needs the correct election definition, EQC, and election key. All three may be obtained by querying a PEB (see Section 7.2.3).

Impact: Once a PEB has been read, emulated PEBs can be used to authorize voting sessions in any iVotronic terminal that shares the same EQC, election definition, and election key as the cloned PEB. Hence, a malicious voter can repeatedly authorize himself and cast multiple votes. Since the emulated PEB is accepted by all iVotronics with the same configuration, a distributed version of this attack is possible. Here, pertinent data from a single cloned PEB is distributed to multiple attackers, all of whom may cast multiple votes.

7.2.5 Buffer overflow in poll opening process is exploitable

Description: The iVotronic does not verify that allocated memory buffers are sufficiently large to store variable length strings read from the (possibly emulated) PEB. During the poll opening process, the iVotronic copies variable length strings from the PEB into memory allocated on the machine's stack. If the PEB contains specially crafted large strings, an exploitable buffer overflow occurs.[9] During our testing, we confirmed that a specially crafted PEB or PEB emulator can exploit this buffer overflow to take complete control of the iVotronic.

Prerequisites: The PEB or PEB emulator must contain a suitable malformed ballot definition. Additionally, the (emulated) PEB must have an EQC that matches that stored in the iVotronic.

Impact: The buffer overflow allows an attacker to compromise an iVotronic terminal and take complete control over it. Once compromised, the following actions are available to the attacker:

- upload unauthorized and malicious firmware via either the PEB slot or the unprotected serial port (see Section 7.2.14);
- download current election results and/or audit data via either the PEB slot or the serial port;
- modify or erase current election results and/or audit data;
- print arbitrary text to the VVPAT (including forged VVPAT ballots);
- modify or erase the contents of an inserted PEB or Compact Flash card;
- write random data to the firmware, causing the iVotronic to become unusable until it can be disassembled and reprogrammed

7.2.6 Buffer overflow in "Hotspot" image processing is exploitable

Description: The logic that reads image files from the Compact Flash card has an exploitable stack-based buffer overflow.[10] These *hotspot* image files define areas on the screen that function as touchscreen buttons. To support a variable number of hotspots, the headers for such image files are variable length. If the image file specifies a larger-than-expected number of hotspots, memory copied from the Compact Flash card overflows the allocated buffer.

[9] See Section 22.7.2.5 in the confidential Annex.
[10] See Section 22.7.2.6 in the confidential Annex.

The iVotronic contains two independent code patterns for accessing hotspot images. The first pattern, which is used only in one location, reads a fixed number of bytes from the image file into a local buffer of inadequate size. The second pattern, which occurs in ten different locations, copies an unbounded number of bytes from the image file into a memory location on the stack.

Due to the manner in which variables are arranged on the stack, the first code pattern allows an attacker to overwrite the called function's return address, but not with data under his/her control. Therefore, it cannot be used to execute malicious code. However, it can be used to crash the iVotronic, causing denial-of-service. In contrast, the second pattern allows the attacker to execute arbitrary code.

Prerequisites: To carry out the attack, the attacker must write an appropriately crafted ballot header to the Compact Flash card. There are a number of different methods an attacker might employ to introduce a malicious Compact Flash card into the voting process:

- An election insider could prepare a malicious Compact Flash card that is then sent to precincts;
- A virus that previously compromised the Unity server (see Section 7.1) could modify the Hardware Programmer Manager (HPM) to create malicious Compact Flash cards;
- A poll worker could introduce a malicious Compact Flash card either before opening the polls or during a sleepover;
- A voter could extract the Compact Flash card from the iVotronic, modify it to contain the exploit, and reinsert the (now malicious) card into the iVotronic. This approach requires the voter to break and reattach seals. Additionally, the iVotronic will stop responding when the Compact Flash card is removed. However, a voter could claim that a crash occurred during the normal voting operation and convince the poll worker to reactivate the machine. If the machine is reactivated, the exploit code will then be executed and the iVotronic will be compromised.

Impact: The buffer overflow exploit allows the attacker to execute arbitrary code on the iVotronic, permitting any of the actions described in Section 7.2.5.

7.2.7 Buffer overflow in Supervisor iVotronic initialization process is exploitable

Description: The iVotronic supervisor terminal election initialization process contains a stack-based buffer overflow.[11] A device that is connected to the supervisor terminal during initialization (usually the Unity backend system) can exploit this overflow and execute arbitrary code on the supervisor terminal.

Prerequisites: To exploit the buffer overflow, an attacker needs to either compromise the Unity backend system or connect the supervisor terminal to a device (via its serial port) that mimics Unity.

Impact: Since the supervisor terminal is responsible for configuring PEBs for an election, a compromised iVotronic supervisor terminal can write malicious data to the PEBs in order to exploit vulnerabilities in iVotronic voter terminals (e.g., by exploiting the vulnerability described in Section 7.2.5).

[11] See Section 22.7.2.7 in the confidential Annex.

7.2.8 Service Menu Mode in iVotronic does not require properly configured PEB

Description: By pressing the **VOTE** button and inserting a PEB, the iVotronic enters a service menu mode which allows various configuration and election settings to be viewed and/or modified. In our tests, we found that few checks are carried out to determine whether the inserted PEB is properly qualified. Consequently, any PEB (even one not configured for the current election) can be used to enter the service menu mode.

Prerequisites: Most service menu items require the user to enter a password. The iVotronic defaults to specific passwords unless they have been explicitly changed. Note that password checks are bypassed when a (possibly emulated) Factory QA test PEB is inserted (see Section 7.2.10).

Impact: Anyone with an unqualified PEB (or a device that acts like one) and knowledge of the appropriate passwords can carry out the following actions:

- lock the terminal;
- close the terminal early;
- adjust the date and time (which, in effect, may also close the terminal);
- disable or enable audio ballots for ADA versions of the iVotronic;
- (mis)calibrate the touchscreen (see Section 7.2.13)

7.2.9 (Emulated) Initialization PEB can close polls and reset passwords

Description: An initialization PEB may be used to set terminal menu passwords to arbitrary values and close the polls early. The initialization PEB can be inserted at any time, including when polls are open. We have confirmed in our testing that it is straightforward to emulate an initialization PEB using a handheld device with an infrared port (see Figure 7.1). The initialization process takes approximately one minute to complete and hence can easily be accomplished by a malicious voter.

Prerequisites: A specially formatted PEB (or a PEB emulator) can carry out this attack. No election secrets (e.g., the EQC or the election encryption key) are required.

Impact: By closing the polls before the close of the election, this attack causes denial-of-service against a particular iVotronic terminal. Once closed, an iVotronic terminal cannot be reopened unless a new ballot definition is loaded onto the terminal.

The initialization PEB also causes all iVotronic passwords to be set to values configured on the initialization PEB. By setting these passwords to non-default values or values that cannot be entered using the on-screen keyboard, the attack makes it difficult to restore the terminal to a usable state until another initialization PEB is used to reset the passwords.

7.2.10 (Emulated) Factory QA PEBs bypass all password checks

Description: The iVotronic system contains a backdoor that bypasses all passwords when a *Factory QA PEB* (also called a *Debug PEB*) is inserted into the iVotronic.[12] A Factory QA PEB is a special PEB that

[12] See Section 22.7.2.10 in the confidential Annex.

follows the same protocols as a standard PEB. When a PEB is inserted into the iVotronic, the iVotronic polls the PEB for its type. If the type corresponds to a Factory QA PEB, the iVotronic enters a state in which all password prompts are automatically bypassed.

Emulating a Debug PEB is as easy as emulating a standard PEB (see Section 7.2.2). All PEBs (supervisor, voter, and debug) have exactly the same hardware and use the same protocol to interact with the iVotronic terminal. Debug PEBs differ only in the PEB type identifier they report to the iVotronic. Consequently, a PEB emulator can easily identify itself as a debug PEB.

Prerequisites: The debug PEB can be emulated using any device with an IR port (e.g., Palm Pilot). The emulator must be capable of communicating using ES&S' proprietary IR protocol. This protocol is simple (consisting primarily of *send-block* and *receive-block* commands) and can be easily understood by probing a legitimate PEB.

Impact: Factory QA PEBs are presumably used for testing iVotronics as part of internal quality assurance testing. They function exactly like any other PEB except that they bypass all password checks in all iVotronic menus. Debug PEBs allow any voter or poll worker to access important service menu items (see Section 7.2.8) on the iVotronic without the knowledge of any passwords or other election-specific secrets. Further details of various attacks using emulated Factory QA PEBs are described in Sections 7.2.11, 7.2.12, 7.2.13, and 7.2.14.

7.2.11 (Emulated) Factory QA PEB can clear a terminal, erasing all vote and audit data

Description: A (possibly emulated) Factory QA PEB can be used to initiate the iVotronic's clear-and-test procedure, causing all vote data and audit logs to be deleted from the iVotronic flash memory. Clearing a terminal takes less than a minute and can be done surreptitiously by a malicious voter with a Factory QA PEB or PEB emulator.

Prerequisites: If an election has already been loaded on the iVotronic, an Initialization PEB (or emulator) first needs to be used to perform terminal initialization (see Section 7.2.9). The initialization process closes the open election and enables the "Clear and Test Terminal" option in the iVotronic services menu. No election-specific secrets are required.

Impact: Using a (possibly emulated) Factory QA PEB, a malicious poll worker or stealthy voter can quickly (i.e., within a few minutes) conduct the following attacks:

- Delete all previously recorded vote data;
- Upload new firmware to the iVotronic through the Compact Flash interface. Since the Compact Flash card slot is normally protected by a tamper evident seal (of questionable quality, see Section 6.1.2), uploading firmware in an undetectable manner requires the removal and subsequently replacement (or reattachment) of the security seal;
- Exploit the buffer overflow in the terminal opening process (see Section 7.2.5) to take complete control of the iVotronic. This includes the ability to upload firmware via either the unprotected serial port or the iVotronic's PEB interface.

7.2.12 (Emulated) Factory QA PEB can lock a terminal

Description: A (possibly emulated) Factory QA PEB permits a malicious voter to access the services menu without a password and lock the terminal. The locked terminal cannot be used for voting until it is unlocked by keying in the unlock password.

Prerequisites: A Factory QA PEB or emulator is sufficient to lock the terminal. No election-specific secrets (e.g., EQCs or election encryption keys) are required.

Impact: A locked terminal cannot be used for voting until it is unlocked by keying in the unlock password. Assuming poll workers know the unlock password, this is a minor denial-of-service attack which disrupts the voting process.

7.2.13 iVotronic touchscreens can be miscalibrated to deny certain ballot selections

Description: If the **VOTE** button is pressed while a Supervisor or Factory QA PEB is inserted into an iVotronic, the iVotronic enters the service menu. By pressing the **VOTE** button again, the iVotronic enters the calibration screen mode in which the user is required to press crosshairs shown on the screen to calibrate the touchscreen's sensors. By deliberately touching the screen at incorrect places, it is possible to carefully miscalibrate the screen, preventing the voter from making certain selections while marking a ballot.

Prerequisites: A (possibly emulated) Supervisor or Factory QA PEB is required to recalibrate the iVotronic. It is not necessary for the PEB to contain the correct EQC or any other election specific information. No password checks are used for screen calibration, so even an unqualified PEB from a different election can be used.

Impact: By carefully miscalibrating the iVotronic, a malicious voter with access to a PEB or PEB emulator can prohibit future voters from making certain ballot selections.

7.2.14 Connection to RTAL/VVPAT printer is physically unprotected

Description: The RTAL/VVPAT printer is connected via an unprotected serial port on top of the iVotronic (see Figure 7.2). The VVPAT printer cable is not protected by any seals or fastened by any screws and can be easily unattached by a voter.

Prerequisites: The attacks described below require only a hardware device with a serial interface.

Impact: Access to the iVotronic's serial interface enables the following attacks:

- A voter who can access the election services menu (e.g., by emulating a Factory QA PEB, see Section 7.2.10) can download encrypted audit log information to a device with a serial port. Here, the attacker disconnects the RTAL printer and connects the iVotronic's serial port to a handheld device. If the attacker has knowledge of the election encryption key (e.g., by probing a PEB, see Section 7.2.1), s/he can decrypt the audit log to obtain voting records.
- The iVotronic communicates with the printer using a simple protocol via the serial port. By connecting a device to the exposed printer connector, it is possible to print counterfeit audit information as well as additional VVPAT ballots. This is particularly troublesome given that Ohio law specifies that "*[for] any recount of an election in which ballots are cast using a direct recording electronic voting machine*

Figure 7.2: The connection to the RTAL/VVPAT printer port is physically unprotected

with a voter verified paper audit trail, the voter verified paper audit trail shall serve as the official ballot to be recounted."[13]

- The RTAL/VVPAT printer has the ability to rewind the printer tape by approximately one-half inch. (This scrollback limitation is due to hardware, so a malicious voter cannot sufficiently rewind the printer to read printouts from previous voter sessions.) An attacker can utilize this functionality to repeatedly print over the same line of text, creating the appearance of a paper jam. This attack can diminish voter confidence in the election, particularly since VVPAT records are considered the official ballots of record.

- A malicious voter can implant a small serial device between the printer cable and the iVotronic's serial interface. If not removed, such a device could intercept all communication between the terminal and the printer, including voter selections. The recorded votes may be recovered either by adding a wireless transmitter to the serial device or by retrieving the device at the close of an election (perhaps by another voter acting as an accomplice).

- A small device may be attached to the iVotronic's serial interface that mimics the VVPAT/RTAL printer. The iVotronic will incorrectly detect an attached printer, although no future audit information or VVPAT ballots will be printed to the disconnected VVPAT/RTAL printer.

[13] *Ohio Revised Code (O.R.C.) § 3506.18: Electronic voting machine - verified paper audit trail as official ballot in recount.* ⟨URL: `http://codes.ohio.gov/orc/3506`⟩.

7.2.15 Audit data is insufficiently randomized

Description: The iVotronic does not properly randomize voter selections in its audit logs. To maximize voter privacy, voter selections should be randomized according to a uniform probability distribution (that is, each audit record in the audit log should have equal probability of corresponding to a particular voter record).

The iVotronic uses a weak randomization procedure that results in a partial ordering of voter selections.[14] When a voter casts a ballot, the voter's selections are uniformly at random assigned to one of approximately thirty sectors in the iVotronic's internal flash memory. The voter selection is appended to the first available location in the chosen sector (if the sector is full, a new sector is random selected). Consequently, each sector contains a sequential ordering of voter selection images.

Prerequisites: Access to audit information is required.

Impact: An election official with access to audit information may ascertain a partial ordering of voter selections. Such information may reveal voter trends throughout the election.

7.2.16 VVPAT barcodes contain timestamps

Description: After a voter casts a ballot, a barcode containing the current time is printed to the VVPAT printer.[15] These "timestamps" can be used to reconstruct the ordering of votes, even if VVPAT records are detached from the roll to mask vote order (and protect voter privacy) after the close of the election.

Prerequisites: Access to VVPAT records is required.

Impact: An attacker can reconstruct the order of VVPAT printouts using the timestamp encoded in the barcodes, revealing voter trends throughout the election. Additionally, if the order of voters is also known, the attacker can determine how a particular voter voted.

7.2.17 VVPAT barcodes contain vote information

Description: After a voter casts a ballot, a barcode encoding the voter's ballot selections is appended to the VVPAT record.[16] An example of such a barcode is depicted in Figure 7.3. Under the assumption that most voters cannot decipher barcodes, encoding vote information on a VVPAT that cannot easily be *voter verified* constitutes a dangerous and unnecessary practice.

The purpose of such barcodes is unclear. As noted in Section 7.2.14, Ohio law stipulates that "*[for] any recount of an election in which ballots are cast using a direct recording electronic voting machine with a voter verified paper audit trail, the voter verified paper audit trail shall serve as the official ballot to be recounted.*"[17] The tabulation of election results based on barcodes rather than on human comprehendible text eliminates the safeguards promised by voter verified paper audit trails. Interestingly, the iVotronic operator's manual[18] recommends the use of a 2D barcode reader for tabulating votes in the event of a recount, although

[14] See Section 22.7.2.15 in the confidential Annex.

[15] See Section 22.7.2.16 in the confidential Annex.

[16] See Section 22.7.2.17 in the confidential Annex.

[17] 'Ohio Revised Code (O.R.C.) § 3506.18: Electronic voting machine - verified paper audit trail as official ballot in recount.' (as in n. 13).

[18] Election Systems and Software, Inc., *The iVotronic Voting System Operator's Manual version 9.1*. January 2006.

Figure 7.3: A VVPAT barcode encoding voter selections

the document also warns that barcodes should not be used if prohibited by law.[19]

Furthermore, the counterargument that barcodes significantly improve tabulation efficiency and do not undermine security since they can be verified by the accompanying text is fundamentally flawed. Validating all barcodes requires processing all text portions of the ballot, eliminating the efficiency advantage provided by the barcode. Hence any speedup due to the barcode comes at the expense of failing to validate the voter-verified portions of the audit trail.

If barcodes are not used, they should not be included on VVPAT records. The presence of barcodes may induce an election worker (who is likely under substantial pressure to quickly tabulate election results) to take the quick-and-easy path and utilize the barcodes.

Prerequisites: Reading barcodes from a VVPAT printout requires a commodity 2D barcode reader.

Impact: Election officials may tabulate votes using the barcodes printed on VVPAT records, eliminating the safeguards offered by voter verified paper audit trails.

[19]Ohio's recount procedures are vague on the question of permitted uses of barcodes during recounts. An Ohio Secretary of State Directive stipulates that hand counting must be done by "physical examination of the VVPAT roll" and permits "electronic tabulation" if the hand count shows no discrepancy. It is unclear if "electronic tabulation" includes counting VVPAT ballots with a barcode reader. *See:* Ohio Secretary of State, *Directive 2007-30: Recount Procedures*, retrieved November 25, 2007 from `http://www.sos.state.oh.us/sos/ElectionsVoter/directives/2007/Dir2007-30.pdf`

7.2.18 Accuracy testing mode detectable

Description: The logic and accuracy testing (LAT) of the iVotronic can be performed in two different ways, either automatically or manually. Automatic testing seems to be what is most common. The automatic testing is performed by selecting a special logic and accuracy command in the administrator menu of the iVotronic. The automatic test will then cast votes in a specific pattern without any user intervention. In the manual mode the user manually casts votes, and the result is tallied as a special logic and accuracy result.

The logic and accuracy test is effective at catching problems with the ballot programming, but has very limited security benefits. Any malicious firmware running on the iVotronic can detect that the logic and accuracy test is running, and function properly during this test. During an automatic test, no GUI functionality is ever tested. All our example exploits hook into the GUI functions to perform the malicious activity, so no action at all needs to be taken by the exploit writer to pass the automatic test mode. The manual test mode does utilize the GUI, but a special variable in the iVotronic is set to indicate that test mode is running. The exploit code can check this variable, and whenever it is set the code can behave properly.

Prerequisites: The malicious firmware needs to know what special variable to check to determine if it is in a logic and accuracy test or not.

Impact: A malicious firmware can test the special variable and detect the current operating mode and perform different functions accordingly. For example, the malicious firmware could perform a correct count during the LAT procedure and then introduce errors during the actual voting procedure.

7.3 M100

7.3.1 The M100 does not password protect the procedure for firmware uploads

Description: The procedure for uploading firmware to the M100 does not require a password. If the PCMCIA card inserted into the M100 contains new firmware, the M100 displays the install new software menu, permitting anyone to install unverified firmware.

Prerequisites: To install new firmware an attacker requires a properly formatted PCMCIA card and physical access to the M100 PCMCIA slot.

Impact: The firmware on the M100 controls every procedure and operation of the M100. Therefore, malicious firmware would allow an attacker complete control of the machine.

Possible attacks include:

- altering vote totals
- falsifying results tapes
- altering menu displays
- denying service
- removing the ability to install additional firmware (this subsequently makes the M100 unrecoverable unless by the use of additional hardware)

7.3.2 The M100 does not perform cryptographic integrity checks on firmware uploads

Description: There are no cryptographic integrity checks on new firmware before it is uploaded onto the M100. Any correctly formatted M100 firmware (including malicious code created by an attacker) can be loaded onto a PCMCIA card and the M100 will accept it as valid.

Prerequisites: An attacker needs to have knowledge of the hardware components of the M100 in order to create malicious firmware and would need physical access to the M100 in order to install it. Knowledge of the hardware components of the M100 is not particularly hard to obtain and is within the scope of a sufficiently motivated attacker.

Impact: As stated above in Section 7.3.1 the installation of malicious firmware on the M100 gives the attacker complete control of the machine.

7.3.3 The M100 uses the same facility for configuring an election as it does for firmware uploads

Description: The procedure for installing new firmware and the procedure for loading a new election definition on the M100 are both performed by inserting a PCMCIA card into the PCMCIA card slot on the M100 and turning the machine on. If a PCMCIA card containing malicious firmware was placed in an M100, due to issues described in Sections 7.3.1 and 7.3.2, it is possible that a distracted poll worker could unknowingly and unintentionally install malicious firmware.

Prerequisites: A deliberately altered PCMCIA card

Impact: A poll worker, given an infected PCMCIA card, going through typical election day procedures could unintentionally upload malicious firmware to the M100. Additionally, if Unity were infected, the malicious firmware could be distributed and created directly from Unity and installed by a poll worker on election day.

7.3.4 The M100 locks are easily manipulated

Description: The M100 cabinet, Scanner, Ballot Box, PCMCIA slot and power on/off switch are locked with simple cam locks which can be easily picked in a short time using improvised tools.

Prerequisites: An attacker would require privacy and a paper clip.

Impact: Access to the PCMCIA slot would permit an attacker to perform any of the attacks mentioned here which make use of a PCMCIA card. Additionally, since these locks do not provide any evidence of tampering, the fact that they are so easily bypassed makes discovery of this attack difficult.

7.3.5 The keys for the M100 locks are the same across all M100 machines

Description: There are two keys for the M100. The ballot box key locks the M100 scanner in place and protects the PCMCIA slot, ballot box, and the cabinet doors. The scanner key, turns the power on and off to the M100. These two keys are differ from each other, but are the same for all M100 machines. These keys are available online from ES&S for $5.00 each.[20]

[20] Scanner Key Part Number #75506, Ballot Box Key Part Number #75505.

Prerequisites: Access to either a scanner or ballot box key or both.

Impact: A compromise of the ballot box key would grant the attacker access to the PCMCIA card slot, and complete access to the ballots in the ballot box where an attacker could remove or add ballots. A compromise of scanner key would give an attacker the ability to turn on and off the M100.

A compromise of both keys would create a perfect scenario for an attacker to install malicious firmware. The attacker would first need to compromise the ballot box key, after which he can insert a malicious firmware PCMCIA card. The attacker would then compromise the scanner key, turning the machine off then on in order to reach the software installation menu. Once in the software installation menu, the attacker can install malicious firmware and finish the installation by again turning the machine off then on. This could occur without detection.

Additionally, the key blanks for a scanner and ballot box key are easily duplicated, so a compromise of either key could affect machines nation-wide.

7.3.6 CRC routines used on the M100 are easily derivable

Description: CRC routines are used in many places by the M100 to verify the integrity of the data on the PCMCIA card. An attacker can easily discover the constants used to generate the CRC, alter the data on the PCMCIA card, and regenerate the CRC.

Prerequisites: Access to a valid M100 PCMCIA card and a commodity PCMCIA card reader/writer.

Impact: As described in Section 6.4, CRC routines do not provide cryptographic integrity protection. An attacker can write anything to the PCMCIA card, regenerate the CRC and it will be accepted as valid by the M100.

7.3.7 The M100 PCMCIA cards do not contain cryptographic integrity checks

Description: There is no cryptographic integrity protection used on the M100 PCMCIA card. This means that there is no way to verify that the data on the card was created from an authorized source.

Prerequisites: Knowledge of the CRCs as in Section 7.3.6.

Impact: An attacker can arbitrarily change data on the card. The altered data would then be accepted as valid by either Unity or the M100.

7.3.8 M100 uses data offsets defined on the PCMCIA card to determine pointers used by the firmware

Description: During the boot process, the M100 loads a number of structures from the ballot PCMCIA. Some of these structures contain a reference to other structures in the form of an offset value.

After the structures have been loaded, a function is invoked to translate each offset to a pointer. This is accomplished by replacing each offset with the value of the base pointer increased by the offset. Since the attacker can control the offset values and since the function does not check that the resulting pointers are correct, it is possible to make these pointers point to any location in memory. Furthermore, because the code then writes to some of the pointed structures, it is possible to override any sequence of 4 bytes in memory.

Note: Our effort with this vulnerability was hampered by the fact that we did not have all of the source code for the M100 firmware. We requested the missing firmware source numerous times, but it was never delivered. As a result, we spent a lot of time and energy reverse-engineering the firmware.

This is of particularly concern, because it indicates that the M100 contains firmware that is not contained in the firmware provided to the Independent Test Authorities (ITA). The extra firmware is usually not updated, and seems to be the kernel of the operating system running on the scanner (QNX).

Although the kernel is developed by a third party and could be considered a COTS component, it has to be configured with the appropriate hardware drivers, etc. in order to be used with a particular kind of hardware. Since this configuration would be specific to the M100, it should not be considered COTS.

Prerequisites: The attacker needs a way to craft a special PCMCIA card and needs access to the M100. Sections 7.3.6 through 7.3.7 discuss the details of the prerequisites for this vulnerability.

Impact: A number of different attacks can result from this vulnerability. Perhaps the most prevalent is the ability to perform a denial-of-service that could disrupt the ballot counting process. Addition denial-of-service attacks can be performed on the audit data region of the card or on the card header section, the region that defines the layout for the entire card.

It may even be possible to change vote data, that is if the write that follows the altered pointer is done during poll closing. The M100 could unintentionally write into the counter block region changed votes for particular races and subsequently stuff the ballot box.

7.3.9 M100 accepts counterfeit ballots

Description: Portions of the paper ballots read by the M100 are printed using special inks whose reflection properties cause them to be ignored by the optical scanner. To insure against counterfeit ballots, specially designated areas of the ballot are printed using this special ink. To detect counterfeit ballots, the M100 checks for the presence of marks in these specially designated portions of the ballot. Because photocopied ballots are printed using a single (and detectable) ink, a normally copied ballot reproduces these marks. The optical scanner detects no markings in these areas when legitimate ballots are used and perceives the marks if the ballot is a simple photocopy.

Visual inspection is sufficient to distinguish the two inks used to print the paper ballots. By covering areas printed using the reflective ink, it is possible to create duplicate ballots that are accepted by the M100. Although ballots without the special portions may fail visual inspection if they are ever manually scrutinized (since the portions printed using the reflective ink are absent), better forgeries can be constructed by obtaining IR reflective non-read ink. Such ink is available for general purchase at online vendors, or can be easily made from commodity ink jet printer supplies . The M100 accepts a commodity paper in a variety of weights.

Prerequisites: Access to the M100 ballot box; see 7.3.4 and 7.3.5.

Impact: A motivated forger can produce paper ballots that visually appear legitimate and that are accepted by the M100.

7.4 M650

7.4.1 M650 runs executable on Zip Disk on power up

Description: On boot, the M650 checks for the presence of a *firmware update disk*. Firmware update disks are commodity Zip disks provided by ES&S that contain updated programs (binary executables) and an installer script. The M650 determines whether a Zip Disk is a firmware update disk by checking for the existence of three files.[21] If these files exist and the reported version number does not match the currently installed firmware version, the installer script located on the Zip Disk is executed. Only the contents of the file containing the update version number are examined. That is, the M650 checks only for the existence of the three files and performs no integrity or authenticity checks. Hence, any person with physical access to the machine can load new software onto the M650, either on, before, or after the election. As is also the case with the M100 (see Section 7.3.3), both firmware and ballot definitions are loaded through the same medium (in this case, the Zip Disk drive). By surreptitiously inserting malicious firmware onto a ballot definition disk, a corrupted Unity system could carry out this attack.

Figure 7.4 shows the display on the M650 after a forged firmware update disk was used to load unauthorized (in this case, benign and conspicuous) software onto the machine.

Prerequisites: To successfully load new software onto the M650, the attacker must learn the three filenames that convince the M650 that an inserted Zip Disk is a firmware update disk. These filenames can be easily obtained either by cloning a legitimate firmware update disk or by examining the firmware update procedure, a plaintext shell script stored in the M650 (although the latter technique requires prolonged and unfettered access to the machine). Any person with knowledge of these three filenames and approximately three minutes of unmonitored access to the M650 can replace all software on the machine with malware.

Impact: An attacker who can forge a firmware update disk can take total control over the M650. This includes the ability to learn election results, change results, mimic the behavior of an uncompromised M650 (to avoid detection), thwart future attempts to load new firmware (while reporting success to the operator), and/or cause the M650 to become inoperable until its internal storage can be reimaged using separate and uncompromised hardware.

7.4.2 M650 does not authenticate election definition and election macro language (e-code) files

Description: The M650 accepts any well-constructed election definition file or election macro language (e-code) file located on an inserted Zip Disk. No cryptographic authentication checks are used to ensure that the loaded definitions correspond to the paper ballots. Malicious election definitions and e-code can be trivially loaded by inserting a Zip Disk when the system is booted.

Prerequisites: The attacker requires a Zip Disk containing a seemingly valid election definition (that is, one that conforms to the data structures defined in the M650 Functional Specification document.[22]) The election definition and e-code obtained from a legitimate Zip Disk can be used as a template to construct a malicious election definition.

Impact: Using a malicious election definition Zip Disk, an attacker can trivially swap candidate positions,

[21] See Section 22.7.4.1 in the confidential Annex.

[22] Election Systems and Software, Inc., *Software Specifications: Model 650 Central Count Ballot Tabulator, version 2.0.0.0*. July 2004.

Figure 7.4: M650, with display controlled by forged firmware update disk

ignore arbitrary positions on the ballot, and/or alter the weight (the number of votes) assigned to a particular oval position on the ballot.

7.4.3 M650 fails to validate variable-length strings

Description: The M650 fails to check that constant sized buffers are sufficiently large to hold variable-length strings. A malicious ballot definition can define large strings that exceed the storage capacity of the allocated buffer. In particular, the M650 does not check that the length specified in the ballot definition for the election title does not exceed the number of bytes allocated for its buffer.[23] By specifying a large election title, a malicious ballot definition can overflow the allocated buffer and write arbitrary data to the M650's memory.

Prerequisites: To overflow the stack, the attacker needs a Zip Disk containing a seemingly valid election definition (that is, one that conforms to the data structures defined in the M650 Functional Specification document.[24]) The election definition obtained from a legitimate Zip Disk can be used as a template to construct a malicious election definition.

Impact: Due to its location on the stack, it does not appear that the stack overflow can be exploited to run arbitrary code on the M650. However, it is likely that a malicious ballot definition can sufficiently corrupt

[23] See Section 22.7.4.3 in the confidential annex.

[24] Election Systems and Software, Inc., 'Software Specifications: Model 650 Central Count Ballot Tabulator, version 2.0.0.0' (as in n. 22).

the M650 to cause the machine to halt until it is rebooted. It may also be possible to corrupt election results, although such an attack has not been confirmed.

7.4.4 M650 fails to protect against integer overflows

Description: The M650 dynamically allocates memory on the heap to store indices for the precincts (these indices are later used in the tabulation process). The amount of allocated memory is based on the number of precincts reported in the election definition. For example, if there are 10 precincts and 20 bytes are required to store an index for each precinct, then 200 bytes should be allocated. However, the M650 fails to ensure that the product of the number of precincts and the storage requirement of each precinct does not exceed the maximum value that can be represented as an integer. If the product exceeds this limit, then an integer overflow occurs (in which case the result of the multiplication "rolls over" and appears quite small) and insufficient memory is allocated. An attacker can force this integer overflow by specifying a large number of precincts in the election definition file, allowing him to write arbitrary data onto the heap (the amount of data is limited only by the storage capacity of the Zip Disk).[25]

Prerequisites: To overflow the heap, the attacker needs a Zip Disk containing a seemingly valid election definition (that is, one that conforms to the data structures defined in the M650 Functional Specification document.[26]) The election definition obtained from a legitimate Zip Disk can be used as a template to construct a malicious election definition.

Impact: Under some circumstances, the heap overflow may be exploited to inject code that will take control of the M650.

7.4.5 M650 accepts counterfeit ballots

Description: Portions of the paper ballots read by the M650 are printed using special inks whose reflection properties cause them to be ignored by the optical scanner. To ensure against counterfeit ballots, specially designated areas of the ballot are printed using this special ink. To detect counterfeit ballots, the M650 checks for the presence of marks in these specially designated portions of the ballot. Because photocopied ballots are printed using a single (and detectable) ink, a normally copied ballot reproduces these marks. The optical scanner detects no markings in these areas when legitimate ballots are used and perceives the marks if the ballot is a simple photocopy.

Visual inspection is sufficient to distinguish the two inks used to print the paper ballots. By covering areas printed using the IR reflective non-read ink, it is possible to create duplicate ballots that are accepted by the M650. Although these ballots may fail visual inspection if they are ever questioned (since the portions printed using the reflective ink are absent), better forgeries can be constructed by obtaining IR reflective non-read ink. Such ink is available for general purchase at online vendors, or can be easily made from commodity ink jet printer supplies. The M650 accepts forged ballots made of commodity paper in a variety of weights.

Prerequisites: The M650 is a batch scanner used primarily to scan mail-in (absentee) ballots. To be scanned, forgeries must be mailed to the appropriate office in official election envelopes (which, of course, may also be forged).

[25] See Section 22.7.4.4 in the confidential Annex.

[26] Election Systems and Software, Inc., 'Software Specifications: Model 650 Central Count Ballot Tabulator, version 2.0.0.0' (as in n. 22).

Impact: A motivated forger can produce paper ballots that visually appear legitimate and that are accepted by the M650.

CHAPTER 8

ES&S SOFTWARE ENGINEERING ISSUES

It is widely recognized that large and complex software is more difficult to implement correctly and securely than small and less intricate software. The diffuse control flow, data sources, databases, data structures, and usage patterns inherent in large systems decrease the ability to reason about the behavior of the software. Consequently, complex design and implementation permit the introduction of bugs, some of which may be exploited by malicious adversaries.

Software engineering techniques – "best practice" methodologies and tools used to develop and test complex software – attempt to reduce the number of such bugs. Although no software engineering technique can eliminate all complexity or guarantee secure code, common software engineering practices are effective at improving the correctness and the resiliency of the system. Good software engineering practices are critical to develop secure and reliable software, and are particularly vital in the security-sensitive context of electronic voting. In contrast, bad software engineering practices indicate haphazard design and implementation that is potentially rife with error. Hence, the software engineering practices employed by a system often speak to the overall quality of the code.

In this section, we identify design and coding practices which deviate from our perception of good software engineering techniques.

8.1 Complexity

We were provided with nearly 670,000 lines of code written in 12 programming languages and compiled for five hardware platforms (see Figure 8.1). (Lines of code were counted using SLOCCount[1] and the Linux *wc* utility.) Given the size and breadth of the code base, it is unlikely that any quality assurance (QA) process could conduct a comprehensive analysis of the entire system. As a result, the QA process will likely fail to find unintentional bugs and is even less likely to find malicious code surreptitiously inserted into the source code.

8.1.1 Programming Languages

Approximately 63% of the source code is written using programming languages that are *memory unsafe*. These languages (C, C++, and assembly) allow several classes of vulnerabilities, including stack and heap overflows, integer overflows, and format string attacks. More modern programming languages (for example,

[1] David A. Wheeler, *SLOCCount*. ⟨URL: `http://www.dwheeler.com/sloccount/`⟩.

Component	Platform	Language	Lines of Code
Unity 3.0.1.1	Intel 686 (Pentium)	BAT scripts, C, C++, COBOL, Java, Visual Basic	364,136
AIMS 1.2	Intel 686 (Pentium)	C++, SQL, Visual Basic	59,984
iVotronic firmware 9.1.6.4	Intel 80386	C, x86 assembly	87,157
iVotronic PIC source code	Microchip PIC Microcontroller	C, PIC assembly	1,753
M100 firmware 5.2.1.0	Intel 80286	C, x86 assembly, shell script	29,952
M650 firmware 2.1.0.0 (including "display" utility)	Intel 686 (Pentium)	C, C++ shell script	22,458
VAT (AutoMARK)	Intel IXP425	ARM assembly, C, C#, C++, Visual Basic	104,305
			Total: 669 745

Figure 8.1: Platforms and lines of code associated with each component of the ES&S system.

Java and C#) are *type safe* and are not susceptible to such memory violations, and are therefore sometimes favored when developing security-critical applications. Although techniques exist for annotating C code (comprising roughly 23% of the ES&S source code) to achieve type safety,[2] such approaches were not utilized.

8.1.2 Third-Party Software

ES&S makes use of third-party software, including the Windows and QNX operating systems, CompactFlash device drivers, and PCMCIA flash device drivers. Although such software is useful for rapid software development, it is difficult to analyze the security properties of closed-source third-party systems. Without access to the source code or a time-consuming analysis of the third party software's binary objects, these third-party systems are used as "black boxes" under the (possibly incorrect) assumption that they are secure. Since any vulnerability in a third-party product is automatically inherited by the system using it, such trust is unwarranted for security conscious applications.

8.1.3 Insufficient Documentation

Creating and configuring a ballot definition is an arduous process involving at least the following steps:

- Defining precincts, districts, and polling places

[2] T. Jim et al., 'Cyclone: A safe dialect of C'. In USENIX Annual Technical Conference. June 2002; George C. Necula et al., 'CCured: type-safe retrofitting of legacy software'. ACM Trans. Program. Lang. Syst. 27 2005, Nr. 3, ISSN 0164–0925.

- Assigning precincts and districts to polling places
- Defining candidates, contests, ballot questions, and referendums
- Configuring election options (e.g., candidate positions, votes per contest, etc.)
- Generating ballot style information
- Configuring the layout of paper ballots
- Generating "tabular" files for the paper ballot layouts
- Preparing election data for the M100 and M650
- Uploading election data to Compact-Flash cards and PEBs for the iVotronic

The provided documentation is wholly insufficient for election workers to complete the above requirements. During our ten week analysis of the ES&S voting system, we were unable to configure an election from start to finish (in the process experiencing unexplained and frequent software crashes) despite receiving several hours of vendor training and access to user manuals.

According to the office of the Ohio Secretary of State, a majority of counties in Ohio rely at least partially on ES&S technicians to generate their election definitions and tally election results,[3] practices we presume are due to the complexity of the configuration process and the lack of pertinent documentation. Such outsourcing introduces security risks, as the maintainers of the election delegate critical work to the voting machines' vendor. Subtle miscommunication between election officials and ES&S regarding the wide assortment of configuration options could lead to incorrect configurations and tallying. Since they will not necessarily cause election "Pre-Tests" or logic and accuracy tests to fail, incorrect configurations may be particularly difficult to detect.

Such a scenario appears to have transpired in the November 2007 elections in Ohio. The Mount Vernon news service has recently reported that a contract employee of ES&S double-counted more than 400 ballots by failing to "clear one report before running another report", resulting in incorrect election night results.[4] The mistake was noticed only when the report was regenerated (without repeating the mistake) several days later.

8.2 Improper Message Passing

In several instances, Unity makes use of the Windows filesystem to communicate between components. For example, Unity components pass sensitive unencrypted information to other modules by writing the data to disk. Such message passing techniques pose unnecessary security risks. Unauthorized users with access to the filesystem can gain access to election keys and configurations, bypassing Unity's authentication and authorization mechanisms (many Unity applications require a valid username and password). The ability to access sensitive files is increased when Windows filesharing is activated, a scenario apparently realized in some counties in Ohio.

[3] *Personal correspondence with Ohio Secretary of State Office employee*. November 28, 2007.

[4] *Election board verifies results*. Mount Vernon News, November 29 2007 ⟨URL: `http://www.mountvernonnews.com/local/07/11/29/elc.results.html`⟩.

Additionally, message passing via files creates dangerous *race conditions* in which an attacker who has write access to the filesystem can modify files after they are created by one Unity component but before they are processed by another. An attacker who can overwrite files can cause Unity to process arbitrary and incorrect election results. As before, this attack is worsened when Windows filesharing is employed.

Rather than use files to communicate data, good software engineering practice utilizes operating system features to more securely and reliably share information. Microsoft Windows provides such interprocess communication mechanisms in the form of COM objects, pipes, and sockets.

8.3 Static Code Analysis

It is customary for large and complex software systems (particularly those that operate in security-centric domains) to undergo rigorous internal quality assurance testing. As part of the QA process, static code analysis tools – software that searches the source code for bugs and potential security vulnerabilities – help software manufactures catch potential weaknesses before the product is released and used.

Our analysis of the quality of the source code indicates that an acceptable level of static code analysis was never performed. During our review, we found several notable programming errors, all of which were detected using readily available source code analysis tools and techniques:

- Using the Microsoft Visual Studio 2005 compiler (a more recent version of the compiler than that used by ES&S), we discovered that compilation failed due to several serious programming errors. For example, integers declared in "for loops" were used in out-of-scope contexts. This constitutes a violation of the ANSI C++ standard and results in compile time errors under most C++ compilers with which we are familiar. The compiler used by ES&S (Microsoft Visual Studio 6) optionally allows such unorthodox C++ code (presumably by inferring correct addresses for out-of-scope variables).

  Additionally, we discovered at least one instance when a C++ function call that takes no arguments was invoked without the required parentheses. Under most C++ compilers (including later versions of Visual Studio), such references to function names resolve to the address of the function and when invoked in a standalone fashion result in compile time errors. Oddly, such programming practices are accepted under older versions of Microsoft's compiler.

  Arguably, these programming practices do not constitute errors since they are accepted by the compiler. However, incorrect function invocation and the use of out-of-scope variables is a dangerous practice. In both cases, the C++ compiler must infer the programmer's intent (e.g., the scope of the variable or that the function reference should be treated as a function invocation). An incorrect conclusion could produce undesirable behavior. Additionally, if in the future the code is compiled using a different (and more standards-compliant) compiler (including more recent versions of Microsoft's C++ compiler), the best-case scenario results in compilation errors. At worst, the nonstandard programming practices could result in unstable and unpredictable behavior.

- The ES&S source code makes extensive use of memory-unsafe string operations. In particular, the *strcpy* and *sprintf* functions are frequently used throughout the various ES&S system components. These functions copy data into a buffer under the (unchecked) assumption that the target buffer is sufficiently large. If the size of the copied data exceeds the size of the buffer, the strcpy and sprintf functions write beyond the allocated space, potentially overflowing onto other data or process flow structures. Such a condition is known as a *buffer overflow* and represents a common (if not the most common) cause of software exploits and instability.

To prevent possible buffer overflows, memory-safe versions of *strcpy* and *sprintf* should instead be used. The *strncpy* and *snprintf* functions (note the additional *n*) take as a parameter the maximum number of bytes to write to the target buffer, eliminating the possibility of writing beyond the allocated space (assuming correct buffer sizes are provided).

Static source code analysis tools such as Fortify SCA[5] can be used to locate the use of memory-unsafe string operations in a system's source code. In addition, some compilers, including newer versions of Microsoft's Visual C++ compiler, produce compile-time warnings when the *strcpy* and *sprintf* functions are used.

- Using the Fortify SCA source code security analyzer, we were able to quickly identify several programming errors. Although such security analysis tools are imperfect, they serve as a useful means to quickly locate security vulnerabilities. (More thorough testing and analyses are required to find bugs missed by automated tools and to rule out false positives.) In our testing, Fortify SCA reported hundreds of "hot" (those deemed most likely to be correctly diagnosed and potentially exploitable) buffer overflows in Unity's source code. To a lesser extent, Fortify SCA also noted many other types of security problems, including string format vulnerabilities, integer overflows, and the use of non-null-terminated strings (in C and C++, a special character called the null character is used to denote the end of a string). Some of the reported errors are likely false positives (i.e., they refer to correct code) and many of the diagnosed problems cannot necessarily be exploited by an attacker to gain control of the system. However, the discovery of so many potential vulnerabilities implies that serious security and reliability problems likely do exist (a conclusion we repeatedly confirm; see Chapter 7) and, arguably equally troublesome, indicate that the vendor did not sufficiently validate their code.

[5] *Fortify Source Code Analysis (SCA).* ⟨URL: `http://www.fortifysoftware.com/products/sca/`⟩.

CHAPTER 9

ES&S EXAMPLE ATTACK SCENARIOS

This chapter reports on the "red-teaming" exercises performed to evaluate the ES&S Voting System. Prior to the study reported here, the ES&S system had not received a detailed source code and red-teaming security review. There were two studies, however, that should be mentioned. In December, 2006, a team, led by Florida State University's (FSU) Security and Assurance in Information Technology (SAIT) Laboratory, was commissioned to conduct a static software analysis on the iVotronic's version 8.0.1.2 firmware source code. The intent was to determine and identify flaws, vulnerabilities or anomalies, if any, that may have caused or contributed to the higher than expected under-vote rate in the District 13 Race between candidates Vern Buchanan and Christine Jennings.[1] The second study was carried out by two Ohio privacy activists James Moyer and Jim Cropcho. They demonstrated how the time-stamped paper trails produced by the iVotronics could be combined with a list of voters in the order they voted to determine which voter voted and in which way. Since Ohio law permits anyone to request and obtain these two documents, this is a clear violation of the secrecy of personal ballots.[2] We did not attempt to reproduce the findings of either of these groups.

To carry out the red team experiments, we crafted special tools that are designed to obtain information about an election and to circumvent the operations of an election. We also performed a number of physical attacks that enabled us to use the equipment in ways that allowed us to alter or thwart an election.

The first section of this chapter presents some of the special purpose tools that we developed. Next, the security seals and physical security issues are discussed. Finally, a number of explicit attack scenarios are presented.

9.1 Tools

There are several basic tools that we have built to make the process of developing attacks on the ES&S system more efficient and repeatable. These are a framework for delivering the malicious payload to the iVotronic system, a tool for reading and writing PEBs, a serial debugger for the iVotronic, a tool for reading and writing M100 PCMCIA memory cards, a JTAG hardware debugger, and a tool for extracting QNX files from the M100. These are discussed in the following sections.

[1] Alec Yasinsac et al., *Software Review and Security Analysis of the ES&S iVotronic 8.0.1.2 Voting Machine Firmware*. For the Florida, Department of State, February 23, 2007 ⟨URL: `http://election.dos.state.fl.us/pdf/FinalAudRepSAIT.pdf`⟩.

[2] Declan McCullagh, 'E-voting predicament: Not-so-secret ballots'. CNET News.com, 2007 ⟨URL: `http://www.news.com/2100-1014\_3-6203323.html`⟩.

Copies of these tools have been delivered along with the private part of this report.

9.1.1 A framework for delivering a malicious payload to iVotronic systems

We have developed a collection of routines that infect the iVotronic firmware and take control over key voting machine functionality. We refer to the collection as the "payload," because it is delivered by a Personalized Electronic Ballot (PEB) as part of an exploit. More explicitly, the payload is meant to be used by exploits in order to infect a voting machine when a PEB is inserted in the iVotronic. The payload is made up of four different parts: shellcode, flasher, linker, and hooks. These are discussed in the following sections.

Shellcode. The shellcode is a small (32 bytes) piece of code that is shipped with the exploit. The main purpose of the shellcode is to execute code stored on the PEB. This is achieved by loading one block of the PEB into RAM and then jumping to it.

Flasher. The flasher code, which is part of the malicious code on the PEB, is loaded into RAM by the shellcode. The task of the flasher is to first load the rest of the payload (including the linker) into RAM and then to copy this code to an unused area of the iVotronic's firmware flash memory. The code is copied to the flash memory so that it will be persistent. The final task of the flasher is to execute the linker.

Linker. The linker's task is to modify the original iVotronic firmware so that key function calls are intercepted and diverted to the "hooks." More specifically, the linker overwrites the target addresses of the function calls that are to be intercepted and replaces each with an address that points to the payload's hooks. The linker also updates the firmware's Cyclic Redundancy Check (CRC), so that it will pass the iVotronic validation tests.

Hooks. The hooks are a set of functions that are linked into the original firmware. Most hooks perform malicious activity like stealing votes or infecting PEBs inserted into the iVotronic with exploits.

An Example Malicious Payload Using the Framework. One version of the payload intercepts control just before the vote summary page is displayed. The payload steals votes at this point by changing them to select the attacker's candidate. The voter could notice that the ballot has been modified and try to correct the mistake by recasting his/her vote. If this is the case, the malicious firmware detects that the miscast vote has been discovered by intercepting the function that handles the pressing of the "back" button on this page. That is, if the vote changing is detected, the malicious firmware stops stealing votes for a period of time so that it is less likely that someone will discover that something is going on.

The framework could also be modified to use a Compact Flash card as the delivery mechanism, instead of a PEB.

9.1.2 Pebserial: a tool for reading and writing PEBs

Pebserial is a program we developed to read and write PEBs using the serial interface. Pebserial configures the serial interface in raw mode and implements the simple serial protocol used for communication between PEBs and ES&S voting machines (i.e., iVotronic and Unity). Pebserial has the following functionality:

- it retrieves the PEB's Election Qualification Code (EQC) and general election information, such as the PEB's type (e.g., supervisor, user), serial number, and version;
- it reads data blocks stored on the PEB;

- it writes blocks of data to the PEB.

By leveraging these basic operations, pebserial allows one to dump the contents of a PEB and to create PEBs with arbitrary contents.

Pebserial can correctly handle both unencrypted and encrypted PEBs. For encrypted PEBs (identified by a particular value in one of the fields of the EQC), the data is stored in encrypted format using the Blowfish algorithm. It is trivial to determine the encryption key, because the key is formed from fields of the EQC, which is always stored in the clear in the PEB. Therefore, encryption adds no additional security; anyone with access to a PEB can easily bypass its encryption and access its contents.

9.1.3 The iVotronic Serial Debugger

In order to facilitate the development of the iVotronic exploits, we created a specialized firmware with debugging support. The debugging firmware allows the exploit developer to attach a computer to the iVotronic via a serial cable connected to the iVotronic's serial port. The attached computer runs a debugger program (gdb) over the serial port. The attached debugger offers full debugging support of the iVotronic, including memory inspection and single stepping, and partial support for breakpoints.

The debugging firmware was created by attaching a debugging stub to the original firmware. The stub is provided as part of the gdb debugger. The stub can be concatenated to the original firmware and only a few pointers in the original firmware need to be changed in order to hook the stub into the firmware. No source code is needed in order to perform the modifications. Any attacker with access to a binary firmware image can create a firmware with debugging support.

9.1.4 A tool to read and write M100 PCMCIA memory cards

We developed a tool that contains python scripts to read and write PCMCIA cards and firmware update cards. The tool also sets all of the headers and CRC values appropriately. This enables us to put any malicious software or data that we want on the cards.

9.1.5 The JTAG hardware debugger

The iVotronic DRE is equipped with a JTAG connector on the circuit board. JTAG is a standard for low-level hardware debugging. We utilized this port in order to read and write directly to the flash chips without the help of the firmware. We needed this functionality mainly for two reasons. First, we wanted to validate that the firmware installed on the iVotronic was the firmware that had been placed in escrow and also that no other code was installed on the machine. Checking the version number displayed by the iVotronic would not suffice, since any malicious firmware could provide us with the correct version number. Second, we needed a way to recover from a situation where the firmware was corrupted in such a way that upgrading the firmware the usual way would not work. Since we were going to create a modified firmware for the voting machine, chances were that the machine could end up completely disabled if we made any mistakes when modifying the firmware.

In order to communicate with the onboard flash chips we modified OpenOCD,[3] which is an open source

[3] http://openocd.berlios.de/web/

JTAG tool created for ARM processors. Unfortunately, the processor used by the iVotronic is drastically different from the processors supported by OpenOCD; therefore, we had to develop support for this processor.

The M100 scanner utilizes the same processor as the iVotronic. This processor has JTAG support, but the M100 is not equipped with an easy to access JTAG port. In order to be able to read the firmware of the scanner and update it if necessary, we mounted a JTAG connector on the scanner. After dumping the content of the scanner's firmware we noticed that the firmware held in escrow was only part of the scanner's firmware. The second unknown part of the firmware was located at the top 64KB of the firmware flash. It appears to contain the QNX kernel, but we have no way to validate this.

We asked ES&S for source and binaries of this piece of the firmware, but we never received it. ES&S said that the firmware is the bootloader code that is part of the burned image on the hardware chip that is installed on the motherboard as part of the manufacturing process. They also said that it is write protected and cannot be changed or updated after installation on the motherboard.

We have no way of validating that the code located in the upper 64KB of the firmware flash contains the original QNX kernel. It is correct that the bootblock is write protected, but the protection is not fool proof. More specifically, we accidentally overwrote two bytes of this part of the firmware during our evaluation. The scanner stopped working at this point and we had to reprogram the bootblock using our earlier firmware dump.

9.1.6 A tool for extracting the QNX filesystem from the M100

The main part of the M100 firmware consists of a filesystem that is mounted as the root filesystem by the kernel. This filesystem contains a set of hardware drivers and startup scripts in addition to the main scanner application. The filesystem is a proprietary QNX filesystem specifically made for flash memory. Even though we had access to the latest QNX development environment, we were not able to extract the files contained in this filesystem. The filesystem contained in the firmware image was created with an old version of the QNX development tools, which is no longer supported by QNX.

Since we could not find any tools to access the filesystem or any description of the format of the filesystem, we had to reverse engineer the format. The content of the filesystem is compressed in order to save memory, which further complicated the reverse engineering.

We created a tool that given a firmware image will extract and decompress all the files contained in the filesystem. The tool is also able compress a file using the same compression as the filesystem, but there is no support for generating a filesystem from scratch.

We had access to the uncompressed versions of some of the files contained in the filesystem (all the COTS components and text configuration files). We validated the correctness of our dump utility by comparing the output files to the files we had access to. The file we were most interested in getting off the filesystem was the main scanner application, which we did not have a copy of. We were able to extract this application and disassemble it.

9.2 Physical Security and Security Seals

It is our understanding that in some counties in Ohio the doors on the front of the iVotronic, the Compact Flash slot in the iVotronic, and the PCMCIA card slot in the M100 are guarded by tamper-evident seals. Unfortunately, we were not given any samples of these seals; therefore, we could not determine how effective the seals are at preventing access to these components.

We also noted that none of the hardware devices contain factory installed tamper-evident seals. Neither the M100 nor the iVotronic contains factory installed tamper-evident seals. These devices are, however, shipped with a "warranty void" sticker on them, but this is just a regular sticker that can easily be removed and replaced without a trace. In addition, the sticker is not very solid and often breaks just by handling the equipment. Therefore, a broken sticker is not considered suspicious.

The lack of factory installed seals allows anyone with access to the equipment to open it and tamper with the inside. The counties could install seals on receipt of the equipment, but the equipment we received did not have any traces of county installed seals. We find it unlikely that the county officials would have removed all traces of the seals before shipping the equipment to us. It is more likely that the equipment never was sealed.

In addition to the seals, the M100 optical scanner has a number of locks that are supposed to keep intruders from getting at the scanner itself and at the PCMCIA cards. It was our experience that an inexperienced lock picker in our group was able to successfully pick the M100 lock in a matter of a few seconds, using two paper clips. After the lock was opened, the scanner could be opened by removing a few screws. The PCMCIA cards could then be removed and reinserted by removing the two small nuts and bolts that hold the protective covers over the cards. These covers are supposed to have seals on them, but the cover fixture can be removed and put back on without disturbing the seals.

The PEB communicates with the DRE and the PEB reader using an infrared link. The PEB contains a battery that is normally in the off state. In order to turn the PEB on, a magnet has to be in proximity of the bottom of the device. The PEB slot on the iVotronic contains a magnet and an infrared link for this purpose. When the iVotronic doors are sealed there is not enough space to insert a PEB in the iVotronic slot. We discovered, however, that by placing a strong magnet on the outside of the iVotronic doors when they are closed and by slipping the inside of a PEB, which is quite thin, behind the door and close to the PEB slot (but not in the slot) we were able to activate the PEB and load malicious firmware on the iVotronic. This was the same malicious firmware as is used for Scenario peb.1 in Section 9.3.1. If the doors are only sealed on the top, as we have seen in some literature, we could slip the whole PEB under the bottom of the doors and do the same thing, again using a strong magnet on the outside of the closed iVotronic door.

9.3 Successful Attack Scenarios

We implemented and tested all of the attack scenarios in this section.

9.3.1 Attack Scenario peb.1: Changing an Unattentive Voter's Vote

This scenario assumes an unattentive voter. A malicious PEB was crafted to use the PEB ballot header overflow vulnerability, discussed in Section 7.2.5, and the PEB was introduced into the system using one of the methods presented in that same section. The malicious code on the PEB contained the payload binary

as presented in Section 9.1.1.

Before the election, the PEB is inserted in an iVotronic machine. When the PEB is inserted, the exploit is automatically triggered, and, as a result, the malicious firmware is installed on the iVotronic. The malicious firmware behaves normally during the pre-election, but it starts to actively modify the election results as soon as the actual election starts. The malicious firmware uses the LAT detection vulnerability discussed in Section 7.2.18 to determine whether the iVotronic is in election mode.

The malicious firmware monitors the votes being cast and modifies the ballot to give advantage to a certain candidate. The firmware uses the payload "hooks" and links in just before the vote summary page is displayed. It steals votes at this point by assigning them to the other candidate. The modified vote shows up on both the screen and the Real-Time Audit Log (RTAL). If the voter actually checks the printed output of his/her votes and discovers that an error has been made, the malicious firmware detects that the voter recasts his/her vote, because the firmware is also hooked into the "back" button on this page, which is used to recast a vote. That is, if the voter detects the modified vote, the malicious firmware allows the voter's corrected vote to be recast and stops stealing votes for a period of time. In this way, it is less likely that someone will discover that something suspicious is going on.

It is worth noting that both the summary page and the paper trail report the modified selection, rather than the original one. From an attacker's perspective, it is better to keep the screen and paper consistent, because, if an abnormality is detected, then it is more likely to be attributed to a screen miscalibration rather than to an attack.

If one assumes that many voters do not check that their vote was properly cast, then this attack scenario can modify the results of an election, and it cannot be detected by a manual audit. The assumption that people do not check the paper trail is supported by at least two studies. In Everett's thesis[4] the author reports that over 60% of the voters she tested did not notice their votes had been changed in the review screen. It is reasonable to expect that a similar result would be obtained by changing the vote on the paper record. In another report, Selker and Cohen present a study in which errors were intentionally inserted into the VVPAT.[5] No voters reported the errors during voting, and only 8% agreed that there were errors in the VVPAT when asked.

9.3.2 Attack Scenario peb.2: Changing a Careful Voter's Vote

This scenario assumes that voters check their votes on both the screen and the printed ballot, but that they are not familiar with all of the details of how their votes are recorded on the paper audit tape. In this scenario, the iVotronic machine is compromised in the same way as described for the peb.1 scenario. However, in this case the voting process proceeds normally. That is, the voters actual choices are displayed and printed as cast by the voter.

After the voter has completed all of his/her votes, the voter has to cast and confirm his/her choices. Once the voter touches the "cast ballot" button and confirms the vote by touching the "confirm" button, the malicious firmware takes over. That is, the malicious firmware does not intercept the normal process until after the cast ballot and confirm pages have been presented to the voter.

At this point the malicious firmware changes the voter's electronic ballot, and the RTAL prints "Race S Canceled: Candidate X" and "Race S Selected: Candidate Y", where Y is the attacker's choice for Race S. The RTAL then immediately prints "Accepted," the other normal tabulation information, the bar code, and

[4] S. Everett, *The Usability of Electronic Voting Machines and How Votes Can Be Changed Without Detection*. Ph. D thesis, Rice University, 2007.

[5] T. Selker and S. Cohen, *An active approach to voting verification*. May 2005 (28). – Technical report.

the time stamp. This whole printing process plus the paper scrolling up and out of site takes less than 4 seconds. The printer is also scrolling forward and back when printing this information, which makes any attempt to read what is printed even more difficult.

After carefully checking, casting, and confirming their votes, most voters do not pay attention to what is printed on the RTAL.[6] Also, unless the voter is watching closely as the RTAL is printing, he/she will not be able to see what was printed.

This version of the attack is more likely to evade detection, since the stealing happens only after the voter has confirmed his/her selection. By doing this, the modified selection is never shown on the screen and is printed on the audit log only when the paper is scrolling up, immediately before the barcode is printed.

9.3.3 Attack Scenario peb.3: Canceling the Vote of a Fleeing Voter

In this scenario, the iVotronic machine is compromised in the same way as described for the peb.1 scenario. However, this time the malicious firmware takes advantage of "fleeing" voters. These are voters that leave the voting station before having completed their voting, which is not uncommon. In Ohio the votes of fleeing voters are discarded.[7]

When a voter flees, the iVotronic makes a chirping sound after about twenty seconds, which alerts a poll worker. The poll worker has to insert a supervisor PEB in the iVotronic and follow a specific procedure to discard the ballot. For privacy reasons, the poll worker has no access to the content of the ballot.

For this scenario, the malicious firmware intercepts the call to the routine that enables the chirping sound, indicating a fleeing voter. The malicious firmware behaves differently depending on whether or not the fleeing voter cast a vote for the attacker's candidate.

Case 1: Voter voted against the attacker's candidate. If the fleeing voter did not vote for the attacker's candidate, then the malicious firmware does nothing and lets the chirping program perform as it should. In this case, the fleeing voter's ballot will be discarded, and there will be one less vote for the undesired candidate.

Case 2: Voter voted for the attacker's candidate. If the fleeing voter voted for the candidate that the attacker wants to win, then the malicious firmware completes the voting process, by faking the pressing of the "VOTE" button. This results in another vote being cast for the attacker's candidate.

This could result in a lower than average number of fleeing voters. Also, it might be possible to detect that all of the fleeing voters had voted against the attacker's candidate. Since this could possibly arouse suspicion, the firmware only completes the voting process for a certain percentage of the fleeing voter ballots that are for the attacker's candidate.

9.3.4 Attack Scenario peb.4: Canceling a Vote by Faking a Fleeing Voter

In this scenario, the iVotronic machine is compromised in the same way as described for the peb.1 scenario. However, in this case the firmware fakes a fleeing voter. If a voter does not select the candidate that the attacker wants, the malicious firmware intercepts the confirmation page's confirm function and pretends to cast the ballot: the normal "thank you" page is displayed, but nothing is printed on the audit tape. After

[6] Everett (as in n. 4); Selker and Cohen (as in n. 5).

[7] This differs from California, where a fleeing voter's ballot is cast by the poll worker.

waiting a few seconds (during which time the voter likely leaves the booth) the firmware again displays the confirmation page. After some time, the firmware calls the fleeing voter code and the machine will start chirping. A poll worker will think the voter was a fleeing voter, and, in accordance with Ohio's procedures, the ballot will be canceled.

9.3.5 Attack Scenario flash.1: iVotronic Denial-of-Service

This scenario causes an iVotronic to crash when a malicious flash card is inserted and the iVotronic attempts to read an image file from the card. A malicious flash card was crafted to take advantage of the flash card hot spot buffer overflow vulnerability, discussed in Section 7.2.6, and the flash card was introduced into the system using one of the methods presented in that same section.

In Section 7.2.6 we saw that there are two different code patterns that are used in the iVotronic to read image headers. For this attack, code that uses the first pattern for reading the image header is exploited. Recall that executing this code allows an attacker to overwrite the stack return address, but not with data under the attacker's control; therefore, a system crash is likely.

A denial-of-service attack was implemented by replacing one of the system's election image files with a file that overflows the hotspot handling function that uses the first code pattern. Because the attacker cannot control what gets written on the stack, he/she cannot introduce malicious code. However, triggering the image overflow vulnerability can cause the iVotronic to crash.

When the modified image was to be displayed, the iVotronic crashed, resulting in the expected denial-of-service.

9.3.6 Attack Scenario flash.2: Voter Confusion

In this scenario, the iVotronic machine is compromised in the same way as described for the flash.1 scenario. That is, a malicious flash card was crafted to take advantage of the flash card hot spot buffer overflow vulnerability, discussed in Section 7.2.6, and the flash card was introduced into the system using one of the methods presented in that same section. However, for this attack code that uses the second code pattern described in Section 7.2.6 is exploited. Recall that when executing this code an attacker can overwrite the stack return address with an address pointing to code of the attacker's choosing. Therefore, the possibilities for the attacker have no limitations.

To demonstrate how the election process could be subverted, one of the system's election image files was replaced with a file that overflows an image hotspot handling function, exploiting the second code pattern. The exploit crafted on the image file triggered the image overflow vulnerability and introduced code that displays an obviously wrong message on the screen to confuse the voter. Although displaying this message was basically benign, it demonstrates that one could introduce code of their choice and could subvert an election. For instance, exploits peb.1, peb.2, peb.3, and peb.4 could all be realized using this image overflow vulnerability. The attacker need only change the code introduced.

Recall that in Section7.2.6 there were ten different locations in the iVotronic's code that read image headers using the second of the two code patterns. Although the attack presented in this section was for only one of the ten occurrences of the second pattern, the other instances could be used for delivering malicious code in the same way.

9.3.7 Attack Scenario unity.1: Unrestricted Access to Unity Ballot Preparation Software

The Unity election management system is used to prepare ballots for all of the iVotronics and M100s in all of the precincts. Unfortunately, as shown in Section 7.1.9, the authentication process for the Election Data Manager (EDM), Ballot Image Manager (ESSIM), and Audit Manager components of the Unity election management system can be bypassed with a simple SQL injection. This gives the attacker unrestricted access to these components.

This scenario demonstrates an SQL injection exploit that uses a secret constant string[8] as the username parameter, and the password parameter is left blank.

We tried this for EDM, ESSIM, and for the Audit Manager. In all cases the user was logged on with no further authentication checks.

9.3.8 Attack Scenario unity.2: Compromising the Unity Election Reporting Manager

The Unity election management system is used to compile results from all of the precincts. The most common method of delivering the results is on a results PEB. Unfortunately, the Election Reporting Manager (ERM) component of Unity, which is used to compile the results, has a buffer overflow in the code that reads the results PEB.

For this scenario, a malicious PEB is crafted to take advantage of the ERM PEB reader buffer overflow, which was presented in Section 7.1.1. The malicious PEB is introduced into the system using one of the methods presented in Section 7.2.5. The malicious code on the PEB creates an account "attacker" with password "weownyou."

When the PEB is read by the ERM using the PEB reader, the data on the PEB overflows the stack of the ERM application. The return address of the current function is overwritten to point to malicious code, which is then executed. The malicious code adds the "attacker" account and control is returned to the application, which, after a few seconds delay, displays a message indicating that reading the PEB has failed. This is expected, since the malicious code in its current form does not take care to exit cleanly after it is executed.

The success of the attack is verified by logging in to the system as user "attacker" with password "weownyou." The login was successful.

9.3.9 Attack Scenario m100.1: Changing the Firmware on the M100 Scanner

The M100 optical scanner is used to count physical election ballots. The firmware on the M100 is updated by inserting a special firmware PCMCIA card. These are usually provided by the ES&S service representatives.

The development of a malicious firmware was complicated by the lack of tools required to both compile the M100 sources and build the final image. For this scenario the QNX filesystem extraction tool and the PCMCIA read/write tool described in Sections 9.1.4 and 9.1.6, respectively, were used to craft a malicious PCMCIA update card.

The malicious firmware that we developed behaves exactly like the original firmware, except that it gives all the votes to the first candidate. To accomplish this, we had to extract the binaries from the firmware installed on the M100 and patch them to steal votes.

[8] See Section 22.9.3.7 in the Annex of the private report for the actual string.

When the card is inserted in the M100 and it is powered up the card will be accepted and the CRC check will be passed, because the card was crafted with an appropriate CRC value and in the appropriate format. Next, the M100 will display a menu asking the user if he/she wants to replace the firmware. A "yes" response results in the firmware being updated.

9.3.10 Attack Scenario m100.2: M100 Denial-of-Service

The M100 optical scanner, which is used to count physical election ballots, uses ballot PCMCIA cards to get ballot information from Unity.

This scenario uses the PCMCIA read/write tool described in Section 9.1.4 to craft a malicious PCMCIA ballot card, which forces the M100 to write to an invalid address. This results in a segmentation fault. The kernel intercepts the segmentation fault signal, prints an error on the printer, and freezes the M100, asking the user to reboot.

Our effort with this vulnerability was hampered by the fact that we did not have all of the source code for the M100 firmware, even though we requested the missing firmware source numerous times. As a result, we wasted a lot of time and energy reverse-engineering the firmware. The result is that we did not have enough time to find a way to control what is written. Therefore, this scenario results in a system crash.

9.3.11 Attack Scenario virus.1: Compromising Entire Election Process with a Virus

The ES&S voting process forms a loop. That is, the Unity election management system is used to initialize Personalized Electronic Ballots (PEBs), Compact Flash cards, and PCMCIA cards. These are in turn used to initialize the iVotronic DREs and the M100 optical scanners with ballot information. The PEBs are also used to activate the DREs for voting, power the machines on, activate supervisor functions, and to transport the election results from the iVotronics to Unity. Figure 9.1 shows this flow of information. Inserting malicious code at any step in this process could result in a virus spreading to all of the other components, completely compromising the election.

This exploit uses the payload framework presented in Section 9.1.1 to implement the propagation of malicious code from an infected iVotronic DRE to another iVotronic DRE, where "infected" refers to a running malicious firmware. Similarly, the payload also implements spreading from an infected iVotronic DRE to the Unity election management system, such that subsequent PEBs generated from an infected Unity installation will propagate this payload to all iVotronic DREs in its jurisdiction. In this manner, a persistent viral infection of malicious code in an ES&S electronic voting infrastructure has been implemented.

In the case of spreading from iVotronic to iVotronic, a PEB is infected when it is inserted into an infected DRE to activate it for voting. This allows a malicious firmware to infect a master PEB used for pre-election logic and accuracy tests. Then, on election day, the master PEB can spread the infection to all machines in a polling location as they are activated.

Spreading from iVotronic to Unity is accomplished in a similar manner, except that a PEB is infected when inserted to collect votes as the terminal is closed. The PEB used for this function will subsequently be loaded into Unity. Taking advantage of scenario unity.2 in Section 9.3.8 it can be used as a vector for spreading the virus to election central. That is, when the PEB is loaded into Unity, a program will be installed to infect all future PEBs generated from that Unity installation with the malicious code to implement the previous component of this virus.

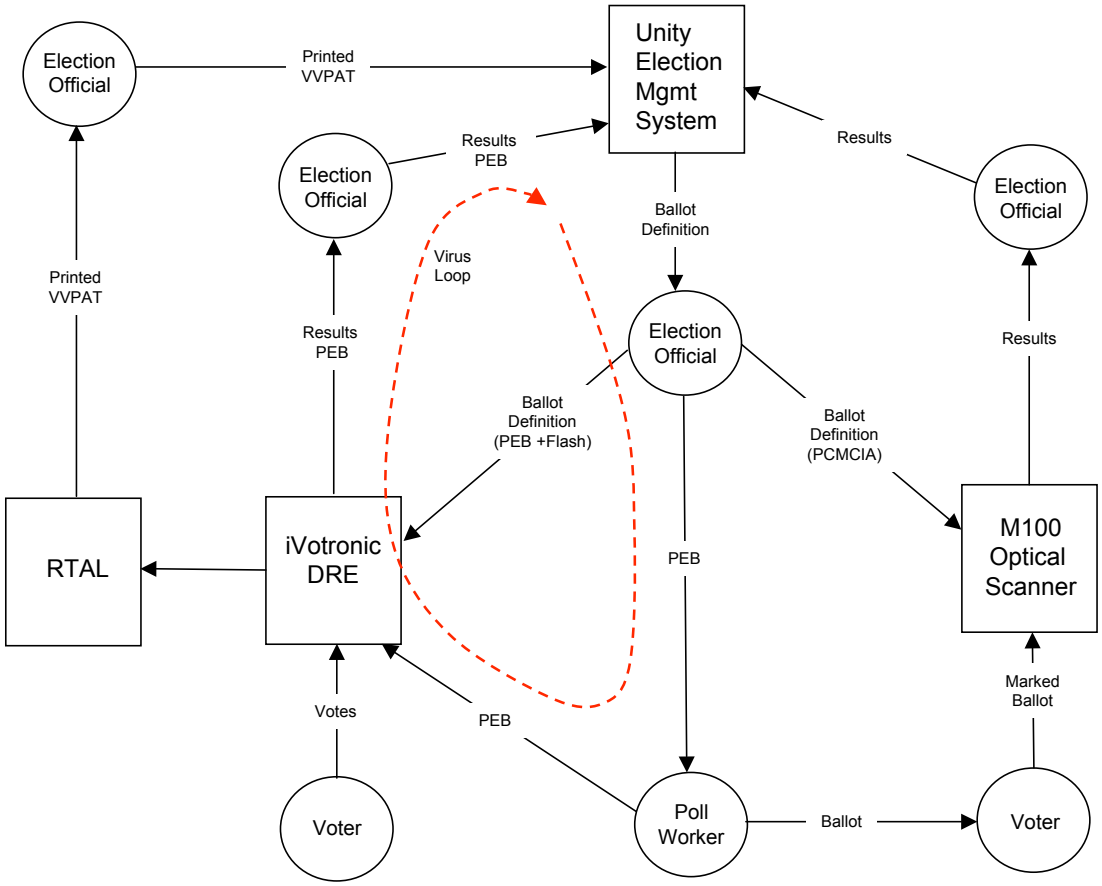

Figure 9.1: ES&S Architecture and Virus Loop

Scenario unity.2 in Section 9.3.8 could also be used during the initial testing to modify the Unity code. Again, the modified code puts a virus on all of the PEBs that are distributed to all of the precincts. These PEBs, in turn, spread the virus to all of the iVotronic DREs in all of the precincts. The result is that every iVotronic now has the malicious code that compromises the election.

9.4 Potential Attack Scenarios

9.4.1 Attack Scenario flash.3: iVotronic Exploits Using a Flash Card as Delivery Mechanism

Given more time, we could modify the payload framework discussed in Section 9.1.1 to work with the Compact Flash code vulnerability discussed in Section 7.2.6 to deliver a payload similar to those discussed in Section 7.2.5, which used a PEB as the delivery mechanism. This would give us an alternate path to introduce malicious code into the system if PEBs or PEB-like clones were not available.

Part III

Analysis of the Premier Elections Solutions, Inc. Voting Systems

CHAPTER 10

PREMIER EXECUTIVE SUMMARY

The study included in this part of the EVEREST report evaluates the ability of the Premier voting system to guarantee a trustworthy election. The review team was provided access to the Premier source code and election equipment. The reviewers studied these materials in order to identify any security issues that can be exploited to affect an election. As part of that analysis, the reviewers were asked to identify best practices that may limit or neutralize the impact of discovered issues.

Our analysis suggests that the Premier system lacks the technical protections necessary to guarantee a trustworthy election under operational conditions. Flaws in the system's design, development, and processes lead to a broad spectrum of issues that undermine the voting system's security and reliability. The resulting vulnerabilities are exploitable by an attacker, often easily so, under election conditions. These vulnerabilities are the result of the following failures of the Premier system's design or implementation:

- *Failure to effectively protect vote integrity and privacy* - Numerous vulnerabilities allow an attacker to modify or replace ballot definitions, to change, miscount, or discard completed votes, or to corrupt the tally processes. Further issues expose voter choices and can lead to voter coercion and vote selling.
- *Failure to protect election from malicious insiders* - The Premier system does not provide adequate protections to ensure election officials, poll workers, or vendor representatives do not manipulate the system or its data. These attacks are often invisible after the fact, and therefore misuse is difficult or impossible to uncover later.
- *Failure to validate and protect software* - The Premier system makes only limited and often ineffective attempts to validate the software running within system. Thus, an attacker may exploit software and replace it with their own with little fear of detection. Further, the recommended means of installing and upgrading software is frequently highly dangerous.
- *Failure to provide trustworthy auditing* - The auditing capabilities of the Premier system are limited. Those features that are provided are vulnerable to a broad range of attacks that can corrupt or erase logs of election activities. This severely limits the ability of election officials to detect and diagnose attacks. Moreover, because the auditing features are generally unreliable, recovery from an attack may in practice be enormously difficult or impossible.
- *Failure to follow standard software and security engineering practices* - A root cause of the security and reliability issues present in the system is the visible lack of sound software and security engineering practices. Examples of poor or unsafe coding practices, unclear or undefined security goals, technology misuse, and poor maintenance are pervasive. This general lack of quality leads to a buggy, unstable, and exploitable system.

> We found the Premier software to be unstable. Frequent crashes, system lock-ups, and unexplained errors were commonplace in our experiments. Stability problems were acute in the GEMS server, where failures occurred during normal use and under limited loads.

Our findings are consistent with those of previous studies. When taken as a whole, this and previous studies highlight a central point of concern: there is a demonstrative lack of improvement in the security of elections conducted using the Premier system. Initial reviews of the Premier system were undertaken as early as 2001. After six years of reviews and many new software and hardware upgrades, reviewers not only continue to find the same and similar problems as reported earlier, but continue to uncover new serious issues. Thus, the only reasonable conclusion that one can draw is the engineering approaches undertaken by Premier to eliminate previous problems and avoid new ones are failing.

The flaws in the Premier system place the security of an election almost entirely on physical procedures. Our analysis suggests that when those practices are not uniformly followed, it will be difficult to know when attacks occur. Even when the attacks are identified, it is unlikely that the resulting damage can be easily contained and the public's belief in the accuracy and fairness of the election restored.

The review team feels strongly that the continued issues of security and quality are the result of deep systemic flaws. Thus, we agree with previous analyses and observe that the safest avenue to trustworthy elections is to reengineer the Premier system to be secure by design.

CHAPTER 11

PREMIER STUDY OVERVIEW

11.1 Part Structure

The following chapters detail the Premier portion of the EVEREST review. Readers are directed to Part 1 of this report for a discussion of the review's goals and methodologies. Those readers not experienced in information security or Ohio election practices are also encouraged to read the threat model in Chapter 3. The content in that chapter is instrumental in gaining a detailed understanding of the substance and impact of the issues identified throughout.

The issues and commentary described in this section of the report are *reflective of our analysis of the Premier system only*. None of the included material should be considered to apply to either the ES&S or Hart systems, nor should any statement be construed to be making any quantitative or qualitative comparisons between the different systems. Such comparisons are expressly outside the scope of the review, and we have not developed any opinions on the relative merits of any one system with respect to the others. Nothing in this review should be considered as a positive or negative commentary on electronic voting in general.

The remainder of this part of this report is structured as follows. We begin in the next section by giving a brief overview of the Premier system and its use in Ohio elections. Chapter 12 broadly characterizes the security and reliability of the Premier system by highlighting several architectural and systemic issues encountered in the review. Chapter 13 provides detailed technical information on the confirmation of previously reported issues and Chapter 14 provides similar detail on new issues uncovered by our analysis. Chapter 15 describes a number of attack scenarios that demonstrate how the identified issues may be used in concert to subvert or otherwise indict an election.

11.2 Architecture

The purpose of this section is to familiarize the reader with both the architecture of the Diebold/Premier Electronic Voting Systems. The contents of this report have been taken from a variety of public documents and personal experience, and are not intended to be an exhaustive or authoritative description of the system.

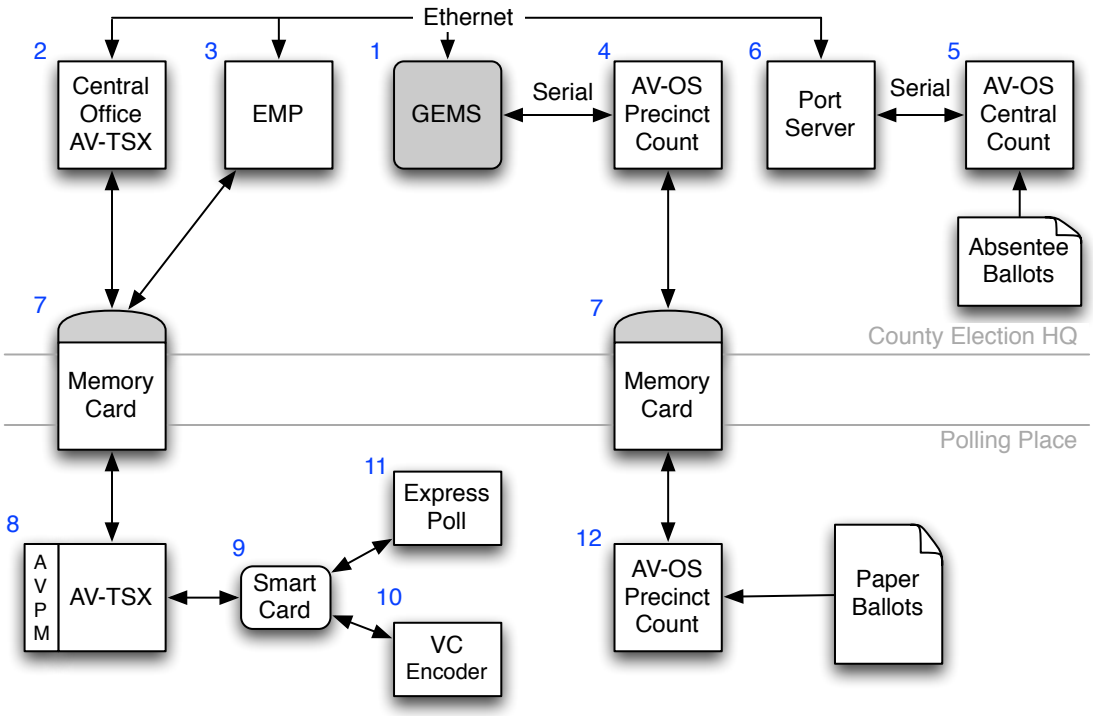

Figure 11.1: The major components and their interconnections for counties using Premier Electronic Voting Systems. Unless otherwise indicated, arrows depict physical transport of cards and ballots.

11.2.1 Components at County Election Headquarters

A number of voting components reside at the county election headquarters. Refer to Figure 11.1 for a pictorial representation of component interaction.

1. **Global Election Management System (GEMS):** The GEMS server is responsible for running back-end election processes. Accordingly, it is stored at the county's election headquarters. Election administrators use GEMS to create ballot definitions, program memory cards and tally all votes when an election closes. It is connected to an AV-OS Precinct Count via serial cable and either the Election Media Processor (EMP) or the Central Office AV-TSX in the county election headquarters by either a dedicated or office-wide LAN.

   The GEMS server runs on a server-class machine running Windows Server 2000. The Premier software typically running on this physical server includes:

   - **GEMS:** Software used to define, tally, and report an election.
   - **KeyCardTool:** Software used to create administrative smart cards, including the *Security Key Card*, the *Central Administrator Card*, and the *Supervisor Card*.

   To harden the GEMS server against previously reported vulnerabilities, Ohio GEMS servers additionally include third party security software: *Verdasys Digital Guardian*, *Sygate Security Agent* network firewall, and *McAfee VirusScan*.

2. **Central Office AccuVote Touchscreen (AV-TSX):** The Central Office AV-TSX is a touchscreen voting device used to write ballot definitions and software updates to memory cards before an election and read election results from memory cards after an election. Any AV-TSX can be used to support these tasks. However, counties deploying many memory cards may optionally use the EMP.

   The Central Office AV-TSX runs on the AV-TSX hardware with the addition of an Ethernet card that plugs into a memory card slot. The central office mode of operation is selected by an administrator. The device runs Microsoft Windows CE and includes the following Premier software:

   - **BallotStation:** Software through which election officials create memory cards (same software as voters make their choices).
   - **Bootloader:** Software used to load the operating system on the AV-TSX hardware.

3. **Election Media Processor (EMP):** Because efficiently encoding memory cards before and reading election results after an election is difficult using Central Office AV-TSXs (which can only read a single PCMCIA card at a time), Premier offers the EMP server. This device can read as many as six memory cards in parallel, significantly speeding up the above operations.

   The EMP server can run on either a Windows 2000 or Windows XP machine. Premier software running on this physical server includes:

   - **EMP:** Software used to communicate with GEMS and interface with the memory cards.

4. **AccuVote Optical Scan (AV-OS) Precinct Count:** The AV-OS Precinct Count (AV-OS PC for short) machines use an older style memory card incompatible with those used by the AV-TSX and EMP. Therefore, the central office contains a dedicated AV-OS PC in order to write ballot definitions to memory cards before an election and read election results after an election. The central office AV-OS PC software and hardware are identical to the units used at the polling place.

   The AV-OS PC does not have a dedicated operating system. The Premier software running on this physical machine includes:

   - **AV-OS PC:** Software used to operate the AV-OS hardware, read ballots, and tally results.

5. **AccuVote Optical Scan (AV-OS) Central Count:** The AV-OS Central Count (AV-OS CC for short) is an optical voting scanner used read and record absentee ballots. The AV-OS CC uses identical hardware to the AV-OS PC; however, it runs different software. The only difference between the AV-OS PC and AV-OS CC is the installed memory chips. The AV-OS CC communicates with the GEMS server throughout its operation using the Port Server, which changes the serial connection to an Ethernet connection.

   The AV-OS CC does not have a dedicated operating system. The Premier software running on this physical machine includes:

   - **AV-OS CC:** Software used to operate the AV-OS hardware, read ballots, and communicate with the GEMS server.

6. **Port Server**: The Port Server connects machines using serial port communication to Ethernet networks. Premier uses the Digi PortServer II at the county election headquarters, which allows them to connect up to 16 AV-OS CC machines to the GEMS server. AV-TSX units located at the county election headquarters use an Ethernet card plugged into a memory card slot and therefore do not need to use the Port Server to communicate with GEMS.

7. **Memory Cards**: Premier Election Systems rely on memory cards as the major avenue of communication between the back-end servers and the polling place. Before an election, for instance, an AV-TSX will read the card to find the ballot definition, sound files, translations into other languages (.edb and .xtr files), interpreted code (.abo files) used to print reports and other configuration information. Votes are also stored on memory cards (.brs files), which are returned to the county election headquarters for tabulation at the close of an election.

   Premier systems rely on two types of memory cards. As described above, all AV-TSX units use standard 128 MB PCMCIA memory cards. Such cards are readily purchased at any electronics retailer, and all files can be read by any PC with a PCMCIA card reader, commonly found on laptops. The AV-OS units rely on a 128 KB EPSON 40-pin memory card. Manufacturing of this card ceased in 1998. Because two different technologies are used, the AV-TSX memory cards cannot be used in an AV-OS machine and visa-versa.

A Premier election system may also include other components at the county election headquarters. However, for various reasons, the following components are not used in Ohio. We list them for completeness.

- **AccuFeed:** The AccuFeed unit is used in the county election headquarters to assist with the tallying of paper ballots. Specifically, this device helps feed a large number of paper ballots to the AV-OS Central Count unit. This unit is not used in Ohio and is therefore not examined in this study.
- **Modem Connections:** Communication between voting machines deployed in precincts and the servers in the county's election headquarters is possible using modems. While this practice is strictly forbidden by law in Ohio, all voting machines contain the necessary software and fully enabled hardware to perform such operations. In some precincts, use of these devices is inhibited using a plastic or screw-attached metal cover.

11.2.2 Components at Polling Place

The remaining voting components reside at the polling place. Refer to Figure 11.1 for a pictorial representation of component interaction.

8. **AccuVote Touchscreen (AV-TSX):** As previously mentioned, the Precinct Count AV-TSX is exactly the same as the Central Office unit (i.e., any unit can be used for either mode). These units are deployed in each polling place and allow for touchscreen voting to occur. All AV-TSX machines used in Ohio also include an AccuVote Printer Module (AVPM), also known as a Voter Verifiable Paper Audit Trail (VVPAT) printer unit, which creates a physical copy of a cast ballot on thermal paper.

   The AV-TSX runs Microsoft Windows CE and includes the following Premier software:

   - **BallotStation:** Software through which voters make their choices.
   - **Bootloader:** Software used to load the operating system on the AV-TSX hardware.

9. **Smart Card**: The AV-TSX determines what a user can do based on the inserted smart card. A smart card is a credit card like device that contains a small chip capable of performing cryptographic operations. The smart cards come in four varieties: *Voter Access Card*, *Supervisor card*, *Central Administrator card*, and *Security Key Cards*. When one of these cards is inserted into an AV-TSX, the person using this card is allowed to access an associated set of operations on the machine. For instance, a user entering a Voter Access Card would not be allowed to perform administrative operations.

10. **Voter Card Encoder (VCE):** The VCE is a standalone, calculator-like device used to make *Voter Access Card* smart cards valid for casting one ballot. The VCE is used by polling places that use traditional paper voter log books to identify legitimate registered voters.

    The VCE includes the following Premier software:

    - **VCEncoder:** Software used to read and write Voter Access Cards.

11. **ExpressPoll:** The ExpressPoll tablet devices functions as an electronic replacement for the traditional voter log book. As users enter the polling place, a poll worker can confirm their eligibility and issue voter smart cards.

    The ExpressPoll runs Windows CE and includes the following Premier software:

    - **ExpressPoll:** Software that allows a poll worker to find registered voters in a electronic database.
    - **CardWriter:** Software that allows the Express Poll to write to inserted smart cards.

12. **AccuVote Optical Scan (AV-OS) Precinct Count:** The AV-OS Precinct Count (AV-OS PC for short) voting machine is primarily used in the polling place (the AV-OS PC described above exists solely to create memory cards to be used at the polling place). Voters filling out paper ballots in their precinct use this device to scan and tabulate their results. After scanning a ballot, The paper copy of the ballot is dropped into a locked plastic bin in case an audit is required.

    The AV-OS PC does not have a dedicated operating system. The Premier software running on this physical machine includes:

    - **AV-OS PC:** Software used to operate the AV-OS hardware, read ballots, and tally results.

11.3 Election Procedures

Given the election headquarters and polling place components described in Sections 11.2.1 and 11.2.2, we now discuss how an election proceeds. Note that these procedures vary from county to county.

11.3.1 Pre-Election Procedures

- Before an election, the election administrators define the ballot. These definitions are created on the GEMS server.
- When ready, the GEMS server contacts the AV-OS PC, AV-TSX, and optionally EMP machines residing in the central office via the appropriate serial cable and network interfaces. These devices then receive the ballot definitions.
- The central office AV-OS, AV-TSX, and EMP machines encode the ballots onto memory cards. One memory card must be encoded per machine in the county. Memory cards encoded for touchscreen voting machines are not physically compatible with those encoded for optical scanning machines.
- Memory cards are then sent to each polling site. The details of how they get there, however, vary. Administrators can insert the cards into the appropriate machines before they are delivered to the polling place. Alternatively, cards can be delivered separately and inserted by poll workers.

11.3.2 Polling Place Election Procedures

The polling place election procedures for the AV-TSX proceed as follows.

- An election begins when a polling place administrator inserts a supervisor smart card. This card allows the administrator to open and close elections and also view machine logs.
- When voters arrive at the polling place, poll workers ensure they are legitimate registered voters using either the traditional paper voter log or the ExpressPoll.
- After voters are authenticated, they receive a *Voter Access Card* smart card, which allows them to cast a single ballot. With this card in hand, the voter approaches the AV-TSX.
- Upon reaching the AV-TSX, the voter inserts the *Voter Access Card* into the machine and follows the on screen instructions to cast a ballot. Before the ballot is officially cast, the voter is asked to confirm the available print-out is correct.
- After the ballot has been recorded by the machine (and stored on the memory card), the AV-TSX reprograms the smart card so that it can not be used until re-encoded by either the VCE or ExpressPoll.
- At the close of election, the polling place administrator closes the election by again inserting the supervisor smart card and selecting the appropriate option.

The polling place election procedures for the AV-OS Precinct Count proceed as follows.

- AV-OS machines are configured by inserting one of the above mentioned memory cards. Machines then run the Zero Report to demonstrate that they do not currently contain votes.
- Voters fill out ovals on the paper ballots at the polling place. When finished, ballots are inserted into the AV-OS PC. A paper feeder similar to a fax machine is used to take the ballot for optical scanning. Ballots are then physically stored within the AV-OS until the election concludes.
- Users are notified if their ballots are incorrectly filled out. For instance, if a voter selects two candidates in a race where only one candidate can be chosen, the ballot will be returned and the voter will be instructed to have the ballot invalidated/voided. The voter can then receive a new paper ballot.
- At the close of election, the polling place administrator unlocks the top panel of the ballot box and holds down the two buttons while feeding the scanner a special ender card. This action closes the election and causes the AV-OS PC to print an election summary.

11.3.3 Post-Election Procedures

- Votes can be sent to the County Election Headquarters in one of two ways. One option is to remove the memory cards and physically transport them to the central office. Alternatively, the entire voting machine can be transported. Note that votes can also be reported via modem; however, Ohio law prohibits the use of modems with election equipment.

- Once memory cards arrive at the central office, the AV-TSX memory cards are inserted into either the central office AV-TSX or the EMP, and the AV-OS PC memory cards are inserted into the central office AV-OS PC. Note that the memory cards are not physically compatible, so an AV-OS machine can not be used to tally votes from an AV-TSX. The memory card contents are sent to the GEMS server for tabulation.

- Once all memory cards are reported to the GEMS server, an official election results summary is printed.

11.4 Verdasys Digital Guardian

Due to the results of previous studies of the Diebold/Premier elections equipment, the state of Ohio has requested that Premier include additional third party security software to harden the GEMS server. Specifically, the GEMS server setup in Ohio includes: *Verdasys Digital Guardian*, *Sygate Security Agent* network firewall, and *McAfee VirusScan*. The latter two security tools provide standard system protection and warrant little discussion. However, Digital Guardian is presented as a remedy to a number of significant GEMS vulnerabilities such as the ability for an attacker to perform arbitrary modification of an election database simply by having access to the GEMS server filesystem (see Issue 13.1.2). We were unable to find significant technical specifications of Digital Guardian from public resources. Therefore, we performed penetration testing of the Digital Guardian protected GEMS server. Note that because Digital Guardian is considered COTS software, Premier was not required to provide any source code, nor were we provided any technical documentation describing how the system works. However, we were provided the current policy specifications and some notes from a Premier technician, which greatly aided our understanding of how Digital Guardian protects a system.

Digital Guardian was designed to protect a system running Windows 2000 or XP. It allows an administrator external from the local system to specify policies that control how all local users are allowed to execute programs and access files. In Ohio's setup, a state employee possesses a special laptop called the Digital Guardian console. Each GEMS server contains the Digital Guardian Agent enforces the policy specified by the console. The only way the Digital Guardian Agent can be disabled is if a state employee directly connects the Digital Guardian console the GEMS server and specifies that the agent should be disabled.

The Digital Guardian Agent running on all GEMS servers enforces two high level policies (keep in mind that in Ohio, all county GEMS servers are administered by Premier employees). First, the election databases should only be accessed by the GEMS program. Second, the vendor employee should not have access to any GEMS data. The remainder of the policy installed in each Digital Guardian Agent exists purely to retain the system integrity and keep an attacker from circumventing Digital Guardian.

In order to provide separation between users, three Windows users have been created: *Administrator*, *GEMSAdmin*, and *GEMSUser*. The *Administrator* account performs basic administration and maintenance of the GEMS server, but operations that involve GEMS data are forbidden. The *GEMSAdmin* account is not a system administrator, rather, it is the only user allowed to perform file manipulation operations, e.g., copy, move, delete, on the election database files. Finally, the *GEMSUser* account may only modify election database files using the GEMS program, and it should not be able to delete, copy, or paste the files. Additionally, *GEMSUser* is allowed to burn backups of the election database, as this is a necessity on election day.

Our study investigates Digital Guardian's ability to enforce the high level protection policies. In doing so,

we concentrated on discovering ways to disable Digital Guardian. In addition, we considered ways in which users could perform actions explicitly denied by the policy. Finally, our analysis only considers the GEMS server. While we were provided a Digital Guardian console laptop, we were unable to analyze network communication due to lack of time. We recommend future investigations thoroughly study ways in which an attacker can remotely compromise or bypass Digital Guardian.

CHAPTER 12

PREMIER SYSTEMIC AND ARCHITECTURAL ISSUES

This report examines the Premier system's resistance to manipulation of electoral outcomes, procedures and public perception. We have found substantial flaws that seriously inhibit the ability of the Premier system to meet these goals. Many of the issues presented in the following chapters relate to critical functional or procedural problems embodied in the Premier design and implementation. Such problems undermine the security of the system and provide practical avenues to compromise critical election data and systems. This chapter broadly characterizes[1] the report's findings by highlighting the five areas of most concern/interest:

- **Ineffective Data and Privacy Security** - The methods used to protect the integrity and privacy of important election data are circumventable, often trivially.
- **Ineffective Cryptographic and Hardware Security** - The use of many standard security technologies is deeply flawed.
- **Unsafe Software Management** - The means by which the software of election equipment is installed and upgraded are often highly dangerous.
- **Design and Code Quality Problems** - Universal poor design and coding practices lead to a brittle system that is a potentially irreversible source of security problems.
- **Previously Unreviewed Components** - This chapter concludes with a brief characterization of four previously unreviewed Premier components: the EMP server, Digital Guardian, ExpressPoll and the Voter Card Encoder.

The findings detailed in this and the following chapters are fully consistent with previous studies. In particular, we found the architectural and systemic findings of the Premier Source Code Report of the California TTBR study to be universally true. Here, we build upon those observations and attempt to more generally characterize the specific classes of security issues within the Premier system.

Note that the descriptions below are simply a sampling of the problems that arise from confirmed and newly identified issues. There are many other attacks resulting from these and other issues that can have serious negative consequences for real elections.

[1] **Style note**: This chapter references numerous issues raised in the following chapters as demonstrative or enabling of some broader concern. These issues are denoted by reference to the chapter/section in which they are defined. For example, issue 13.2.1 discusses the lack of protections on Premier memory cards, and is referenced as "(13.2.1)". Readers are directed to the referenced section for in-depth detail of the issue, its impact on the system, and procedural or technical mitigations, if any exist.

12.1 Ineffective Data and Privacy Security

12.1.1 Memory Cards

Memory cards are the central device for storing and communicating election data. The cards are programmed by GEMS, carried to the polling place, used to collect voter choices during the election, carried back to the election headquarters, and tallied at the end of the election. We found the memory card protections were ineffective at preventing an attacker from viewing and modifying data held on these cards.

Memory cards in the AV-OS are unprotected (13.2.1)–an adversary who gains access to a card may modify it in arbitrary ways, by creating or eliminating votes, changing ballot definitions, or even modifying the programs that run on the election devices (13.2.10, 13.2.11). Attempts to protect the card are ineffective (13.2.3, 13.2.5). Memory cards in the AV-TSX are protected using a *Data Key*. However, this (and other) keys are not adequately protected (13.3.12, 13.3.4)–thus, and attacker can extract the key from Premier devices and perform the same forgery/modification as in the AV-OS.

12.1.2 Voter Privacy

One of the core requirements of an election is voter privacy; a voter's choices should not be exposed to anyone, including election officials. The Premier system failed to meet this goal. On the AV-TSX, the votes are stored in order on the memory card (13.3.19), with a timestamp in the VVPAT (13.3.20), and in memory with a serial number that is generated deterministically using the system's Data Key (13.3.21). Any one of these vote representations can be used in conjunction with poll books and observed voter order to determine voter choices.

The AV-OS does not record individual votes, but tallies them as they are read from the optical scan. The vote order is retained by the stacking of ballots in the ballot box (14.4.2). By exploiting this, an attacker could recover voter choices as in the AS-TSX.

12.1.3 GEMS

The GEMS server is a central repository for election data. It maintains the election data in database files on the local hard disk (with the file extension `.mdb`). These databases are largely unprotected, and can be freely accessed (13.1.2). The review team built an application to access and modify the election database within a few minutes using simple and widely available Visual-Basic tools. Moreover, GEMS uses the Microsoft Jet interface to manipulate databases (13.1.1), whose use in election systems has been advised against by Microsoft for stability and accuracy reasons.

The GEMS implementation also provides other means of exploiting the systems and gaining access to the internal data. For example, GEMS does not provide adequate input filtering from fields returned from the database (13.1.7). This can be used to launch buffer overflow attacks. There are numerous other methods the exploit errors in the system to compromise GEMS and therefore alter internal data (and the databases) with impunity (13.1.4, 13.1.6, 13.1.9, 14.3.1).

Access to the GEMS functionality is governed by passwords that can be extracted and cracked using standard password cracker tools, or simply by adding an attacker "user" into the GEMS database and giving them administrator rights (13.1.8). Such access can be used to manipulate election data and results through normal

Premier interfaces, possibly arousing less suspicion than other more invasive activities.

12.2 Ineffective Cryptographic and Hardware Security

12.2.1 Key Management

Cryptographic keys are used in AV-TSX, EMP, and GEMS to preserve the secrecy and integrity of election data, and for securing communication between devices. We have found that the creation, storage, and use of the keys is insufficient to ensure an attacker cannot view or modify election data.

The AV-TSX uses several keys to implement security; a *data key* encrypts votes on the memory card, a *smart card key* used to authenticate the AV-TSX to a smart card, and a *system key* used to secure the keys. The data and smart card keys are placed in a *key file* encrypted with the system key in the AV-TSX.

The mechanisms used to store these cards on the AV-TSX are insufficient to prevent an attacker from obtaining them. The system key is derived from the serial number printed on the side of the voting device, and is therefore easily attainable (13.3.5). An attacker that gains access to the filesystem on the AV-TSX (by exploiting, for example, a buffer overflow) can retrieve the key file and readily recover the other keys–thus enabling any number of other attacks on the election and results.

Note that the data key is the same for all devices in the county (14.1.2). This is poor security design–compromise of any one of potentially hundreds of devices is sufficient to subvert an entire county. Here the Premier design violates a basic *isolation* tenet of security engineering; compromise of a single precinct provides materials to compromise any precinct and the election headquarters. Further, if such compromise occurs, it will be impossible to identify which precinct is responsible for the breach.

The EMP and AV-TSX devices can directly communicate with GEMS over a network. This communication can be secured using SSL, a popular and secure communications protocol commonly used for web applications. However, the keys used are the same within the county (14.1.6) and are protected with the weak fixed password "diebold" (13.3.11).

12.2.2 Password Management

GEMS governs access to important system features via user logins. Users logging into the system provide a username and password. Each user has a set of rights to system functions, e.g., backup, election activities, auditing. Users defined as having administrator rights generally have more rights than non-administrator users. The users and their passwords are maintained in a database file on the local operating system. In Ohio, these files are protected from misuse by the Digital Guardian (DG) application.

DG can be disabled or bypassed (14.7.1, 14.7.3, 14.7.4) by a user who has access to the GEMS desktop (i.e., is able to login to Windows). Thereafter they can freely extract password data (hashes) from the database by accessing the files directly (13.1.8). Once retrieved, an attacker can use widely available password crackers to recover the unhashed password. However, this process can take hours or days on a powerful computer. The attacker may decide instead to simply replace a target victim user's password with one of its choosing. Thereafter, they could login under the victim's account, perform whatever operations they choose, and logout. It they desire to cover their tracks, they could restore the original password. They can thus become any back-end user and perform arbitrary election management operations *without ever learning any GEMS password*.

The AV-OS supervisor PIN (used to gain access to the administrative menus on the AV-OS) can be read directly from the memory card (it is obfuscated in a trivially breakable way) (13.2.8). The Smart Card PIN (used to authenticate smart cards for the AV-TSX) programmed into each security card is not recoverable by any technical means that we know of.

12.2.3 Hardware Tokens

Voter smart cards are used in part to authenticate users to the AV-TSX. The Voter Card Encoder (VCE) is a handheld device that creates these "voter cards". The poll worker inserts a smart card into a slot on the device. The VCE then encodes them with authentication information that the AV-TSX recognizes using a cryptographic key. The smart card is given to the voter, who carries it to the AV-TSX to begin voting. The smart card allows the user to vote once.

The encoding process is rather open–once enabled, the VCE remains enabled until it is turned off (14.5.2). Thus, an attacker able to obtain an enabled VCE can manufacture any number of cards. Moreover, if proper procedures are not followed, default poor keys can be used (14.5.4).

12.3 Unsafe Software Management

The Premier system provides a number of software management features that allows users to update or otherwise "program" the system. These features lead to vulnerabilities that allow an attacker to compromise the system in serious and sometimes undetectable ways.

12.3.1 Installation Procedures

The AV-TSX automatically upgrades its software without any protections. Any memory card inserted into the AV-TSX while it is restarted is accepted as an authentic update. The AV-TSX loads files from the memory card to replace its bootloader and operating system (13.3.1) or Premier election software (13.3.2). These operations and errors in the installation process itself (13.3.3) allow an attacker to completely replace all software on the voting terminal. Note that no logging of the software upgrade occurs–once completed, there is no way of knowing the software has been replaced.

The ExpressPoll electronic poll book operates in essentially the same way–the bootloader and operating system (14.6.2) can be replaced by an adversary. These attacks could be used to monitor poll activity and possibly aid in recovering voter choices (13.3.19), or to add fraudulent voters or remove legitimate ones (14.6.3).

The Voter Card Encoder (VCE) loads whatever software the user provides it (14.5.3). The attacker restarts the device, accepts the potentially malicious software upgrade through the button interface, and the VCE reprograms itself with software provided by an attached laptop or other device.

12.3.2 AccuBasic

AccuBasic is an interpreted computer programming language that creates reports generated from the election data. The AccuBasic scripts (programs) are read from local memory cards by the AV-OS and AV-TSX

voting systems and executed on the local machine, for example, to print "zero-reports" at the beginning of an election day.

AccuBasic programs stored on the memory card can be used to compromise the AV-OS (13.2.10) or AV-TSX (13.3.13). There are design and code errors (e.g., buffer overflows) which can be used to compromise the voting device. Once compromised, the attacker can modify vote counts, ballot images, and forge reports without restriction. Moreover, improperly designed scripts can be used to print false reports or crash the machine entirely without exploiting any bug in the AccuBasic interpreter. These attacks can be masked in the AV-OS by exploiting a design flaw in the audit log design (13.2.4). A previous analysis of AccuBasic also identified deep flaws in its implementation, and highlighted a number of serious issues that arise from its use in elections.

12.4 Design and Code Quality Problems

Errors in coding and design are widespread in the Premier system. These issues are visible in the descriptions in the following chapters, and lead to serious vulnerabilities that can affect the processes and accuracy of an election. Recognition of these broad failures in the engineering of the Premier system is not new to this analysis–the underlying reasons for these issues have been widely reported. This section highlights some of the more important of these causes.

The issues found in the Premier system can most centrally be attributed to:

- *Complexity* - The Premier system is comprised of many components. The devices run on a variety of hardware platforms and run software built on many thousands of lines of source code developed in several different programming languages. Even under the best of circumstances, ensuring security in such a complex environment is exceptionally difficult.
- *Lack of Software Integrity* - The Premier system does not provide the basic mechanisms to ensure the integrity of software. As a result, it is prohibitively difficult to determine if the software running on the Premier system during an election has be compromised or replaced by an attacker.
- *Security and Software Engineering Practices* - Premier does not exhibit best security practices appropriate for high value systems. The apparent lack of detailed security requirements engineering and modeling, absence of security training, poor coding, failure to apply appropriate software testing and red-teaming has lead to the vulnerabilities discussed in this report.
- *Reliance of COTS software* - Much of the Premier system is built on commodity software such as the Windows operating system. Such systems have many vulnerabilities that are exploitable by an attacker. Moreover, these vulnerabilities are uncovered constantly, thus presenting nearly insurmountable problems in defending the voting equipment from an attacker (who may know about an exploit long before a patch exists to fix it). Such problems are not limited to the operating system–there are several third-party commercial software libraries upon which the systems are built. The evaluation teams did not have access to the source code of these libraries.

One might be tempted to believe that coding or design errors lead to inconsequential vulnerabilities. Such a belief is unfounded–even small coding errors can expose the system to fatal misuse. For example, simple failures in design and coding result in issues that allow a voter to cast as many votes as they want (14.8.8) or allow a malicious election official to "pre-stuff" the ballot box before the polls open (13.2.7).

The remainder of this section identifies three areas of central concern, and demonstrates the how these failures lead to often serious system issues. These descriptions do not attempt to catalog all such issues, but highlight several the are emblematic of the root causes of the classes of failures.

12.4.1 Inadequate Security

One of the central limitations of the Premier system design is its failure to anticipate attacks. For example, removable memory card contents are unprotected from reading or modification in the AV-OS (13.2.1). This design introduces vulnerabilities that require draconian procedural controls over the card. Such controls prohibit such common practices as equipment sleep-overs. To cite a second example, a single AV-OS administrator password is used throughout a county. Therefore, anyone authorized to access any AV-OS in a county is implicitly authorized to access all AV-OS devices in the county. This is a failure of *least privilege*, an accepted principle of security design that mandates no user should be given more access than is strictly necessary. The same least privilege failure is manifest in the use of singular country-wide keys (14.1.2) and certificates (14.1.6).

Other design problems reflect a failure to understand required security functionality. For example, the audit log on the AV-OS allows up to 512 log entries (13.2.4). After the 512th log entry is created, the oldest log entry is overwritten. The purpose of an audit log is to be able to track the historical operation of the system. This functionality is essential to uncovering and understanding attacks, and for providing evidence of correct operation. The AV-OS log fails to meet this requirement in that it only understands a limited window of history. This leads to issues that can compromise or invalidate the audit function–any attacker who can force the system to create audit log entries (such as by causing errors) can cover his tracks.

Still other failures represent failures to embrace good security practices. For example, the very idea of a "default cryptographic key" is counter to good security (14.1.3). There is no such thing as a default secret, as under a reasonable security definition, if it is a default, then it is not a secret. Thus, by providing an insecure default the system behaves in an insecure way unless it is explicitly configured to do otherwise. Other examples of failures of good practice are the lack of effective authentication in software installation and update (13.3.1, 14.6.2) and the lack of safe programming practices (see below).

12.4.2 Improper Use of Security Technology

Improper use or implementation of security is the source of a large number of issues in the Premier system. Such problems are a consequence of a lack of security training for the designers and coders, and in many ways reflect an incomplete or flawed security model for the system.

One class of technology misuse is the improper implementation of cryptography. Standardized methods for ensuring integrity of data and software are not applied, e.g., digital signatures and hashed message authentication codes. Instead, home-brewed and often trivially breakable solutions are used. For example, check-summing (13.2.3) and other forgeable metrics (13.2.5) are used for integrity metrics (to determine if an attacker has corrupted data or software). Thus while their intent is laudable, the misunderstanding of common principles of security renders their function essentially useless. Other examples of misuse of security include, among many others, faux encryption (breakable obfuscation 13.2.8), poor security protocol implementation (13.2.2), and misuse of digital certificates (14.1.6).

A second class of technology misuse relates to a failure to appreciate the limitations of applied security technologies. To illustrate, much of the security of the back-end systems and data relies on the Windows

security infrastructure. Such controls have a long history of bugs, and there are many limitations of their use. In one such case, GEMS restricts access to key features by "graying out" (disabling) certain menu items on the user interface when logged-in users are not authorized to use them. However, disabled menu items in applications can be enabled using freely available Windows tools–thus, a user can simply circumvent the security mechanisms by re-enabling the menu items (13.1.3). Other mechanisms attempt to protect important data such as the election database using Windows filesystem access rights, but such protections again are easily circumventable (13.3.10). The Digital Guardian security tool attempts to augment the Windows security mechanisms, but flaws in its design and implementation allow it to be readily circumvented (14.7.1, 14.7.3, 14.7.4, see Section 12.5.2 below).

12.4.3 Unsafe Programming Practices

Attackers often compromise systems by exploiting simple coding errors. Many of the problems identified in this report are directly related to these errors. Our analysis shows that poor coding practices are visible throughout the system.[2] We illustrate the poor code quality by highlighting the substance and impact of two demonstrative classes of coding failures below:

- *Input checking* - A program that assumes but does not check that a particular input is *correct* can often be mislead, often critically, into crashing (14.1.4) or behaving in malicious ways (13.3.18). Common attacks that exploit improper inputs include buffer overflows (13.2.6), integer overflows (13.2.7), injection attacks (13.1.6), and `printf` attacks (13.2.6). These attacks can be used to infect the system with a virus, to change vote totals, to extract cryptographic keys and passwords, or any number of other malicious activities.

  Note: Input checking errors are ubiquitous in the Premier system. We found many more than was feasible to report, and we are certain that there exist many more than we found. It seems unreasonable to expect that such errors will or can be completely eradicated from the source code, and thus they will continue to represent serious sources of security and reliability problems.

- *Misplaced Trust* - Programs that assume external entities or data sources are *trustworthy* are often subject to manipulation. For example, misplaced trust leads to incorrect software being installed (where every memory card and local filesystem is trusted to provide correct updates–a possibly unreasonable assumption, 13.3.2, 14.6.2). Note that misplaced trust is not just a coding issue–misplaced trust in the broader design can also lead to vulnerabilities that exploit falsely placed trust in the users or system components.

Ultimately, the Premier code exhibits a pervasive lack of *defensive programming*. To simplify, defensive programming is a style and practice of writing source code that goes to great lengths to identify and respond to bad or fraudulent data and environments. This includes, among many other strategies, checking for errors, validating inputs, failing gracefully when errors are detected, and auditing security relevant events in the system. Thus, a necessary component of improving the security of the Premier system is not only fixing the bugs in the system, but also changing the mindset of the developers as they fix them. A defensive philosophy must be part and parcel of future design and development efforts. Failure to embrace such philosophies will result in continued security and quality issues.

[2]The poor quality of the Premier system code has been widely reported. We do not provide a detailed treatise on secure programming in this report, but refer to numerous and extensive critiques in previous reports, programming texts, and scientific literature on software and security engineering for the problems of and solutions to poor code quality.

12.5 Previously Unreviewed Components

This study included four Premier components that had not been subject to prior analysis. We briefly characterize the findings of our evaluation of these components below.

12.5.1 EMP

The Election Media Processor (EMP) is stand-alone device that creates and reads up to six memory cards at a time. This device is used to make the process of setting up and tallying an election more efficient. EMP runs a Windows operating system. The vast majority of the EMP system was developed using source code from the AV-TSX touchscreen terminal (the only device previously available for creating or tallying memory cards).

The EMP appears to have imported several code quality issues from its AV-TSX predecessor. Stability issues are prevalent; poorly or maliciously formatted data from a AV-TSX (14.1.11), the GEMS server (14.1.10), or user input (14.1.4) can freeze or crash the EMP in dangerous ways. Such problems may allow an attacker to compromise the EMP software. Once compromised, the malicious EMP can then compromise other GEMS, AV-TSX, and AV-OS devices through virus-infected communications or memory cards.

Further security issues found in prior systems are also present in EMP; the cryptographic keys used to protect critical components are fixed–thus insecure–(14.1.10), can be poorly chosen (14.1.3), are widely distributed (14.1.2), or are manipulable (14.1.8). The impact of these issues is that an attacker who gains even brief access to the system is able to modify or other forge election data or introduce viruses that can compromise all election systems within the county (14.1.1).

12.5.2 Digital Guardian

The Digital Guardian (DG) software package is installed on the GEMS server to improve its security. This serves as a extra layer of protection against attackers who gain access to the computer and attempt to manipulate Premier and Windows files and applications. DG implements a local *policy* (configuration) that specifies which users may access files and databases, and which programs can and can not be used.

We found that DG was circumventable. The utility could be eliminated from running at all by booting the system from an external source (14.7.3), by circumventing boot protections (14.7.1) or by simply by disabling of the utility through the operating system management tools (14.7.4). These attacks can be used to render DG inoperable, and thus remove its protections.

DG's policy enforcement was also ineffective. The means by which dangerous applications are prevented from running are easily circumventable (14.7.6) or fail altogether (14.7.7). Moreover, the scope of such protections was very limited (14.7.9), and could be used to completely bypass the protections on, for example, elections databases (14.7.8).

12.5.3 ExpressPoll

ExpressPoll is used to register voters at a polling place, and is intended to replace the physical poll book. We were not given source-code to perform our analysis. Therefore, there may be errors in the implementation that we did not have resources to discover.

We were able to completely compromise the ExpressPoll within a few hours; however, once discovered, the attacks take seconds. The voter database is unprotected (14.6.3), leaving it vulnerable to an attacker. Further the unsafe software updates (14.6.2), booting procedures (14.6.2), and software configuration practices (14.6.4) render the ExpressPoll highly vulnerable to compromise. These vulnerabilities could be used to introduce a number of illegitimate voters into a precinct, or prevent legitimate voters from voting by removing them from the database.

Finally, note that while the ExpressPoll is currently not certified in Ohio, the ExpressPoll unit we received was from a precinct in Richland county.

12.5.4 Voter Card Encoder

The Voter Card Encoder is a pocket-calculator sized device used by poll workers to give voters access to cast their ballot on the AV-TSX. Before an election occurs, an election administrator inserts a valid Security smart card to program the device with the correct cryptographic keys. The election administrators at each precinct can then activate these devices by inserting their Supervisor smart card. Poll workers can then simply insert Voter smart cards and press "YES" to encode them.

This simple device acts as the gateway to an election; however, it is not protected as such. Once activated, we found no restrictions in number of valid voter cards that can be created. An attacker able to steal such a device could therefore cast multiple votes (14.5.2). Simply having access to the Voter Card Encoder is enough to load arbitrary software onto the device (14.5.3), an attack that is exploitable to make the use of a Supervisor card unnecessary. Moreover, this device accepts Supervisor smart cards with a default cryptographic key that has been posted on the Internet for years (14.5.4).

12.6 Audit Procedures

Audit logs are a necessary tool when performing a forensic examination of any computer. If a machine malfunctions or an attacker gains control, such logs can alert an auditor of the presence of such events. However, the ability to use audit logs is entirely dependent upon their accuracy. If an attacker can gain control of a machine and then edit logs or if legitimate machine failures are not recorded, the value of such logs are minimal at best.

All voting equipment created and used by Premier, except Voter Card Encoder devices, records audit logs in some fashion. However, sufficient mechanisms to protect the integrity of these devices are not present in these systems. Because of the key management issues in the AV-TSX (13.3.5), a virus can delete or fabricate logs and hide its presence. Potentially critical changes, such as loading a new bootloader or operating system (13.3.1) or BallotStation software (13.3.2) are simply not logged. The AV-OS can record a maximum of 512 operations in its audit log, after which it simply copies over old information (13.2.4). An attacker could simply compromise such a machine and perform 512 operations to cover their tracks. A less patient adversary could simply change the logs on the memory card as they are not protected with any key (13.2.1). The log files on the EMP server (14.1.5) and the ExpressPoll (14.6.6) are also not protected by any key allowing anyone with access to the system to delete or forge this information. The audit logs for the GEMS server, which are stored in the election database itself, can also be modified by anyone with access to the file (13.1.2). Digital Guardian, which has logging capabilities, does not record logs in Ohio (14.7.10).

Secondary sources of audit logs, such as VVPAT units, are also easily circumvented or disabled. Because the

plastic housing protecting this device is flexible, an attacker can easily cut the wires to the printer (14.8.1) or steal (14.8.2) the paper record. An attacker could also simply inject a common household chemical into the AV-TSX housing and, without breaking any seals, destroy all previously cast ballots (14.8.3).

The audit logs created by Premier voting equipment are therefore not sufficiently protected to provide trustworthy information for forensic analysis.

CHAPTER 13

PREMIER ISSUE CONFIRMATION

The sections of this chapter detail our confirmation of the issues raised in the Diebold/Premier Source Code report developed as part of the California TTBR study. The sections are organized by component, and provide a brief summary of each issue raised, detail of the substance and potential exploitation of the issue, the means by which the issue was verified, and a strategy for the mitigation of the issue (if any level of mitigation is possible). The public version of this report contains references to sections of the private report. These non-public sections contain confidential information identifying where in the source code or design the issue is manifest, or other vendor-confidential information. Note that all source code, file names, and other apparent vendor specific information in the public version have appeared in previous election study analysis reports.

We were able to conclusively confirm 41 of the 44 issues raised in the original California TTBR report and all 3 additional issues identified in the Connecticut study of the AV-OS PC. For two of the California TTBR report issues (CA issues 5.1.10 and 5.2.13), our inability of conclusive confirmation hinged only on a lack of access to unredacted California reports and extensive time required to reproduce the analysis. These issues were confirmed by two previous studies, and we have no reason to believe they are not correct. Of the remaining exceptions, CA issue 5.1.5 (13.2.5) indicates that the AV-OS PC memory card can be replaced while the ballot is being scanned; however it was previously only analyzed from the source code aspect. Our analysis of the source code came to the same conclusion as the CA review; however, we were unable to produce an environment under which we could remove the card without the system halting.

13.1 Premier GEMS Vulnerabilities

13.1.1 GEMS uses the Microsoft Jet data layer

The election database is stored in a single file interfaced by the Microsoft Jet data layer. This interface is the same used by Microsoft's Access office productivity software. Many question Jet's integrity under large concurrent loads. An attacker causing undue stress to the GEMS server during tabulation could cause general miscalculation of results.

Description: The GEMS server stores its election database, including information about races and post election results, in a Microsoft Access `.mdb` file using the Jet data layer. While this provides convenience when backing up an election by burning a single file to CD-R, many question the integrity of Jet, and

Microsoft itself recommended against using Jet in Cuyahoga County, Ohio.[1] Microsoft recommends using MS SQL server for critical database applications. During our investigation of the GEMS, we found that the source code appears to contain support for MS SQL server. Hence, moving to an architecture that uses MS SQL server may not require significant changes to the GEMS source code.

This issue was previously identified by Ryan and Hoke,[2] and reiterated by the CA Diebold source code review (5.3.1). Ryan and Hoke provide a more detailed look at the use of Jet.

Prerequisites: Participation in the tabulation procedure.

Impact: By using Jet, the GEMS tabulation is subject to potential corruption. Scott Massey, a Microsoft Spokesman, indicated that connection loss during data transfer can result in database corruption. An attacker causing undue stress to the GEMS system or connected network during the tabulation process could cause general miscalculation of results.

Procedural Mitigations: None.

Verification: This issue was confirmed through source code analysis and study of documentation.

13.1.2 GEMS databases can be modified with access to the local hard disk

When an election database is opened, GEMS prompts the user for a username and password and restricts what parts of the election database the user can read and modify. An attacker with access to the local hard disk can directly read or modify the election database to overcome these restrictions. This technique can be used to obtain passwords, or change the election setup after it has been finalized.

Description: Every election database contains a list of valid usernames and passwords. GEMS uses this list to ensure that only authorized users can view and make modifications to an election. Additionally, once a database has been "set-for-election," no one should be allowed to make changes to the election, e.g., modifying the candidates on the official ballot.

While the GEMS application can restrict how users access the election database through it, it cannot control users who access the database via other means. GEMS stores each election database in a single `.mdb` data file; this is the same format used by Microsoft Access. We confirmed that election databases can be opened and modified by Microsoft Access as well as other file (HEX) editors.

Modifying the election database is easier than described in the CA Diebold source code review report. As a proof of concept, we wrote a five line Visual Basic script using the ADODB library to retrieve the election administrator's password hash (see Issue 13.1.8). The script can be written in any text editor from an attacker's memory in a matter of minutes and can be trivially modified to perform any `SELECT`, `UPDATE`, and `INSERT` query. Additionally, we created a slightly longer script (under 50 lines) that provides an attacker an interactive SQL shell. Such a tool would allow an attacker to freely explore the election database and destroy tamper evidence with minimal prior knowledge.

This issue was previously identified in the CA Diebold source code review (5.3.2).

Prerequisites: To exploit this vulnerability, an attacker requires a few minutes access to the GEMS server file system. In particular, the `.mdb` files used to store election information and results.

[1] Bob Driehaus, 'Audit Finds Many Faults in Cleveland's '06 Voting'. The New York Times, April 2007 ⟨URL: `http://www.nytimes.com/2007/04/20/us/20ohio.html`⟩, ISSN 0362–4331.

[2] Thomas Ryan and Candice Hoke, 'GEMS Tabulation Database Design Issues in Relation to Voting Systems Certification Standards'. In Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop. August 2007.

Impact: An attacker has complete access to the election database to read and make changes to any information. Example modifications include changing a ballot definition after a database has been "set-for-election" (similar to Issue 13.1.5), manipulating the audit log to hide tampering, and creating new database users with administrative access (see Issue 13.1.8). User passwords can also be derived from the database contents.

Procedural Mitigations: Prevent unauthorized access to `.mdb` files. Note, however, that Chapter 14.7 describes vulnerabilities in Digital Guardian, a security tool added to protect these files.

Verification: This issue was confirmed via demonstration with the GEMS server.

13.1.3 GEMS relies on the graphical user interface (GUI) to enforce security

The GEMS application keeps users from performing certain actions, e.g., performing unauthorized administrator actions or modifying an election definition after it has been "set-for-election", by disabling ("graying out") parts of the interface to make them inaccessible. Freely available software used simply to supplement basic Windows features can enable any interface in windows. This software can be used to subvert the GEMS controls.

Description: Each user of the GEMS server has limited access to the database. The practice of least privilege, or granting each user only the rights they need to do their job and none more, is an excellent security practice. Unfortunately, the mechanism by which least privilege is implemented on GEMS is vulnerable to attack.

Access to the database is controlled through the use of "widget" state. For instance, user *Alice* can use GEMS to send ballot definitions to the EMP server. *Alice* can not, however, access administrative options or change fields such as party affiliations of candidates. To prevent access to these fields, GEMS grays out these options. However, the access control mechanism does not extend any further than the GUI.

Using a freely available program , we were able to access all administrative options as the *Alice* user. Every field in the database could be changed even if access was not originally given to a particular user. The property of least privilege was therefore violated and control of the ballot given to the attacker.

In addition to providing privilege separation, GEMS restricts certain actions to take place once a database has been "set-for-election." Using this freely available program, we were able to change various parts of the election database after memory cards had been created. Of most significance, we were able to modify fields that control the election reporting sequence. This makes Issue 13.1.5 significantly worse.

This confirms and extends the vulnerability (5.3.3) reported in the California Source Code Report.

Prerequisites: Anyone with an account on the GEMS server, the ability to copy files to the GEMS server, and an account for the target election database could launch this attack.

Impact: An attacker could change any field in the database, potentially corrupting the ballot before it was sent to polling sites. Election results could also be corrupted by switching the mapping between candidate names and identifiers. With unrestricted access to GEMS and the database, there are few limits to what an attacker could achieve.

Procedural Mitigations: None.

Verification: This issue was confirmed via demonstration with the GEMS server.

13.1.4 Third parties translation services may introduce a virus into the system

The Premier system documentation referring to translation agencies instructs the election administrator to either send the entire election database or a few configuration files to a translation agency and to consider these files as official upon return. An attacker at the translation agency could use these files to introduce a virus in to the system. However, to our knowledge, Ohio does not use third party translation agencies.

Description: Both GEMS and the AV-TSX use RTF files to provide multiple language support. Issue 5.3.4 of the CA Diebold source code review indicated that the "GEMS Election Administrator's Guide" described two processes for translating English into a foreign language. One process (Section 5.4.2.3) includes sending the entire election database to the translator, and the returned database is considered the official database. A malicious translation agency can make undetected changes to the ballot definition and introduce changes that exploit the GEMS server (see Issue 13.1.7). The other process (Section 5.4.2.2) uses export/import RTF features of GEMS to provide the translator only with the RTF files. While safer, the latter process still allows the translator to create malicious RTF files that may exploit buffer overflows in the GEMS and AV-TSX (see Issue 13.3.15).

The CA source code review team was provided revision 8.0 of the "GEMS Election Administrator's Guide," which was published June 2005. We were provided revision 10.0,[3] published May 2006. Note that revision 9.0 was published less than a month after revision 8.0 and contains a very long changelog indicating conformance with GEMS 1.18.24. Revision 10.0 contains a terse changelog indicating: "Entire document re-written." The sections referenced in the CA report are no longer valid in revision 10.0 of the document. The document rewrite appears to have rearranged chapters, shifting sections 5.4.2.2 and 5.4.2.3 to 4.4.2.2 and 4.4.2.3, respectively. These sections capture the same notions of trust in the translation agency as reported by the CA report.

This issue was previously identified in the CA Diebold source code review (5.3.4),

Prerequisites: Access to files at the translation agency.

Impact: Malicious RTF files and election database content can exploit GEMS and AV-TSX software to introduce a virus. However, to our knowledge, Ohio does not use third party translation agencies.

Procedural Mitigations: As a *partial* mitigation, only the Export/Import RTF procedure should be used. Before importing the RTF files, a security audit of the returned RTF files should carefully look for malicious modification.

Verification: This issue was confirmed though review of available documentation.

13.1.5 Race and candidate labels may be changed after GEMS has been "set-for-election"

GEMS allows users to fix spelling mistakes in the race and candidate labels after an election has started. An attacker can "fix" the spelling by swapping candidate and race names.

Description: The GEMS election database uses standard schema design practices for storing and referring to user created information. As such, candidates and races are not identified by their textual name but rather a fixed integer, known as a *primary key* in database jargon. All references to a candidate or race, including vote counters, use the primary key. This design is well accepted and appropriate for database applications such as GEMS.

[3] Diebold Election Systems, *GEMS 1.18 Election Administrators Guide, Revision 10.0*. May 15, 2006.

Unfortunately, a design decision was made to allow race and candidate labels (but not party affiliation) to be changed after the system has been "set-for-election." Presumably this feature exists to fix spelling mistakes. However, it has the unfortunate potential side-effect of allowing a GEMS user to redistribute vote totals, even without the tool described in Issue 13.1.3.

The CA Diebold source code review described a scenario in which two races are swapped. In the first race (Mayor), the republican candidate won, and in the second race (District Attorney), the democrat candidate won. By swapping the race and candidate names, the election results indicate that the democrat candidate won the Mayor race and the republican candidate won the District Attorney race.

This attack exploits the mapping between textual label and database primary key. That is, vote counters refer to the integer primary key, and if the attacker changes the textual name associated with that integer, the results summary will happily display the new text. However, in order to exploit this vulnerability, an attacker must change the database used to generate the results summary. For ease of reporting, the Premier system includes the concept of a "master GEMS server" that can reside in the Secretary of State's office. The results from all counties using Premier equipment are uploaded to the master server, which tallies votes across counties and displays results. Any changes to the candidate and race strings at the county level will not affect the state level report. Hence, to exploit this vulnerability, the attacker must have access to the master GEMS server. However, to our knowledge, Ohio does not use a master GEMS server, therefore the attack can be performed at the county level.

The CA Diebold source code review report also indicated that an observant election reviewer could detect the tampering by comparing the race order in the election summary. In the above example, the race for District Attorney appears before the race for Mayor. Additionally, the change can be detected upon close investigation of the election database in GEMS. However, most if not all of these traces can be removed using the tool described in Issue 13.1.3.

This issue was previously identified in the CA Diebold source code review (5.3.5)

Prerequisites: A GEMS account on the GEMS server used to create the official election results.

Impact: The candidates of two races can be swapped, possibly changing the winner of each race.

Procedural Mitigations: An election summary report should be produced before an election and the ordering of races should be carefully compared to the final election summary. Additionally, summary tapes of all polling places should be manually compared against the official election results.

Verification: This issue was confirmed through source code analysis and via demonstration with the GEMS server.

13.1.6 GEMS fails to filter login field for special characters

An attacker with a valid username and password to an election database can exploit flaws in the GEMS server login process to perform arbitrary database lookups, including finding the password of other users with administrative access.

Description: Each election database has an associated list of users that have access. Users can optionally have administrative access to the database, which provides extra privileges, e.g., creating new users. General security practices recommend each system user has a unique username and password, thereby allowing audit logs to include accountability.

The user opening a GEMS election database is prompted with a dialog box asking for a username and password. GEMS does not perform any filtering on the username field. This allows an attacker to perform an "SQL injection attack" by escaping the database query using the `'` character.

The CA Diebold source code review team discovered a significant limitation of this attack. They found that vulnerability can only be used to perform additional `SELECT` queries on the database, and not `INSERT` and `UPDATE` queries, which modify the database. They believe this limitation is a result of the Microsoft Jet data layer (see Issue 13.1.1, which recommends against the use of Jet) and that if Microsoft SQL server were used instead, the `INSERT` and `UPDATE` queries would be allowed. As GEMS is configured to only work with Jet, we could not confirm if using Microsoft SQL server would allow these queries.

The ability to execute arbitrary `SELECT` queries is damaging in itself. The CA Diebold source code review team showed how the login field could be used to query for the existence of other users and whether the user has administrative access. Additionally, due to the way GEMS stores users passwords (Issue 13.1.8), an attacker can determine all of the usernames and passwords for an election database. This information may be used to gain administrative privileges or even cause malicious database accesses to be logged a different user.

This issue was previously identified in the CA Diebold source code review (5.3.6).

Prerequisites: Anyone with a GEMS server and election database account can mount this attack.

Impact: This vulnerability allows an attacker to retrieve arbitrary information from the election database, including information about other election database users.

Procedural Mitigations: None.

Verification: This issue was confirmed through source code analysis and via demonstration with the GEMS server.

13.1.7 Unsafe handling of election database content may allow a virus to control GEMS

GEMS trusts that data in the election database is malformed, and therefore does not always provide necessary input filtering. An attacker may modify the election database outside of GEMS to exploit this lack of filtering and insert a virus into the system.

Description: In many cases throughout the source code, GEMS trusts that the data returned from the database is not malformed. However, there is no guarantee that the information within the database has not been altered, maliciously or otherwise (Issue 13.1.2). Accordingly, there are a large number of instances throughout GEMS in which buffer overflow attacks are possible. We discuss two below:

One function in GEMS takes an arbitrarily long `CString` as input and returns the same string minus zeros and whitespaces. As the function iterates through the input character by character, it places non-zero and non-space characters into a buffer. The buffer, however, is initialized to hold no more than 128 characters. Accordingly, an attacker can cause a buffer overflow if they alter a field in the database to contain more than 128 characters. This vulnerability was mentioned in the California Diebold Source Code Report.

Another function in GEMS is responsible for queueing jobs. This function takes a filename and a "friendly" [4] filename from the database. Both of these names are of the type `CString`, meaning that each string can hold an arbitrarily long value. Both of these values are placed into a fixed size buffer using `sprintf()`.

[4] A "friendly" file name is more human-readable and can have spaces.

Accordingly, this code is susceptible to a buffer overflow. This particular vulnerability was not previously mentioned in the California Source Code Report.

Such problems are common throughout the code.

This confirms the class of vulnerabilities (5.3.7) reported in the California Source Code Report.

Prerequisites: An attacker requires access to the GEMS database in order to exploit this vulnerability.

Impact: An attacker could use these buffer overflows to gain control over GEMS and run arbitrary software or introduce a virus. The integrity of the election could be compromised by the exploitation of only one such vulnerability.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through source code analysis.

13.1.8 Election database passwords are stored with insufficient protection

GEMS stores election database user passwords in the database using a known weak algorithm. An attacker with read access to the database can retrieve the stored values to derive all user passwords. Additionally, an attacker with write access to the database can change any user's password or create a new user.

Description: GEMS controls access to election databases by usernames and passwords. The password is hashed using the `DES_crypt()` OpenSSL function and a two character salt (created with `rand()`, initialized by the current time). This is similar to the `/etc/passwd` file in early UNIX systems. An attacker who retrieves the hash value can use widely available dictionary attack software to determine the password. Additionally, since GEMS authentication simply looks up a username in a database table, an attacker can simply add a new entry to trivially create a new user.

The vulnerability becomes worse when combined with issue 13.1.6, in which any valid user can execute arbitrary `SELECT` queries. Using Issue 13.1.6, an attacker with a valid username and password can extract the password hash of all other users using a series of carefully constructed queries. The attacker can then run widely available dictionary attack software against the hash.

We confirmed that password hash extraction is possible via SQL queries. Using a binary search technique (suggested by the CA Diebold source code review report), we determined that an attacker requires at most 91 queries to extract a password hash. This accounts for the 13 character output of `DES_crypt()` (including the salt) and performing additional queries to determine the character case (all SQL queries in the current configuration are case insensitive). Such an attack would only take a matter of minutes.

This issue was previously found by the CA Diebold Source code review (bug:5-3-8).

Prerequisites: Extracting the password requires the attacker to have either a valid login to the election database or read access for the `.mdb` database file. Adding an additional user requires that the attacker have write access to the `.mdb` database file. Deriving the password from the hash may take hours or days, depending on complexity of the password, depending on complexity of the password.

Impact: An attacker with local access can create new accounts to control elections in under 30 seconds.

Procedural Mitigations: None.

Verification: This issue was verified using manual source code analysis and testing with the GEMS server.

13.1.9 Poor handling of integers may cause GEMS to crash

When converting integers to strings, GEMS writes the converted values into buffers that are too short. While the amount that the buffer is overflowed is likely too small to run arbitrary code, an attacker may still be able to cause GEMS to crash.

Description: GEMS copies signed integer values into fixed-length buffers. Declared to hold a maximum of 10 characters, these buffers are too short to represent the entire range of possible numbers that can be input. Because a 32-bit integer can contain up to 10 digits, the string representation may require an additional two bytes to for a negative sign and the string terminator ("\0").

Because the amount that the buffer is overfilled is so small, it is unlikely that this vulnerability can be used to run arbitrary code on GEMS. However, an attacker may still be able to cause GEMS to crash.

This confirms the vulnerability (5.3.9) reported in the California Source Code Report.

Prerequisites: An attacker requires write access to the election database for a short amount of time to exploit this vulnerability. Depending on the field, this attack may be exploitable from within the GEMS application.

Impact: An attacker could crash the GEMS server.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through source code analysis.

13.2 Premier AV-OS PC Vulnerabilities

13.2.1 The AV-OS memory card data is not authenticated

The AV-OS PC memory card contains all of the information about an election, including the ballot definition and the tallied election results. Because there are no cryptographic protections for data on the memory card, an attacker could modify various values to control an election and the results.

Description: The AV-OS PC memory card contains the election information required to process ballots and stores results for post-election processing. More specifically, it contains:

- Memory card header: firmware revision number, card size, election status, count mode (e.g., absentee), and global counters (for total ballots counted, overrides, number of election uploads, etc.)
- Election header: voting center name and number, download version number, obfuscated supervisor password (an unsigned integer), election title and date, election type, party code table, election configuration flags, and data checksums.
- Election definition: precincts, ballot cards, races, candidates, and voting positions
- Election data: race and candidate counters
- Audit log
- AccuBasic scripts (compiled)

- Memory card heap: Most of the above data is stored on the memory card heap and referenced using pointers in fixed positions.

Nothing prevents an attacker from maliciously modifying the contents of the memory card. Simple checksums are used; however, not only can checksum values also be rewritten, but it is trivial to replace values without needing to change the checksums (see Issues 13.2.3 and 13.2.9). Additionally, the security sensitive supervisor password is obfuscated, but trivially recoverable (see Issue 13.2.8).

This issue was previously identified in the CA Diebold source code review (5.1.1), however others have looked at changing the memory card contents for red teaming purposes.[5]

Prerequisites: To exploit this vulnerability, an attacker requires write access to the memory card.

Impact: The lack of memory card authentication is a catalyst for many of the AV-OS PC vulnerabilities. It allows an attacker to write anything to the memory card and entirely change the outcome of an election.

Procedural Mitigations: Tamper evident seals are a potential *partial* mitigation to denote access to the memory card slot has occurred. For practical limitations, see Section 3.5.

Verification: This issue was confirmed through source code analysis.

13.2.2 The GEMS server and AV-OS PC communication is not authenticated

The AV-OS PC communicates with the GEMS server to download the election definition and to upload election results. When downloading an election definition, the AV-OS PC does not ensure the GEMS server is legitimate. Hence, an attacker could pretend to be the GEMS server to define an incorrect ballot definition. On election results upload, the GEMS server does not ensure the AV-OS PC is legitimate. Hence, an attacker could upload incorrect election results.

Description: Upon inserting an empty memory card into the AV-OS PC, the user is prompted to initialize the card by downloading information from the GEMS server. The download can be performed either in direct mode (over a serial cable) or using a modem (however, modems are prohibited by Ohio law). Neither the AV-OS PC nor the GEMS server requires any form of authentication token, e.g., a password, and the communication occurs without encryption. The connection establishment uses a primitive challenge response protocol; however, the "passwords" and "keys" (indicated by variable names) are simple 16-bit integers, and the encryption function is the same simple mathematical function described in Issue 13.2.8. Furthermore, even if the protocol used reasonable sized keys and strong cryptographic ciphers, e.g., AES, it would not provide secure authentication.

After an election, the AV-OS PC is placed in "post-election" mode. At this time, the AV-OS PC can upload election results to the GEMS server. Like the download process, the upload does not use encryption or authentication.

This issue was previously identified by the CA Diebold source code review (5.1.2).

Prerequisites: A new election can be downloaded in a matter of minutes. To download an election to the AV-OS PC, the attacker must have a direct connection to the AV-OS PC serial port and the current memory card must be blank. However, if the memory card is not blank, obtaining the supervisor password (see Issue 13.2.8) allows the attacker to blank the card.

[5] Harri Hursti, *The Black Box Report: Critical Security Issues with Diebold Optical Scan Design*. Black Box Voting, July 4, 2005 ⟨URL: `http://www.blackboxvoting.org/BBVreport.pdf`⟩.

To upload election results to the GEMS, the attacker must have a direct connection to the GEMS server (or via proxy by inserting a forged memory card into the batch sent to the central office), and the memory card ID must have not already been used.

Finally, if modem communication is used, the attacker can perform either attack simply by hijacking the connection. However, Ohio law prohibits the use of modems with electronic election equipment.

Impact: An attacker could cause the AV-OS PC to make memory cards that "pre-stuff" an election (see Issues 13.2.9, 13.2.10, and 13.2.11). An attacker could also report incorrect election results to the GEMS server either via direct communication or by placing a forged memory card in the batch returned to the central office. However, the forged card must have a valid memory card ID that has not yet been used. If the GEMS server receives a duplicate ID, it will not count the second card. This effect could also be used to delay election result reporting, as it will require ballots to be recounted.

Procedural Mitigations: As the memory cards with the election definitions should always come from the central office, the modem and serial ports should be physically disabled in a way verifiable without opening the AV-OS PC, e.g., short all of the pins.

Verification: This issue was confirmed though source code analysis.

13.2.3 The memory card checksums do not protect against malicious tampering

The memory cards attempt to protect vote counts using a technique called "checksumming." Checksums are a means of ensuring that accidental changes (errors in network transmissions or hard disks) are not detected. Because anyone can create a valid checksum for any data, malicious changes to data can not be prevented or detected. Furthermore, due to the checksumming method used on the AV-OS PC, an attacker can often change values without changing the checksum value.

Description: The memory card uses checksums to ensure data integrity. First, there is no way to ensure the checksum values, also stored on the memory card, have not been modified. Second, even if the checksums were protected, the checksum algorithm by the AV-OS PC is trivially overcome. Integer values are checksummed by adding the values. For example, as long as the total votes cast for a race remains the same, the vote totals for each candidate can be modified without detection. The character checksum is slightly more complex, as values are XORed with a counter. However, this algorithm is equally susceptible to malicious tampering. Finally, as the checksums are calculated *after* the memory card contents are downloaded from the GEMS server, they do not even protect the system from transmission corruption (malicious or otherwise).

This issue was previously identified by the CA Diebold source code review (5.1.3).

Prerequisites: In order to exploit this issue, an attacker requires the ability to maliciously modify values on the memory card.

Impact: Memory card contents can be modified even without changing the checksum values. The adversary merely needs to ensure the total sum does not change. That is, if one value is lowered, another must be raised. Such manipulation requires insignificant sophistication. This vulnerability is an enabler for more sophisticated memory card attacks to go unnoticed.

Procedural Mitigations: Tamper seals are a potential *partial* mitigation to protect memory cards. For practical limitations see Section 3.5.

Verification: This issue was confirmed through source code analysis.

13.2.4 The audit log is unprotected and old entries are overwritten

The audit log, a security feature used to detect abnormalities, is stored unprotected on the memory card. Due to space limitations, once the audit log fills, it begins overwriting the oldest entries. Not only could an attacker arbitrarily add, modify, and remove log entries, causing the machine to log events could remove all traces of an attack.

Description: The AV-OS PC uses an audit log to record major events and abnormalities. These events are recorded in an audit log on the memory card, which is viewed as a security measure to detect misuse. The event record storage is fixed at 512 entries and when the log runs out of room, the oldest entries are overwritten. Furthermore, there is no integrity check of the log, as described in Issues 13.2.1 and 13.2.3.

This issue was previously identified by the CA Diebold source code review (5.1.4).

Prerequisites: The audit log is appended during normal operation. To overwrite older events, which may reveal tampering, and attacker need only cause the device to report abnormal, but benign events. This may take one to two hours to perform. For the manual manipulation, an attacker would need access to write to the memory card.

Impact: This vulnerability allows an attacker to cover all traces of tampering with the device and/or memory card. It can also be used by a malicious AccuBasic script (see Issue 13.2.10) to hide events corresponding to an attack.

Procedural Mitigations: Tamper seals are a potential *partial* mitigation to protect memory cards. For practical limitations see Section 3.5. However, tamper seals cannot prevent old logs from being overwritten.

Verification: This issue was confirmed through source code analysis.

13.2.5 The memory card "signature" does not prevent malicious tampering

The source code indicates that the memory card contains a variable called a "signature." While one may initially think this variable is a cryptographic signature, it is not. The variable is used to detect if the card was replaced during the scanning operation and does provide any cryptographic protection.

Our evaluation *cannot conclusively confirm or deny the issue*. Our analysis of the source code leads us to the same logical conclusion as the CA report; however, we were unable to recreate an environment to confirm the issue via demonstration.

Description: The AV-OS PC source code contains the structure `MemCardSignature` and a variable of this type stored on the memory card. The signature is recorded just before reading the ballot and checked immediately after returning. If the signature does not match, the machine halts.

The "signature" does not provide cryptographic protection. It consists of three counters: the total number of votes thus far, the number of nonabsentee votes, and the number of absentee votes. It does not protect any other values on the card.

The CA Diebold source code review identified this issue; however, they were unable to experiment with the equipment. Given that the AV-OS PC runs a single threaded operating system, our interpretation of the source code was that all the logic could not detect if the memory card was replaced during the ballot reading, as processing time is devoted to driving the motors and reading the timing marks. However, in experimenting with the AV-OS PC, we were unable to produce an environment that allowed the memory

card to be replaced during the ballot scan operation. In our of our attempts, the system halted as soon as the memory card was removed. As we discovered this late in our analysis, we were unable to further analyze the situation; however, we suspect the situation relates to system interrupts.

This issue was previously identified in the CA Diebold source code review (5.1.5).

Prerequisites: This vulnerability requires access to the memory card during the ballot scan operation. It also requires knowledge of the number of votes cast on the memory card up until the time of the attack; however, this is displayed on the AV-OS PC LCD.

Impact: As long as the total number of votes match, the memory card can be replaced during the scan operation.

Procedural Mitigations: Tamper seals are a potential *partial* mitigation to protect memory cards. For practical limitations see Section 3.5.

Verification: This issue was confirmed through source code analysis; however, we were unable to recreate an environment to confirm it via demonstration.

13.2.6 Improper string handling can result in arbitrary code execution

The memory card contains a number of textual strings to describe an election. The AV-OS PC assumes these strings are of a certain length and blindly uses them when communicating with the GEMS server. Due to the lack of string length checking, an attacker controlling the memory card could craft strings that cause the AV-OS PC to insert code into critical parts of the system, thereby allowing the attacker to take control.

Description: The source code that uploads election results from the AV-OS PC memory card to the GEMS server contains flaws that could allow an attacker to take control of the machine. The `sprintf()` function is misused in at least four places such that it may be exploitable if an attacker specially crafts fields on the memory card. One of these vulnerabilities allows the attacker to take control in enough time to dictate the entire reported election results.

The recommended replacement for `sprintf()` is `snprintf()`, which forces the programmer to specify the length of the buffer. The AV-OS PC source code does not currently have the ability to use `snprintf()`, as it is not defined in the custom IO library used for the device.

This issue was previously identified in the CA Diebold source code review (5.1.6)

Prerequisites: To exploit this vulnerability, an attacker needs to control the data written to the memory card. This may be from GEMS or by directly writing to the card.

Impact: A malicious memory card can result in arbitrary code execution that can control the election results reported to GEMS.

Procedural Mitigations: None.

Verification: This issue was confirmed through source code analysis.

13.2.7 Candidate vote counters are not checked for overflow

Computers store numbers in fixed size storage. When a number grows to large, it wraps around back to zero (a.k.a., overflow). The AV-OS PC does check to see if vote counters overflow. This allows an attacker to

start a candidate out at a disadvantage by initially assigning a very large number with the expectation that enough votes will occur to result in an overflow.

Description: The AV-OS PC uses 16-bit unsigned integers to store counters that track votes for each candidate. This means there is a maximum count of 65,535 for each candidate. When a 16-bit unsigned integer of value 65,535 is incremented, it takes the representation of 0. There is no checking to see if the counter overflows, therefore, an attacker can "pre-stuff" an election to give one candidate an advantage. See Issue 13.2.11 for an example.

This issue was previously identified in the CA Diebold source code review (5.1.7). A related attack was first demonstrated by Hursti.[6]

Prerequisites: To exploit this vulnerability, an attacker requires the ability to write to the memory card. A successful attack also requires other vulnerabilities for "cover-up." See Issue 13.2.11.

Impact: The lack of overflow checking is an enabler for more sophisticated attacks. See Issue 13.2.11 for a description of how this vulnerability can be used to allow an attacker to create a scenario where one candidate is at a disadvantage at the start of the election. This issue also has the unfortunate side-effect that if a candidate legitimately receives more than 65,535 votes on one AV-OS PC machine, the total will be reset without detection.

Procedural Mitigations: Tamper seals are a potential *partial* mitigation to protect memory cards. For practical limitations see Section 3.5. Additionally, one memory card should never be used for more than 65,535 votes.

Verification: This issue was confirmed through source code analysis.

13.2.8 The supervisor "password" is stored with inadequate protection

An attacker with access to the memory card can discover the supervisor password. The password is stored in an obfuscated form; however an attacker could trivially derive the real value.

Description: The supervisor "password" is a simple integer (like a PIN) required to perform supervisor operations on the device. Such operations include setting the GEMS dial-up phone number, changing feeding options, configuring communications ports, changing the election status, duplicating memory cards, and clearing memory cards. The password is stored in obfuscated form on the memory card. The obfuscation routine combines the password with a key (another integer stored in cleartext on the memory card right next to the obfuscated password) and two 20-bit "magic" constants hardcoded in the source code. The obfuscated form obf of the supervisor password (PIN) is as follows: $[obf = encode(pass, key) = pass + (key \times MAGIC_1) + MAGIC_2]$ To recover the password, an attacker simply needs to compute: $[pass = decode(obf, key) = obf - (key \times MAGIC_1) - MAGIC_2]$ This algorithm appeared in the CA review and is trivial to reverse-engineer given a few minutes access to the machine.

Obfuscation is never a substitution for cryptographic protection. However, simply applying cryptographic protection, e.g., storing a cryptographic hash of the password, will not fix this vulnerability. Once the attacker acquires the hash, it would take under a second to brute force guess the password on even low powered devices such as a PDA.

This issue was originally identified by Kiayias et al.[7] (who did not have source code) and confirmed by the

[6] Hursti, 'The Black Box Report: Critical Security Issues with Diebold Optical Scan Design' (as in n. 5).

[7] A. Kiayias et al., *Security Assessment of the Diebold Optical Scan Voting Terminal*. UConn Voting Technology Re-

CA Diebold source code review (5.1.8).

Prerequisites: To recover the password, an attacker requires read access to the memory card, e.g., reading the card directly or from the serial port in diagnostics mode (see Issue 13.2.14). The password is stored near the beginning of the card, therefore the entire card does not need to be read. Using diagnostics mode and the serial port, the obfuscated password and key can be retrieved in under a minute. If the magic values are known, the password can be calculated in microseconds.

Impact: This vulnerability allows anyone with access to the memory card to recover the password and perform special supervisor functions such as reopening an election after it has closed. Furthermore, the supervisor password is set in the "AccuVote-OS" dialog box of GEMS and cannot be modified after the system as been "set-for-election." Therefore all optical scan units within a county use the same password.

Procedural Mitigations: Tamper seals do not completely mitigate this vulnerability, because all AV-OS PC passwords within a county are the same (GEMS has one text box for the password that is "locked" after the database has been "set-for-election"), an attacker can steal the password from one precinct and use it in another.

Verification: This issue was confirmed through source code analysis.

13.2.9 Candidate ballot coordinates can be modified to manipulate election results

Each election candidate has a defined coordinate for an oval on the ballot. An attacker could change the coordinates in the ballot definition on the memory card such that votes are swapped or the oval location is somewhere a voter will never fill in.

Description: The optical scanner detects a vote for a candidate by looking for a filled in oval at a specific coordinate on the ballot (the ballot is viewed as a grid with row and column coordinates). The vote position coordinates are downloaded to the memory card before an election. The only coordinate verification that occurs before an election is the simple checksumming algorithm described in Issue 13.2.3. As long as the sum of the row/column start/end values is equal to the original sum, an attacker can make changes without detection. For example, $(row, column)$ can be changed to $(row-1, column+1)$, without changing the sum. As the checksum operates over *all* candidates, candidate coordinates can be swapped without detection.

This issue was first discovered by Kiayias et al.[8] and confirmed by the CA Diebold source code (5.1.9) and red team reviews.

Prerequisites: To exploit this vulnerability, an attacker requires the ability to write to the memory card.

Impact: An attacker could either swap candidates on a ballot, or shift a candidate's vote position to an area where a mark will never exist, thereby ensuring no votes will be recorded. Both attacks could directly impact the election outcome.

Procedural Mitigations: Tamper seals are a potential *partial* mitigation to protect memory cards. For practical limitations see Section 3.5.

Verification: This issue was confirmed through source code analysis.

search (VoTeR) Center, October 30, 2006 ⟨URL: `http://voter.engr.uconn.edu/voter/OS-Report_files/uconn-report-os.pdf`⟩.

[8] Kiayias et al., 'Security Assessment of the Diebold Optical Scan Voting Terminal' (as in n. 7).

13.2.10 Flaws in the AccuBasic interpreter may allow a virus to control an AV-OS PC

The memory cards contain "live code" that is run by a special interpreter on the AV-OS PC. This interpreter contains flaws that may compromise the machine's integrity. As the "live code" comes from the GEMS server, this vulnerability provides potential for viral spread.

Our evaluation has not conclusively confirmed or denied this issue. Due to time limitations and lack of access to specific unredacted reports, we were unable to identify the vulnerabilities found in the previous studies. However, we have no reason to believe the vulnerabilities do not exist.

Description: The AV-OS PC firmware resides on a read-only hardware chip. To provide flexibility and customization, AccuBasic scripts are placed on the memory cards. These are "live" pieces of code that are executed by the AccuBasic interpreter and perform customized functionality such as printing the Election Zero report.

There are a number of flaws in the way the interpreter handles script contents, e.g., buffer overflows, that allow a script to take over control of the machine. A virus infected GEMS server can create malicious AccuBasic scripts that infect the machine, e.g., it will infect the machine when the supervisor runs the Election Zero report.

This vulnerability was originally discovered by Wagner et al.[9] in an extensive study of the AccuBasic interpreter for the California Secretary of State and subsequently confirmed by the CA Diebold source code review (5.1.10). See Wagner et al. for a detailed description.

Prerequisites: An attacker requires the ability to control the AccuBasic scripts written to the memory card. This could be done either by controlling the GEMS server or by gaining write access directly to the memory card.

Impact: This vulnerability allows an attacker to escape the restricted control of the AccuBasic interpreter, providing the ability to write directly to vote counters and other critical storage. Most importantly, it allows a virus infected GEMS to extend to the AV-OS PC.

Procedural Mitigations: None.

Verification: Our evaluation has not conclusively confirmed or denied this issue. Due to time limitations and lack of access to specific unredacted reports, we were unable to identify the vulnerabilities found in the previous studies. Because this issue has been reported by Wagner et al and then verified in the California report, we have no reason to believe the vulnerabilities do not exist.

13.2.11 Memory card attacks could be masked by modifying the zero report AccuBasic script

The machine instructions to perform an Election Zero report are contained on the memory card. An attacker could hide other attacks, such as "pre-stuffing" an election, by always printing zero totals regardless of the current values.

Description: Memory card tampering may be hidden by modifying the zero report AccuBasic script. Before an election, poll workers print an Election Zero report to ensure that all candidates start with zero votes. With

[9] David Wagner, David Jefferson and Matt Bishop, *Security Analysis of the Diebold AccuBasic Interpreter*. Voting Systems Technology Assessment Advisory Board (VSTAAB), February 14, 2006 ⟨URL: `http://www.ss.ca.gov/elections/voting_systems/security_analysis_of_the_diebold_accubasic_interpreter.pdf`⟩.

no memory card authentication (Issue 13.2.1), an attacker could "pre-stuff" a memory card in a way that cannot be detected by looking at memory card vote totals (Issue 13.2.7). By modifying or rewriting the AccuBasic zero report script to print a zero total for all candidates regardless of the value on the memory card, and attacker could circumvent the zero report protection.

Additionally, as discussed by Wagner et al.[10] (Finding 10), this attack can occur no matter the election mode used to transport the memory card. The election mode stored on the card can be modified by an attacker, and the expected boot behavior can be simulated using AccuBasic scripts.

This vulnerability was originally identified by Hursti[11] and subsequently confirmed by Wagner et al.[12] and the CA Diebold source code review (5.1.11). Hursti and the CA Diebold source code review provide a step by step explanation of the attack.

Prerequisites: To exploit this vulnerability, the attacker requires the ability to write to the memory card.

Impact: Election results can be pre-stuffed in a way that can only be detected by performing a recount.

Procedural Mitigations: Tamper seals are a potential *partial* mitigation to protect memory cards. For practical limitations see Section 3.5.

Verification: This issue was confirmed through source code analysis and logical reasoning of other confirmed issues.

13.2.12 The AV-OS PC software controls the physical paper ballot box deflector

The ballot box uses a main bin for normal ballots and secondary bin for ballots with write-ins. A compromised AV-OS PC could direct all ballots with write-ins so that they are never counted.

Description: The ballot box deflector controls the storage bin to which a ballot is directed. The deflector motor is controlled by a port on the back of the AV-OS. If the AV-OS PC detects that a ballot contains a write-in, it places the ballot in an alternate bin. If an attacker exploits other vulnerabilities and controls execution, all write-in ballots could be directed to the main bin and thereby will not be counted. Additionally, the alternate bin is used if the election is configured to place specific *overrides* in the alternate bin for review. Just as with ballots with write-ins, an attacker could keep ballots with these overrides from being placed in the alternate bin.

This issue was previously identified in the CA Diebold source code review (5.1.12).

Prerequisites: To exploit this vulnerability, an attacker must have already exploited another vulnerability (e.g., Issue 13.2.10) that allows the machine to be controlled by the attacker.

Impact: All write-in votes will not be counted, and specified overrides will not be brought to election officials' attention. Additionally, directing all votes to one bin gives an attacker more information in an attack against voter privacy (see Issue 14.4.2).

Procedural Mitigations: Paper ballot recount is a potential *partial* mitigation. For practical limitations see Section 3.5.

Verification: This issue was confirmed through source code analysis and study of hardware manuals.

[10] Wagner, Jefferson and Bishop (as in n. 9).

[11] Hursti, 'The Black Box Report: Critical Security Issues with Diebold Optical Scan Design' (as in n. 5).

[12] Wagner, Jefferson and Bishop (as in n. 9).

13.2.13 A voter can cause the AV-OS PC to stop accepting ballots

A voter can perform a "low-tech" attack that renders the AV-OS PC useless until an election official resets the election using the supervisor password.

Description: The CA Diebold red team report indicated that a voter could render an AV-OS PC useless for the remainder of the election day via low-tech attacks. We confirmed a specific low-tech attack that ends an election on the AV-OS PC; however, we were able to resume voting after performing special operations that required access to the supervisor password. However, depending on the availability of knowledgeable staff and election procedures, such an attack could render the device useless for the remainder of the election day. Please refer to the private appendix (24.2.13) for details of the attack.

This issue was previously identified by the CA Diebold red team.

Prerequisites: See private appendix (24.2.13)

Impact: A voter can disrupt an election at a polling place.

Procedural Mitigations: None.

Verification: This issue was confirmed via demonstration with the AV-OS PC and ballot box.

13.2.14 Memory card contents can be accessed from the serial port

When the AV-OS PC is booted into diagnostics mode, an attacker can dump the contents of the memory cards to the serial port on the back of the device. This data includes the obfuscated supervisor password.

Description: The memory card contains the election definition, AccuBasic scripts, and sensitive information such the supervisor password. If an attacker can read the contents of the memory card, a duplicate can be made with malicious modifications (see Issues 13.2.10 and 13.2.11), or the supervisor password for the entire county can be discovered (see Issue 13.2.8)

When the AV-OS PC is booted into diagnostics mode (by holding down the "Yes" and "No" buttons on the front of the device), the user is presented with a number of options, including an option to dump the contents of the memory card to the serial port. An attacker could exploit this vulnerability to retrieve an image of the memory card.

This issue was originally identified by Kiayias et al.[13] and confirmed by the CA Diebold red team review.

Prerequisites: To exploit this vulnerability, the attacker requires access to the front and back of the AV-OS PC with a memory card containing the election definition.

Impact: With a few minutes of access, an attacker can retrieve the entire memory card image, including the supervisor password used throughout the county. Using a valid memory card image, an attacker can make a duplicate card with malicious modifications.

Procedural Mitigations: The AV-OS PC should remain locked in the ballot box at all times; however, this should only be considered a partial mitigation, as previous studies have reported the locks are easy to pick or otherwise circumvent. Tamper seals do not stop this vulnerability from being exploited, as the attack is purely external to the AV-OS PC.

[13] Kiayias et al., 'Security Assessment of the Diebold Optical Scan Voting Terminal' (as in n. 7).

Verification: This issue was confirmed via demonstration with the AV-OS PC. We downloaded the memory card image to a PC; however, we did not extract the supervisor password, as this was performed in previous studies with custom tools.

13.2.15 The AV-OS PC accepts the same ballot multiple times

The ballots used by the AV-OS PC have no identifying marks. An attacker that can maintain hold of a ballot can pull it out of the ballot box and resubmit it.

Description: The AV-OS PC reads identical ballots printed on card stock without unique identifying marks. Previous studies have shown that an attacker can maintain hold on a ballot by attaching a string or a long strip of Post-It notes. This allows an attacker to submit the same ballot multiple times. Because the there are no unique identifiers on the ballot, the AV-OS PC will tally the same ballot multiple times.

This issue was originally identified by Kiayias et al.[14].

Prerequisites: To exploit this vulnerability, the attacker requires a few minutes access to the AV-OS PC at the polling place.

Impact: An attacker could vote multiple times with the same ballot. Additionally, an attacker that can make identical copies of ballots could submit multiple votes.

Procedural Mitigations: As a *partial* mitigation, poll workers should pay close attention to voters spending exceedingly long periods of time at the AV-OS PC.

Verification: This issue was confirmed via demonstration with the AV-OS PC.

13.2.16 The AV-OS PC printer can be temporarily disabled without the supervisor password

The AV-OS PC printer is used to print various reports, including the Election Zero report, results summary report, and audit report. An attacker can disable the printer without the supervisor password; however, it will be re-enabled after the AV-OS PC reboots.

Description: The AV-OS PC printer provides important functionally for the poll worker, such as printing the Election Zero report, results summary reports, and audit reports. Disabling the printer may be an important step in mounting an attack, e.g., see step 5 of the 2006 Connecticut study of the AV-OS. This report indicated that the printer could be temporarily disabled by executing supervisor options that require the supervisor password. We confirmed that the printer can be temporarily disabled; however, we also note that the printer can also be disabled by carefully navigating the LCD menus. However, when the machine reboots, the original mapping will be restored.

This issue confirms a finding of Kiayias et al.[15] and extends it in that the supervisor password is not needed to temporarily disable the printer.

Prerequisites: To exploit this issue, the attacker requires access to the front buttons and the ability to power cycle the AV-OS PC. This access may be gained from picking the lock or otherwise circumventing the ballot box design.

[14] Kiayias et al., 'Security Assessment of the Diebold Optical Scan Voting Terminal' (as in n. 7).
[15] Kiayias et al., 'Security Assessment of the Diebold Optical Scan Voting Terminal' (as in n. 7).

Impact: The printer can be temporarily disabled. Alone this issue has minimal impact; however it may be used in conjunction with other vulnerabilities to hide traces of an attack.

Procedural Mitigations: The AV-OS PC should be locked in the ballot box at all times. However, this is a *partial* mitigation, as locks can be picked or the ballot box design may be circumvented.

Verification: This issue was confirmed via demonstration with the AV-OS PC and ballot box.

13.3 Premier AV-TSX Vulnerabilities

13.3.1 The AV-TSX will install an unauthenticated bootloader and operating system

The AV-TSX will automatically load a new bootloader or operating system if files with the appropriate name are present on a memory card. Because these files are not authenticated, anyone with access to a memory card can place any software they wish on the AV-TSX.

Description: In order to allow updates to the bootloader and operating system, the AV-TSX scans all inserted memory cards on boot. If the bootloader finds an update to either itself (`EBOOT.NB0`) or Windows CE (`NK.BIN`), it erases the previous version of the software and loads the new version from the above file(s). The difficulty, however, is that at no time is the source of these files authenticated. Accordingly, any file with these names placed on the memory card by any party will automatically be loaded and executed by the AV-TSX. This weakness was first noted by Hursti in 2006[16].

This confirms the vulnerability (5.2.1) reported in the California Source Code Report and by Hursti[17].

Prerequisites: An attacker would require approximately 30 seconds and an AV-TSX to execute this attack. Although a lock protects the memory card and power button, these can quickly and easily be bypassed through the use of a tool as simple as a ball-point pen.

Impact: An attacker could gain control of an AV-TSX and execute arbitrary software. Note that because no logging occurs during these operations, attempts to exploit this vulnerability would be known by the election administrators.

Procedural Mitigations: Tamper evident seals are a potential *partial* mitigation to denote access to the memory card slot has occurred. For practical limitations, see Section 3.5.

Verification: This vulnerability was confirmed through source code analysis.

13.3.2 The AV-TSX will install unauthenticated application updates

The AV-TSX will automatically load what it believes to be a new version of `BallotStation.exe` if a file with the correct extension is present on a memory card. Because there is no authentication of this file, an attacker with access to a memory card could run arbitrary software on the AV-TSX.

Description: In addition to providing an update mechanism for the bootloader and the operating system, the AV-TSX allows BallotStation software to be updated through a memory card. After booting Windows CE,

[16] Harri Hursti, *Diebold TSx Evaluation: Critical Security Issues with Diebold TSx*. Black Box Voting, Unredacted release July 2, 2006, May 11, 2006 ⟨URL: `http://www.bbvdocs.org/reports/BBVreportIIunredacted.pdf`⟩.

[17] Hursti, 'Diebold TSx Evaluation: Critical Security Issues with Diebold TSx' (as in n. 16).

the AV-TSX runs `taskman.exe`, which eventually begins BallotStation. Before launching BallotStation, however, `taskman.exe` scans the contents of the memory card to determine whether or not a new version of BallotStation is present. Any file with the extension `.ins` is assumed to be a valid new version of BallotStation and is therefore automatically loaded by the voting machine.

As was the problem in the previous vulnerability, there is no checking to determine if an `.ins` file comes from a legitimate (e.g., the county voting headquarters) or illegitimate (e.g., nearly everyone else) source. Accordingly, anyone with 30 seconds of access to a memory card can load whatever software they wish in the place of a legitimate version of `BallotStation.exe`.

This confirms the vulnerability (5.2.2) reported in the California Source Code Report.

Prerequisites: An attacker with 30 seconds of access to any memory card can successfully exploit this vulnerability. The physical attack itself takes no technical skill.

Impact: An attacker could gain control of an AV-TSX and execute arbitrary software. Note that because no logging occurs during these operations, attempts to exploit this vulnerability would not be known by the election administrators.

Procedural Mitigations: Tamper evident seals are a potential *partial* mitigation to denote access to the memory card slot has occurred. For practical limitations, see Section 3.5.

Verification: This vulnerability was confirmed through source code analysis.

13.3.3 There are multiple buffer overflow vulnerabilities in the handling of .ins files

The function responsible for loading new versions of BallotStation do not operate in a safe manner. Poorly formatted input may cause BallotStation to crash. Malicious input may take control of the AV-TSX.

Description: The function used to install new versions of BallotStation contains a number of buffer overflow vulnerabilities. As mentioned in the previous Issue (13.3.2), `taskman.exe` searches the memory card for `.ins` files, which it uses to install new versions of the BallotStation software. When `taskman.exe` begins reading information from the `.ins` file, it assumes that the information detailing the installation of the code will be not be too large. An adversary could simply change these values (without access to any encryption key) and exploit a buffer overflow vulnerability.

This confirms the vulnerability (5.2.3) reported in the California Source Code Report and Feldman et al[18].

Prerequisites: An attacker would need 30 seconds of time to access a memory card in order to successfully launch this exploit.

Impact: An attacker could gain control of an AV-TSX and execute arbitrary software. This would allow them to control and election and its results, possibly undetectably.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through source code analysis.

[18] Ariel Feldman, J. Alex Halderman and Edward Felten, 'Security Analysis of the Diebold AccuVote-TS Voting Machine'. In USENIX/ACCURATE Electronic Voting Technology Workshop (EVT). 2007.

13.3.4 An unauthenticated user can read and or tamper with the memory of the AV-TSX

The AV-TSX provides a mechanism that allows anyone to read or write directly from/to memory. This would allow an attacker to easily gain a copy of the compiled source code and/or maliciously change the software running on the AV-TSX.

Description: As originally noted by Hursti[19], the motherboard of the AV-TSX contains a jumper header marked `DEBUG`. When a jumper is installed at this location, a service menu is displayed over the machine's serial port. From this menu, a user can alter settings including a machine's IP address and the boot order. More critically, through the "Mini Monitor" submenu, a user can read/write to the memory of an AV-TSX. An attacker can therefore recover/rewrite the bootloader, operating system, BallotStation and all of the secrets (i.e., cryptographic keys) stored on the machine. Because there is no logging of this operation, an attacker's changes may be untraceable.

Setting this jumper requires physical access to the internals of the AV-TSX. An attacker would need to remove the eight screws holding the plastic casing together. Once the casing was opened, an attacker could complete the circuit at the jumper header by inserting a staple or paper clip. A serial cable would then be attached to the "VIBS KEYPAD" (exposed on the back side of the AV-TSX) and a computer. The amount of time needed to extract information from a machine would vary depending on the data size (seconds for keys, perhaps hours for the operating system).

This confirms the vulnerability (5.2.4) reported in the California Source Code Report.

Prerequisites: An attacker would need unrestricted access to an AV-TSX for 30 seconds to remove the screws and access the casing. The jumper could be set and the cover replaced in an additional 30 seconds. The extraction of data/programs would depend on the size of these items.

Impact: An attacker could gain control of an AV-TSX and execute arbitrary software. This would allow them to control and election and its results, possibly undetectably.

Procedural Mitigations: No election official should be given unsupervised overnight access to equipment such as the AV-TSX (known as a "sleep-over"). Tamper evident seals are a potential *partial* mitigation to denote access to the memory card slot has occurred. For practical limitations, see Section 3.5.

Verification: This vulnerability was confirmed through penetration testing.

13.3.5 The cryptographic keys are not adequately protected

The keys authenticating smart cards and protecting election data are not adequately protected. Accordingly, an attacker can easily break the mechanisms protecting an election.

Description: The AV-TSX uses a number of keys during the process of conducting an election. In order to authenticate itself to a smart card, the AV-TSX stores a *Smart Card Key* (64 bits). When authenticating an inserted smart card, the AV-TSX compares the value of the *Magic Number* (16 bits) stored in its memory against the one presented by the card. To protect the confidentiality of the votes stored on the machine, the AV-TSX uses a third key known as the *Data Key* to encrypt votes. These three keys are defined as `SCKey`, `SCMagic` and `DESKey` in the code. In previous versions of the code, these three keys were hardcoded into the software; however, they can now be set using the Security Card.

These three keys are stored internally in the AV-TSX in the file `bs-security.cf`. Before being saved to

[19] Hursti, 'Diebold TSx Evaluation: Critical Security Issues with Diebold TSx' (as in n. 16).

this file, these three values are encrypted using the *System Key* (128 bits) and the well-known AES encryption algorithm in Cipher Block Chaining (CBC) mode. The process of creating the System Key, however, creates a vulnerability.

A function then uses the machine's serial number and the well known MD5 hash algorithm to generate the System Key. Because the machine's serial number is prominently displayed in a number of places (e.g., in the bottom left-hand corner of the touch screen, on the back of the AV-TSX unit, internally in the machine's registry, etc), this value can be observed by any person using the AV-TSX. With this information, the adversary could easily run the MD5 algorithm and generate the System Key.

This confirms the vulnerability (5.2.5) reported in the California Source Code Report.

Prerequisites: Any individual capable of seeing the serial number on a machine could generate its System Key. The theft of the other keys would require access to the file `bs-security.cf`, which could be accomplished by placing a virus on a memory card or through Issue 13.3.4.

Impact: This attack creates the potential for more serious attacks. For instance, malicious software[20] (i.e., a virus) could use this knowledge to alter election results, erase system logs and/or leak the keys necessary to create fraudulent smart cards (e.g., Voter Cards).

Procedural Mitigations: Removing the serial number from public display (on the casing and the screen) would lessen an attacker's ability to generate the System Key without touching the voting machine. Because serial numbers are relatively short (6 digits), this is only a marginally effective mitigation. An attacker could calculate all possible system keys in under two seconds before an election occurs.

Verification: This vulnerability was verified through source code analysis.

13.3.6 Malicious software could alter files critical to correctly reporting an election

Because the cryptographic keys are poorly protected, a virus could untraceably change ballot definitions, audit logs and cast ballots.

Description: The correct execution of an election using the AV-TSX relies upon four types of data files:

- **Election Database (.edb) Files:** These files contain ballot definitions. The manual claims that they are named using the digital signature of the file; however, integrity is instead protected by a non-standard message authentication code (MAC) function as first described by Wagner et al[21]. Specifically, an MD5 hash of the file is calculated and then encrypted using AES-128 and the Data Key.

- **Election Resource (.xtr) Files:** These files contain resources including audio and image files, AccuBasic Scripts and other data used in an election. The contents of these files are not encrypted, but are individually authenticated using the MAC construction described above. These files share the same name as their corresponding .edb files, but are identified by their .xtr extension.

- **Ballot Results (.brs) Files:** These files contain the votes cast in on each ballot during the election. The contents of .brs files are encrypted using the AES-128 algorithm in CBC mode using the Data Key.

[20] Feldman, Halderman and Felten (as in n. 18).
[21] Wagner, Jefferson and Bishop (as in n. 9).

- **Audit Log (.adt) Files:** These files store a record of some of the operations of `BallotStation.exe`. These files are protected in a similar fashion as the Ballot Results, but instead use the System Key during the encryption process.

Given a .brs file, the corresponding .adt, .edb and .xtr files can be determined by examining the fields in the header. The current .brs file is located by reading the contents of the file `assure.ini`, which is also stored on the memory card. If no file is present, Ballot Station "may attempt to load the election corresponding to the ballot box file with the latest timestamp, or simply not load an election at all"[22].

Because the key management currently implemented in the AV-TSX is not secure (see Issue 13.3.5), malcode running on the AV-TSX could easily alter all of these files without being detected. If such an attack were to happen before an election, the attacker could use this knowledge to insert a vote-changing virus onto the AV-TSX. The attacker may also attempt to alter ballot layout or election resources (e.g., audio tracks) as an attempt to confuse voters. By changing the audit logs, the attacker could then remove any record of their activities.

This confirms the vulnerability (5.2.6) reported in the California Source Code Report.

Prerequisites: A voter or poll worker would need access to an AV-TSX and the System Key. They could then recover the Data Key and exploit this vulnerability.

Impact: An adversary would gain total control of the AV-TSX and an election. If a virus were inserted, malicious control of the election may be achieved.

Procedural Mitigations: None.

Verification: This vulnerability was discovered through source code analysis.

13.3.7 The smart card authentication protocol can easily be broken

The handshake between a smart card and the AV-TSX is not secure. An attacker can easily exploit this weakness as a means of creating their own voter cards.

Description: The authentication protocol between smart cards and the AV-TSX is insecure. The messages exchanged between the voting machine and the smart card can repeated an unlimited number of times by an attacker to cast multiple votes. The protocol operates as follows:

In an attempt to prevent unauthorized attackers from casting votes on the AV-TSX, the voting machine and smart card exchange messages containing secret values. Upon the insertion of a smart card, the AV-TSX attempts to prove that it is a valid voting machine by presenting two values: the *File ID* (16 bits) and *Smart Card Key* (64 bits). The File ID, which is hardcoded as 0x3d40, is the same for every AV-TSX machine in the country. The Smart Card Key can be reprogrammed using the Security Card. However, in the case that the Smart Card Key is rejected by the Smart Card, the AV-TSX will attempt to use a hardcoded value. This default Smart Card Key, if not changed by an election administrator, is the same for every AV-TSX machine in the country (`0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08`). Regardless of whether the AV-TSX uses a unique or the default Smart Card Key, an attacker can capture this secret simply by inserting a smart card that logs all messages sent to it. Accordingly, the Smart Card Key is instantly recoverable simply by interacting with the AV-TSX.

[22] Diebold Election Systems, *Ballot Station 4.6 Technical Data Package, Revision 1.0*. March 23, 2005.

If the smart card receives the correct File ID and Smart Card Key, it returns a message to the AV-TSX. This message contains information including the type of card inserted (e.g., a Voter Access Card) and the *Magic Number* (16 bits). The AV-TSX checks that the Magic Number stored on the card is the same as the value stored in its memory. If the two values match, the voter is allowed to begin filling out their ballot. If the values do not match, the voter is denied access to the machine. The Magic Number can easily be retrieved by an attacker with access to a smart card reader. Having intercepted the File ID and Smart Card Key above, the attacker can replay these values to the valid smart card. The attacker can then record the magic number and use it to create an unlimited number of forged voter cards.

When a voter casts their ballot, their smart card is rewritten by the AV-TSX to indicate that the card is no longer valid. However, an attacker could program a smart card to pretend to mark itself as invalid. The same card could then be instantly placed back in any AV-TSX to allow the adversary to cast multiple votes.

This confirms the vulnerability (5.2.7) reported in the California Source Code Report.

Prerequisites: This attack could be accomplished by a voter. Because smart card readers/writers are easily portable and included in a number of Personal Digital Assistants (PDAs) and smart phones, such an attack could be easily conducted without notice.

Impact: An attacker could use this weakness to create their own smart cards and cast multiple votes without being detected.

Procedural Mitigations: None.

Verification: The vulnerability was confirmed through source code analysis.

13.3.8 Security Cards can be forged and used to change keys

The smart cards used specifically for security functions can easily be forged by an attacker. Such a card could allow the attacker to make a machine unusable during and votes unreadable after an election.

Description: The AV-TSX uses three keys to protect against unauthorized access: the Data Key (128 bits), the Smart Card Key (64 bits) and the Magic Number (16 bits). The original source code analysis by Kohno et al[23] discovered that all of these values were hardcoded into the software. More recent versions of the software have addressed this vulnerability by allowing election officials to create unique values for each of these keys. By inserting the Security Card into the AV-TSX, an administrator can update these keys.

The key used to authenticate the Security Card is the same default key described in Vulnerability 13.3.7: `0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08`. While the correct Magic Number is also required to gain access to the AV-TSX, this value is also hardcoded as 0xa535. Accordingly, an adversary can create a Security Card capable of successfully interacting with any AV-TSX unit in the country.

More critically, both the Magic Number and the Smart Card Key used to authenticate the Security Card can not be updated. Accordingly, an administrator who is aware of this vulnerability can do little to prevent it being exploited.

This confirms the vulnerability (5.2.8) reported in the California Source Code Report.

Prerequisites: This attack could be accomplished by a poll worker or county administrator. Because smart card readers/writers are easily portable and included in a number of Personal Digital Assistants (PDAs) and

[23] Tadayoshi Kohno et al., 'Analysis of an Electronic Voting System'. In Proceedings of the IEEE Symposium on Security and Privacy. May 2004.

"smart" phones, such an attack could be easily conducted without notice.

Impact: An adversary could affect the election in a number of ways with this attack. By changing the value of the Data Key, the adversary could cause the results of the election to be unreadable by the central tally units. Specifically, because the county officials would not have the correct key to decrypt the results, the votes stored on the machine may not be able to be counted.

Alternatively, the attacker could prevent voters and precinct administrators from being able to use a machine. By changing the Magic Number or the Smart Card key, the attacker could prevent ballots from being cast or voting machines from being used in an election. Moreover, because the Security Cards would typically be kept secure in the county central election facility, it would be unlikely that elections administrators could fix this problem on election day.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through source code analysis.

13.3.9 The System Setup Menu is accessible without using a smart card

An attacker can access some administrative functions without using the proper smart card. This can allow them to change a number of settings and calibrations, potentially making legitimate voting difficult or impossible.

Description: Access control for the AV-TSX is provided through the use of smart cards (e.g., Voter Access Cards, Security Cards). However, an adversary can gain access to the System Setup Menu without possessing a smart card. Access to the System Setup Menu allows a user to change settings including the time and date, adjust the volume, speed and output device used for audio, change network settings and recalibrate the touch screen.

On boot, the AV-TSX will automatically enter this screen under a number of conditions including hardware failures (e.g., disconnected VVPAT printer, smart card reader failure). We were able to cause an AV-TSX to enter this mode by placing a paper clip into the smart card reader. A similar approach was successfully executed by the red team in the California report.

This confirms the vulnerability (5.2.9) reported in the California Source Code Report.

Prerequisites: This attack could be accomplished by anyone with physical access to an AV-TSX during an election.

Impact: This attack can cause a number of problems in an election. By recalibrating the screen, an adversary can make it difficult or impossible for a voter to fill out a ballot according to their wishes. By changing the audio volume and speed, an adversary can also make access for the disabled challenging or impossible. Network and modem settings can also be changed through this screen, potentially exposing the voting machine to additional vulnerabilities.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed in the source code and by physically disconnecting the printer from the AV-TSX.

13.3.10 The protected counter is not protected

The protected counter acts as a fraud detection mechanism by noting the total number of votes ever cast on an AV-TSX. Because this counter is not protected, an attacker can easily add as many votes as they wish and then hide their attack.

Description: Each AV-TSX stores the total number of votes cast on itself during its entire lifetime in a file. This "protected counter" could be used to detect scenarios including the injection of fraudulent votes by an adversary (over-voting) or machine errors leading to votes not being recorded (under-voting). Unfortunately, the correctness of the counter value in this file can not be trusted.

The protected counter is stored in the unprotected file `\FFX\AccuVote-TS\system.bin`. Because there are no protections provided to this file or its contents, it is in fact not a protected counter. Accordingly, malicious software (e.g., a virus) with access to the file system could easily change the value of this counter without detection.

The method through which this counter is incremented also lacks critical input filters. Even if such a file were protected by access control mechanisms in the operating system, malicious software would be able to increment the counter using negative values (and therefore mask the presence of fraudulent votes).

This confirms the vulnerability (5.2.10) reported in the California Source Code Report, which was originally reported by Feldman et al[24]. The lack of input validation was not previously noted.

Prerequisites: A poll worker or a voter capable of injecting a virus into the voting machine could successfully launch this attack.

Impact: Exploitation of this vulnerability would make it possible to hide indications that fraudulent votes were entered into the machine.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through source code analysis.

13.3.11 SSL certificates are not sufficiently protected

SSL is a protocol that can be used to securely communicate between the AV-TSX and the election headquarters. However, the password protecting the cryptographic keys is easily guessable - "diebold".

Description: While the use of modems to communicate election results is not currently allowed in the State of Ohio, every AV-TSX machine comes equipped with a fully functional modem. Should modem-based reporting ever be allowed, communication between precincts could be cryptographically protected using the Secure Sockets Layer (SSL). The implementation of SSL used in the AV-TSX, OpenSSL, is a well-known and publicly-vetted library.

The secure use of SSL hinges upon the ability to protect the private key stored at both endpoints of communication. These private keys are stored along with the public key in an SSL certificate, which is the file `client.pem` on the AV-TSX. While the private key on the AV-TSX is password protected, the password used to gain access to the key is both weak and hardcoded. Specifically, the manufacturers protect the private key in all AV-TSX machines with the password "diebold". Such a simple password is easily guessable and therefore provides no real protection to the system.

[24] Feldman, Halderman and Felten (as in n. 18).

We also suspect that the certificate on all AV-TSX machines is the same. From our analysis of the source code, there does not appear to be a mechanism for certificates to be loaded onto the machine separately or updated. Access to a single AV-TSX, regardless of whether or not modem communications are permitted, may potentially compromise the communications of all AV-TSX units using a modem in the country.

This confirms the vulnerability (5.2.11) reported in the California Source Code Report.

Prerequisites: A voter or poll worker capable of running a malicious program on the AV-TSX could gain access to the private key.

Impact: An adversary can potentially eavesdrop upon and change the results of an election if a modem is used. The GEMS server would also become susceptible to the injection of false election data from computers masquerading as voting machines.

Procedural Mitigations: An AV-TSX can never be attached to a network. This includes overtly by poll workers or covertly by an adversary. Physically removing the modem may provide a straightforward approach to achieving this.

Verification: This vulnerability was discovered through source code analysis.

13.3.12 OpenSSL is not initialized with adequate entropy

SSL can protect communications between an AV-TSX and election headquarters. However, because the "random" key used to encrypt such communication is created in a guessable manner, an attacker can realistically decrypt and change all communications between precincts and election headquarters.

Description: SSL connections can be established between an AV-TSX and the GEMS server to allow secure communication. Before establishing such a connection, the OpenSSL library must be initialized with pseudo-random/unpredictable data in order to ensure the security of the connections. If the psuedo-random/unpredictable data is in fact easily predictable, it becomes possible for an attacker to determine the secrets (i.e., keys) used to protect such communication.

OpenSSL is initialized on boot using two sources. The first takes the contents of the screen. The second records the number of milliseconds since the AV-TSX was booted. Both are easily guessable. In the case of the first, the contents of the screen are identical each time, meaning that this input is deterministic and entirely predictable. The second source is also largely deterministic. Given that a machine will generally boot within a relatively small time window (plus or minus a few seconds), the exact number of clock ticks can quickly be determined.

As noted in the California Report, the GEMS server contains almost identical code for initializing OpenSSL. This code makes a call to `CryptGenRandom()`, a Windows function that is accepted as a good source of pseudo-randomness. This function is also included in the code for the AV-TSX; however, it has been commented out because the version of Windows CE running on these devices does not support its use. Note that Windows CE .NET version 4.2 and beyond support this call.

This confirms the vulnerability (5.2.12) reported in the California Source Code Report.

Prerequisites: An attacker would need access to a phone/network line between the AV-TSX and the GEMS server.

Impact: Because Ohio does not allow for modem communication between the AV-TSX and the GEMS server, the problem would seem solved. Unfortunately, every AV-TSX unit comes equipped with a fully

functioning modem. The adversary may use this device to covertly transmit election results from such a device. Even if SSL is turned on by default, the adversary would be able to intercept such communication and gain access to sensitive information.

Procedural Mitigations: An AV-TSX should never be attached to a network. This includes overtly by poll workers or covertly by an adversary. Physically removing the modem may provide a straightforward approach to achieving this.

Verification: This vulnerability was discovered through source code analysis.

13.3.13 Flaws in the AccuBasic interpreter may allow a virus to control an AV-TSX

The memory cards contains "live code" that is run by a special interpreter on the AV-TSX. This interpreter contains flaws that may compromise the machine's integrity. As the "live code" comes from the GEMS server, this vulnerability provides potential for viral spread.

Our evaluation has not conclusively confirmed or denied this issue. Due to time limitations and lack of access to specific unredacted reports, we were unable to identify the vulnerabilities found in the previous studies. However, we have no reason to believe the vulnerabilities do not exist.

Description: There are a number of flaws in the way the interpreter handles script contents, e.g., buffer overflows, that allow a script to take over control of the machine. A virus infected GEMS server can create malicious AccuBasic scripts that infect the machine, e.g., it will infect the machine when the supervisor runs the Election Zero report.

This vulnerability was originally discovered by Wagner et al.[25] in an extensive study of the AccuBasic interpreter for the California Secretary of State and subsequently confirmed by the CA Diebold source code review (5.2.13). See Wagner et al. for a detailed description.

Prerequisites: An attacker requires the ability to control the AccuBasic scripts written to the memory card. This can be done either by controlling the GEMS server or by gaining write access directly to the memory card; however, the latter may require the attacker to discover the Data Key.

Impact: This vulnerability allows an attacker to escape the restricted control of the AccuBasic interpreter, providing the ability to write directly to vote counters and other critical storage. Most importantly, it allows a virus infected GEMS to extend to the AV-TSX.

Procedural Mitigations: None.

Verification: Our evaluation has not conclusively confirmed or denied this issue. Due to time limitations and lack of access to specific unredacted reports, we were unable to identify the vulnerabilities found in the previous studies. Because these issue has been reported by Wagner et al and then verified in the California report, we have no reason to believe the vulnerabilities do not exist.

13.3.14 Failure to check data on memory cards may allow a virus to run on the AV-TSX

A lack of input checking when loading files from the memory card is potentially dangerous. A virus could take advantage of this weakness to crash or control an AV-TSX.

[25] Wagner, Jefferson and Bishop (as in n. 9).

Description: When a memory card is inserted into the AV-TSX, the machine searches for configuration information in the file `assure.ini`. This file tells the AV-TSX which election-related files (i.e., .edb, .xtr) are to be loaded for the current election. Because this configuration information is relatively small in properly formatted files, the BallotStation software assumes that it will be easily stored in a small, fixed-length buffer. Because this file is neither encrypted nor its integrity protected, an adversary could easily replace this configuration information with a large number of characters. By overfilling this buffer, an attacker would be able to cause either the crash of BallotStation or potentially execute arbitrary code on the AV-TSX.

This confirms the vulnerability (5.2.14) reported in the California Source Code Report.

Prerequisites: Anyone with access to a memory card and a laptop could exploit this vulnerability with a few seconds of access.

Impact: An adversary could potentially run arbitrary code on the AV-TSX. This code could change the results of an election.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through source code analysis and demonstrated through penetration testing.

13.3.15 Unsafe handling of election resource files may allow a virus to control an AV-TSX

Election resource files are trusted to be well-formed. However, weaknesses in the handling of these files can allow an attacker to crash or potentially run malicious code on the AV-TSX.

Description: Election resource files contain additional content to enhance the user's experience during an election. However, due to a lack of input checking on the contents of these files, the AV-TSX is vulnerable to attack. The following two vulnerabilities represent realistic threats to the integrity of the election.

The first weakness is a format string vulnerability. Information used to populate data on the screen, such as the current page number on a multi-page ballot, is stored in language specific RTF files. When the data from these files is read into memory, the system does not handle these strings in a safe manner. For instance, the text `Page %d of %d` in the source code should receive two numbers from an RTF file and use them replace the `%d`s. [26] However, the data destined for this field is loaded via an unsafe call to `sprintf`. Accordingly, replacing the numbers in the RTF file with input such as "%d%s%s%s%s%s%s%s%s%s%s" would cause the AV-TSX to crash. Code could be added to the end of this string to allow the attacker to violate the integrity of the election.

The AV-TSX also uses image files to ease the election process. For instance, when a voter approaches the AV-TSX, the machine displays a graphic of a user inserting their voter card below the text "Insert Card to Begin Voting". When these bitmap files are loaded into memory, the AV-TSX looks at the header of the image file to determine its size. It then copies the file into a buffer in memory based upon this value. Because the machine trusts the header to accurately represent the real size of the file, there is no real input checking. Accordingly, an attacker can change this value and cause a buffer overflow vulnerability.

This confirms the vulnerability (5.2.15) reported in the California Source Code Report.

Prerequisites: The attacker would need access to the RTF files stored on the GEMS server, the GEMS server

[26] %d is a standard indicator used to tell the system that the data supplied should be treated as a number.

itself or a memory card. As previously mentioned, gaining access to the cryptographic keys protecting the integrity of the memory card's contents is easily achieved (Issue 13.3.5).

Impact: An attacker could exploit either of the vulnerabilities to gain control of an AV-TSX and change the results of an election.

Procedural Mitigations: None.

Verification: These vulnerabilities were confirmed through source code analysis.

13.3.16 Unsafe handling of election database files may allow a virus to control an AV-TSX

Election database files are trusted to be well-formed. However, weaknesses in the handling of these files can allow an attacker to crash or potentially run malicious code on the AV-TSX.

Description: At the end of an election, an AV-TSX can upload the ballots it recorded to the GEMS server. As a confirmation of this upload, the AV-TSX prints a "ticket" containing information stored in the election database (e.g., Date, Time, File Count, etc). Certain weaknesses in the ticket printing routines allow an attacker to potentially take control of the voting machine.

In the first vulnerability, the programmers assumed that certain election information would never exceed 512 bytes in length. Accordingly, the code uses a call to `sprintf` to copy such information into a buffer `buf`. Because there is no input checking on this function, it is possible for more than 512 bytes of data to be copied to `buf`. This buffer overflow would allow an attacker to execute arbitrary code on the AV-TSX.

The second weakness is a format string vulnerability. Because the programmers do not properly handle strings in a safe way, an attacker can inject data such as "%s%s%s...%s" and cause BallotStation to crash or execute arbitrary code.

The third vulnerability is also due to unsafe handling of strings. This format string vulnerability uses nearly identical code to the previous vulnerability and therefore can be used to exploit the AV-TSX in the same fashion.

This confirms the vulnerability (5.2.16) reported in the California Source Code Report.

Prerequisites: The attacker would require access to an election database file. Such an attack could occur with access to the GEMS server or a memory card (assuming the attacker was in possession of the correct keys using the weakness noted in Issue 13.3.5).

Impact: An attacker could exploit any one of the above vulnerabilities to change the results of an election.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through source code analysis.

13.3.17 A voter may be able to gain control of an AV-TSX

A buffer overflow vulnerability in IP address handling may allow a voter to execute arbitrary code on the AV-TSX.

Description: The values for the *IP address*, *Subnet Mask*, *Default Gateway* and *DNS Server* can be set by an administrator through the System Setup Menu. As pointed out in Issue 13.3.9, a voter can also gain access

to these settings without using any smart card. When addresses for the above fields are entered through the touch screen, they are temporarily copied into a buffer. This buffer, which can hold a maximum of 256 characters, is then copied into the system registry such that functions responsible for network communication can access this information.

While some input checking does occur, it is possible for an adversary to cause a buffer overflow using this interface. An IP address, which has the form XXX.XXX.XXX.XXX (where XXX is an integer between 0 and 255), can be written in a manner that passes the format checking while still overflowing the buffer. For instance, an IP address of `192.168.1.1` can legally be represented as `0000 ... 0192.168.1.1` (where at least 256 zeros precede "192". When such an address is entered, the AV-TSX crashes.

If additional characters (i.e., the full range of alpha-numeric symbols) could be entered, it would be possible to use this vulnerability to execute malicious code. Input validation on these fields would appear to mitigate this additional threat.

This confirms the vulnerability (5.2.17) reported in the California Source Code Report.

Prerequisites: A voter or a poll worker could accomplish this attack by accessing the System Setup Menu (Issue 13.3.9) and then entering a sufficient number of zeros before a valid IP address. Such an attack could easily be accomplished in one minute.

Impact: This attack causes the AV-TSX to crash, necessitating that the device be reset. Continued crashes may cause the device to be unnecessarily removed from service.

Procedural Mitigations: None.

Verification: This vulnerability was first confirmed through source code analysis and then physically confirmed on an active AV-TSX unit.

13.3.18 A malicious GEMS server can cause an AV-TSX to crash on download

The AV-TSX assumes that all data from the GEMS server is well-formed. Poorly formatted data may cause the AV-TSX to crash. Malicious input may allow an attacker to run a virus on the AV-TSX.

Description: One way in which election information can be delivered to a polling place is through direct download from the county's election headquarters. Included in the information transmitted by the GEMS server is a label to be attached to memory cards. This way, these cards can easily be recognized as being ready for use in an election.

As this information is sent to the label printer, the treatment of some of the data fields is unsafe. An attacker with control of the GEMS server (or the network in between the GEMS server and the AV-TSX if the SSL option is not in use) could send `printf` format specifiers (e.g., messages containing "%s") to cause the AV-TSX to crash.

Other format specifiers may allow an adversary to gain control of the AV-TSX.

This confirms the vulnerability (5.2.18) reported in the California Source Code Report.

Prerequisites: An attacker would need to have access to the GEMS server or a link in the network between GEMS and an AV-TSX.

Impact: Because modem/network communication between elections machines and the GEMS server is not permitted, this vulnerability is less likely to be exploited. However, because active modems are placed in all

AV-TSX machines, the possibility of such an attack can not be completely dismissed.

An attacker could use this attack to crash or potentially execute arbitrary software on the AV-TSX. Should the latter be achieved, the integrity of the election would become questionable.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through source code analysis.

13.3.19 Ballots are stored in the order in which they are cast

When traditional paper ballots are placed in a ballot box, the order in which they were cast is not recreatable. This protects voter privacy. By storing ballots in the order in which they are cast, the AV-TSX potentially allows a specific voter's choices to be identified by an attacker.

Description: When a user casts their ballot, the AV-TSX records votes as records in a `.brs` file. Each result is individually encrypted using AES-128 cipher in CBC mode and the Data Key. Before being encrypted, each ballot is marked with a unique serial number (see Issue 13.3.21), the serial number of the machine on which the ballot was cast and a time stamp. Ballots are then added to the end of the `.brs` file.

The problem with this approach is that a total ordering of the voters using a specific machine is kept. This, in combination with a time stamp identifying when a specific vote was cast, allow an adversary with access to a card the potential to determine how specific ballots were cast. Note that the use of encryption does little to protect the voter in this case due to vulnerabilities such as those discussed in Issue 13.3.5 (e.g., weak, default or poorly protected keys).

This confirms the vulnerability (5.2.19) reported in the California Source Code Report.

Prerequisites: An attacker with access to the AV-TSX who happened to watch when a particular voter cast their ballot could successfully exploit this vulnerability.

Impact: Privacy is critical to the election process. If an attacker can learn the result of a specific ballot, they can threaten individuals (i.e., "voter coercion") to sway election results.

Procedural Mitigations: The machine used by a particular voter must be kept secret from observers.

Verification: This vulnerability was confirmed through source code analysis.

13.3.20 Cast ballots and VVPAT barcodes contain a timestamp

Printed versions of ballots cast on the AV-TSX can contain a timestamp encoded in the barcode. An attacker can therefore determine how a specific voter cast their ballot.

Description: As previously mentioned (Issue 13.3.19), the record of each ballot contains the time at which the ballot was cast. Such information offers an attacker the potential to violate voter privacy by matching a specific ballot to an individual. An adversary with access to the poll book or ExpressPoll and the Data Key (Issue 13.3.5) would be able to match specific ballots to individuals with high probability.

An adversary simply observing the time at which a voter cast their ballot may be able to match this information much more easily. Because the timestamp information is also encoded in a certain type of barcode optionally printed at the bottom of each VVPAT record, an adversary could have an exact matching between individuals and their votes. Whether or not a barcode contains timestamp information depends on the type

of barcode used. If a traditional/1-D (CODE-128) barcode is used, the timestamp is not included. If instead a 2-D (PDF-417) barcode is used, the timestamp information is included.

This confirms the vulnerability (5.2.20) reported in the California Source Code Report.

Prerequisites: An attacker would need to have access to the poll book/ExpressPoll and the Data Key or the VVPAT records with 2-D barcodes to exploit this vulnerability. The easily breakable nature of VVPAT enclosure (Issue 14.8.2) would allow an attacker to accomplish the latter in a matter of seconds and leave little recourse to the polling center.

Impact: Privacy is critical to the election process. If an attacker can learn the result of a specific ballot, they can threaten individuals (i.e., "voter coercion") to sway election results.

Procedural Mitigations: No procedural fixes are known to fix the problem of individual electronic ballots having timestamps. Only traditional barcodes, which do not include the timestamp, should therefore be used.

Verification: This vulnerability was confirmed through source code analysis.

13.3.21 An attacker can reconstruct the order in which ballots were cast

Ballots stored in the memory of the AV-TSX contain a "random" serial number. However, the process by which this serial number is created is not random. Accordingly, an attacker can use this number to determine the order in which votes were cast.

Description: As a means of preserving voter privacy, each ballot is marked with a serial number generated through a cryptographic pseudo-random function. The resulting serial number on each ballot therefore appears to be random to the casual observer. According to comments in the source code, this function is a 20-bit random permutation generator based on a Fiestel network using AES as the round function. The programmer(s) note that this algorithm is described in Bruce Schneier's book *Applied Cryptography*.

The security of this algorithm and the resulting ballot serial number relies upon the key remaining secret. However, the manner by which this key is generated makes it easily reconstructable by an attacker. The key generation function takes the Data Key and the *BRS Signature* as input. As mentioned in Issue 13.3.6, the signature for a .brs file is created by using the AES-128 algorithm and the Data Key to encrypt the result of the MD5 hash algorithm over the header fields. Unfortunately, the fields in the BRS signature, which include values such as a timestamp, the voting machine's serial number and the election mode, are all easily predictable. Many of these values are displayed by the voting terminal screen when it is turned on. An adversary capable of recovering the Data Key (see Issue 13.3.5) can therefore easily determine the order in which ballots were cast.

This confirms the vulnerability (5.2.21) reported in the California Source Code Report.

Prerequisites: A voter or poll worker able to recover the Data Key could easily exploit this vulnerability.

Impact: Privacy is critical to the election process. If an attacker can learn the result of a specific ballot, they can threaten individuals (i.e., "voter coercion") to sway election results.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through source code analysis.

13.3.22 The AV-TSX does not securely delete files

The AV-TSX does not overwrite the contents of files it deletes. Rather, it deletes them by considering the area they occupy unused. Commodity software can allow an attacker to examine the contents of these supposedly unused areas and recover data.

Description: At certain points during the election process, it may be necessary to delete files that are no longer useful. For example, election officials may chose to delete backup files and audit logs from previous elections to free space. Because election databases contain information that might allow an attacker to link a ballot to a specific voter (Issue 13.3.19), it is necessary to ensure that these files can not be recovered by an adversary.

Throughout the code, the developers use the `DeleteFile()` API call. `DeleteFile()` is a standard Windows call that simply removes the file from a directory listing. However, the contents of the file are not destroyed. An attacker can use publicly available tools to easily recover such data.

This confirms the vulnerability (5.2.22) reported in the California Source Code Report.

Prerequisites: An attacker would need access to a memory card or AV-TSX.

Impact: Privacy is critical to the election process. If an attacker can learn the result of a specific ballot, they can threaten individuals (i.e., "voter coercion") to sway election results.

Procedural Mitigations: Memory cards should be securely deleted before and after each election.

Verification: This vulnerability was confirmed through source code analysis.

13.3.23 Logic errors in the bootloader create a vulnerability when loading bitmaps

Some of the code contains logical errors. Checks that are intended to protect the software can therefore never do so.

Description: An error in the bounds checking logic when loading images during the boot sequence creates a buffer overflow vulnerability. The code is shown below:[27]

```
253 void GlibPutPixel(UINT xx, UINT yy, Pixel_t Color)
254 {
255    // Check for library not initialized or (x,y) out of range.
256    if (FrameBuffer != FALSE || (xx < USER_X) || (yy < USER_Y))
257     {
258             // Compute the frame buffer offset and write the pixel.
259             FrameBuffer[FB_OFFSET(xx,yy)] = Color;
260     }
261 }
```

On line 256, the program checks that the pixel to be drawn is within the boundaries of the buffer. However, the developer has used the *or* ("||") operator instead of the *and* ("&&") operator. Accordingly, the above code does not test whether the data to be written is within the bounds of the buffer. Rather, once confirms that FrameBuffer exists, it copies the information to that buffer. Bounds checking therefore never occurs.

[27]The following code was first published verbatim in the public release of the California TTBR.

This confirms the vulnerability (5.2.23) reported in the California Source Code Report.

Prerequisites: An attacker would need to have access to the bootloader to insert a specially crafted bitmap image in order to exploit this vulnerability.

Impact: The impact of this vulnerability is admittedly minimal. If an attacker had access to the bootloader, they could simply load malicious code directly instead of creating a special image file. However, this weakness points to a larger weakness in the software engineering practice. Such an error should have been caught in code review.

Procedural Mitigations: None.

Verification: This vulnerability was verified through source code analysis.

13.3.24 There are a number of errors in the AV-TSX startup code

A number of programming mistakes can be found in the code. For instance, instead of allocating space for a series of characters, the software only allocates space for one. This can potentially be exploited by an attacker.

Description:

```
287 TCHAR name;
288 _stprintf(&name, _T("\\Storage Card\\%s"), findData.cFileName);
```

The code[28] above is an example of a dangerous error found in the AV-TSX. `TCHAR name` represents a single character. However, the call to `_stprintf` on the next line of code writes to `name` as if it were and array or collection of multiple characters. Accordingly, even benign files loaded by the bootloader have the potential to cause the AV-TSX to crash. Such a buffer overflow vulnerability would also potentially allow an adversary to load and execute arbitrary, malicious software on the AV-TSX.

This confirms the vulnerability (5.2.24) reported in the California Source Code Report.

Prerequisites: An attacker would need to have access to the AV-TSX and a PCMCIA memory card.

Impact: An attacker could crash or take control of an AV-TSX by loading malicious software.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through source code analysis.

[28]The following code was first published verbatim in the public release of the California TTBR.

CHAPTER 14

PREMIER NEW ISSUES

This chapter describes vulnerabilities we found in the Premier election system, beyond those discovered in the California TTB Source Code Report for Diebold. The public version of this report contains references to sections of the private report. These non-public sections contain confidential information identifying where in the source code or design the issue is manifest, or other vendor-confidential information. Note that all source code, file names, and other apparent vendor specific information in the public version have appeared in previous election study analysis reports.

14.1 Premier EMP Vulnerabilities

14.1.1 Tampering with a memory card allows attackers to crash or take control of an EMP server.

The EMP server trusts that the information stored in unencrypted/unauthenticated files is benign. Anyone with access to a memory card can create malicious input and use it to crash or gain control over the EMP server.

Description: Originally, multiple AV-TSX units would be set up inside the county election headquarters to accomplish the pre- and post-election tasks of programming and collecting results from memory cards. The *Election Media Processor* (EMP) was designed to expedite these tasks. Consisting of array of up to six PCMCIA card slots, the EMP allows an election administrator to more quickly move ballot definitions from and election results to the GEMS server.

Much of the memory card related source code used in the AV-TSX is also present in the EMP software. So too are many of the vulnerabilities. For instance, Issue 13.3.14, which notes that the PCMCIA card reading function assumes that the contents of `assuri.ini` do not exceed a 1024 character buffer, is present in the EMP software. If an adversary changes the contents of `assure.ini`, the integrity of which are not protected, they can cause a buffer overflow in the EMP. This could be used to potentially change votes on the EMP directly or to use the EMP as a platform to directly attack the GEMS server.

Moreover, this attack could easily be accomplished. An adversary could create their own PCMCIA card (including the identifying AccuVote stickers) and leave it at a polling station. This card would likely make its way back to the EMP where, on insertion, could take control of the election. The infected EMP terminal could then erase the exploit from the memory card to hide its presence from future analysis. Alternatively, an attacker could launch the same attack prior to an election and then infect all memory cards destined for

AV-TSX machines.

The team was able to demonstrate not only that this weakness existed in the source code, but also that it was exploitable. The exploit occurs immediately after a memory card is inserted, leaving no time for an operator to clear the contents of the memory card.

Prerequisites: An attacker would need to have access to any memory card (legitimate or otherwise). This card would need to be read by the EMP.

Impact: The attacker could potentially execute arbitrary code on the EMP. This is more critical than Issue 13.3.14 as the EMP is a central point in the system. Accordingly, the EMP could be used to rapidly distribute a virus.

Moreover, the EMP could be used to infiltrate the GEMS server. Because the two are connected via an Ethernet connection, the EMP could attempt to exploit known vulnerabilities in both the GEMS software and the underlying operating system.

Procedural Mitigations: Chain of custody logs are a potential *partial* mitigation to protect memory cards. For practical limitations, see Section 3.5.

Verification: This vulnerability was confirmed through source code analysis and penetration testing.

14.1.2 A single Data Key is used to encrypt all ballots in a county

The EMP server presents a screen with a single field for the Data Key used to encrypt the results of an election. Because all cards are protected with this key, a single compromised voting machine makes the results on all other memory cards vulnerable to tampering.

Description: As mentioned in Issue 13.3.6, the contents of Ballot Results (`.brs`) files are encrypted on the AV-TSX using AES-128 in CBC mode with the *Data Key*. The Data Key in each machine should be unique. If such a practice were followed, an attacker able to recover the Data Key from one machine would not help them to attack any other machine.

Previous investigations have noted that it is likely that the same data key is used to encrypt all ballots across a county. An examination of the code on the EMP confirms this suspicion. When the header of the `.brs` file is read by the EMP, it checks to see whether the hash of the data key in the header matches that one stored in the machine. The only way two such hash values could match would be if the Data Key stored on each unit was the same. If the key hash values do not match, loading fails.

Testing on the EMP server confirms this issue - only one Data Key can be loaded onto the machine.

This vulnerability has not previously been confirmed. California Issue 5.2.5 notes that "all machines within a particular county most likely share the same ... Data Key."

Prerequisites: An attacker would need to compromise one unit using the Data Key (AV-TSX, EMP).

Impact: An attacker could forge or change results for all precincts in a county with access to a single machine.

Procedural Mitigations: The operator of the EMP should manually enter a unique Data Key for each card that is encoded. This would prevent parallel encoding of cards.

Verification: This vulnerability was confirmed through source code analysis and penetration testing.

14.1.3 The warning that a default key is in use is not sufficient

The EMP server is loaded with a publicly known default Data Key. The warning that this key is in use may not be immediately obvious to all users.

Description: Before an election begins, the EMP user enters the Data Key into a setup menu. The contents of this field (`Data Key:`) are protected using standard password measures (i.e., each character is represented by a "*" and the contents of the field can not be copied and pasted to another location). As a means of checking to see that the correct key is entered, a second field (`Hash:`) containing the hash of the Data Key is displayed below.

Should the user enter the default Data Key for encrypting ballots into this field, the background of the Hash field turns pink (`RGB: 255,191,191`). Other key values entered into the Data Key field should turn the background of the Hash field green (`RGB: 34, 255, 145`). There is no other indication that the default key is being used. The meaning of the background color is explained on page 15 of the EMP 4.6 Users Guide, Revision 2.0 (Section 2.1.16 - Security Tab).

The mechanism is problematic from a number of perspectives. First, one of the members of the team is red/green color-blind. Accordingly, this team member had great difficulty determining whether or not the default key was in use. Secondly, the mechanism that compares the password entered by the EMP user to the default password only checks the first half of the entered password. The size entered for the `memcmp`, (8 bytes), is half the length of the key.

This mechanism fails to ensure that a large portion of the default key was not used in the Data Key. The first eight bytes can be random and the second eight bytes mirror the default key without turning the background of the Hash field pink. If one would argue that the key from the above case is weak because at least half matches the default value, the mechanism would fail to catch this mirror case (which would be equally as weak).

Moreover, the Hash field only displays the first 4 bytes of the 16 byte MD5 hash. An attacker could quickly generate an equivalent hash value. Because the hash value of the fake key would match that of the expected key, the operator of the EMP server would have no means of knowing the wrong Data Key was being used.

Finally, the concept of a default key is dangerous. At no time should any county rely upon the default key hardcoded into any electronic election system. The default key to the Premier system has long been readily available on the Internet.

Prerequisites: An EMP user not familiar with or able to determine that the background color has a specific security meaning would be susceptible to this weakness. Weak keys could also be generated through this mechanism.

Impact: This weakness would allow weak (or known) Data Keys to be used in an election. If an attacker can find a key with the same hash value for the first 4 bytes, they may cause unreadable memory cards to be distributed to precincts (See Issue 14.1.8).

Procedural Mitigations: The Data Key must be changed upon receipt of the EMP server. Default keys must never be used.

Verification: This vulnerability was confirmed through source code analysis and penetration testing.

14.1.4 A malformed IP address can freeze the EMP server

By entering a malformed IP address, anyone with access to the EMP server may cause the server to become unusable. This problem exists across reboots and reinstallations of the EMP server.

Description: As part of its setup, the EMP software requires that a user enter the IP address or host name of the GEMS server. This allows the EMP server to connect to GEMS when performing its uploading and downloading duties. Should this value need to be changed, the EMP software allows a user to reflect these changes in a field in the Communications Setup menu.

In theory, if the EMP user accidentally enters a malformed IP address or hostname, they should be able to use the Communications Setup menu to correct their error. However, the user may never be given such an opportunity. On startup, the EMP server immediately becomes unresponsive and fails to correctly render the user interface. Because none of the menus have yet been rendered on the screen, the EMP user is never given the opportunity to change this setting back to a correct value.

The value for the GEMS server is stored as an entry in the registry. Because the EMP user is given minimal rights (which is a good security practice), they can not edit the registry to fix this problem. Moreover, unless the administrator understands that the host string is stored in the registry, it is unlikely that they will be able to fix the problem. Because the uninstall program included with the EMP software fails to remove these entries from the registry, this problem persists across reinstallations.

Finally, there is no error checking when a new value is entered into the IP address/host name field. The EMP server always trusts that the user correctly entered the data.

Because of the time limitations of this study, the exact cause of this vulnerability was not located. Please see the Private Appendix 25.1.4 for more details.

Prerequisites: An EMP user, malicious or otherwise, would need to be seated at the terminal.

Impact: The EMP server would be rendered temporarily useless. Because rebooting the machine would have no effect, this attack may make it difficult for memory cards to be delivered to districts on time or for votes to be tallied efficiently.

It is also potentially dangerous that a user can enter unchecked values into the registry. This may be useful as part of another attack.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through source code analysis and penetration testing.

14.1.5 The contents of the logs on the EMP server are not authenticated

Log files offer officials an audit trail after an election has concluded. However, there is no protection of the log files on the EMP server. These audit trails can therefore easily be forged.

Description: The EMP server keeps logs of many of the operations it performs. For instance, when a blank memory card is inserted and a new ballot definition downloaded, the EMP server creates a log entry. Logging also occurs when cast ballots are uploaded to the GEMS server or when an error (e.g., connection timeout) occurs. These logs provide evidence with which an auditor can reconstruct the events on an election. Similar audit logs exist on the AV-TSX.

Unlike the logs on the AV-TSX, however, the integrity of the EMP logs is not protected. Accordingly, these logs can be altered or forged by an attacker without being detected. These files can also be deleted without notice. The EMP logs therefore have limited forensic value.

Prerequisites: An attacker would need access to the EMP server. Malicious software could easily provide such access. Moreover, if such logs were moved to another machine for analysis, they could be changed by any user there or during transport.

Impact: An attacker can delete evidence of their attack without detection.

Procedural Mitigations: Ensure that all operations performed on the EMP server are observed by multiple parties.

Verification: This vulnerability was confirmed through source code analysis and penetration testing.

14.1.6 The EMP server shares the same default SSL Certificate as the AV-TSX

Communication between the networked devices can be achieved securely if SSL is used properly. However, the same cryptographic keys/certificate stored in every AV-TSX is also stored on the EMP server. An attacker can use this information to communicate with the GEMS server.

Description: The EMP server communicates over a network with the GEMS server. Because the contents of their communication may be sensitive, the link between the two is secured using strong cryptographic mechanisms. OpenSSL provides both encrypted and authenticated communication that matches the recommended best practices for data secrecy and integrity.

As was noted in Issue 13.3.11, it is highly likely that all AV-TSX units in the country contain the same certificate (which has a public/private key pair). Moreover, there does not appear to be any simple mechanism through which a county/voting district can change these certificates. Accordingly, the compromise of a single AV-TSX would allow an attacker to intercept, decrypt and change the contents of communication between any AV-TSX and GEMS server. This represents a misuse of the cryptographic protections.

Like the AV-TSX, the EMP server also stores a certificate allowing secure communication in the file `client.pem`. Unfortunately, the contents of this certificate and the certificate used on all AV-TSXs are exactly the same[1]. Accordingly, communications between any AV-TSX OR EMP server and GEMS can be intercepted when a single machine is compromised.

This has particular importance in Ohio, where state law prohibits the use of a modem connection between an AV-TSX and the GEMS server. In spite of this law, an attacker compromising a voting machine can use the information recovered in the attack to launch a second attack on GEMS. Because the GEMS and EMP servers may be run on a semi-public network (e.g., the same network used for desktop machines in the county election headquarters), this weakness may allow a remote attacker to spoof voting results.

As was also reported in Issue 13.3.11, the password "diebold" protecting `client.pem` is weak.

Prerequisites: An attacker would need to compromise any AV-TSX or EMP in order to recover the certificate.

Impact: The attacker could potentially attack the GEMS server by pretending to be an EMP server. The fake EMP server could upload forged or altered voting results, thereby compromising the integrity of the

[1] Running `diff`, which compares the contents of two files and reports the differences, returns absolutely no difference between the two files.

election.

Procedural Mitigations: An AV-TSX can never be attached to a network. This includes overtly by poll workers or covertly by an adversary. Physically removing the modem may provide a straightforward approach to achieving this. The certificates in each machine should also be unique.

Verification: This vulnerability was confirmed through source code analysis and penetration testing.

14.1.7 The System Key for the EMP is insecure

The System Key, which protects the other cryptographic keys stored on the EMP, is the same across all machines. More critically, the key is constant.

Description: Like the AV-TSX, the EMP server stores copies of the Data Key in the file `bs-security.cf`. In order to protect these keys, the EMP server encrypts the contents of this file using its System Key. However, the System Key itself is the same on every machine onto which the EMP server is installed. An attacker capable of compromising one EMP server can then easily recover the keys used to protect the AV-TSX from any EMP server.

Like the System Key in the AV-TSX, the System Key for the EMP server is created in a predictable manner. The machine's serial number is fed as input to the MD5 hash algorithm, the deterministic result of which becomes the System Key. On each AV-TSX, this serial number (and therefore the resulting System Key) is unique. However, the serial number used is a fixed value on all machines: 0.

Like the AV-TSX, the EMP server uses a macros to store this value to the registry. However, upon inspection of the registry on machines running the EMP server, the serial number is never actually added. Accordingly, calls to the registry for this value always return NULL, which corresponds to 0. We can not say whether this was intended in the design of the of the EMP or if it was an error. Regardless, this represents a significant threat to security as anyone with access to the file `bs-security.cf` on any machine running the EMP server can subsequently gain access to election results.

Prerequisites: An attacker would need access to the EMP server.

Impact: The attacker knows the System Key. This key could then be used to decrypt `bs-security.cf` to recover the Data Key used in all AV-TSX machines in a county.

Procedural Mitigations: The EMP server must be as protected as the GEMS server.

Verification: This vulnerability was confirmed through source code analysis and penetration testing.

14.1.8 The integrity of the Data Key is not protected

A lack of integrity checking in the EMP server allows an attacker to change the Data Key used to encrypt election results. No alarms are raised when this value is changed until memory cards are placed in an AV-TSX.

Description: The Data Key is encrypted using AES-128 in CBC mode before being stored in `bs-security.cf`. When the EMP server requires the use of these files, it generates its System Key by running the MD5 hash algorithm on its serial number. The contents of `bs-security.cf` are then decrypted and used by the EMP server for their appropriate purposes.

To ensure that the key values correctly decrypted, the EMP server checks the value stored in the 8-byte `id` field. If `id` contains a fixed string, the values assigned with each of the keys are assumed to be correct. However, an attacker can change any part of `bs-security.cf` beyond the first 8-bytes and cause the EMP server to load the incorrect keys.

Assuming that an unknown key were used to encrypt the contents of `bs-security.cf`, an attacker would not be able to change the key to a known value. Rather, a single change in the encrypted contents of the file would lead to unpredictable changes throughout the actual key. For instance, if the attacker changed the contents of previous fields the Data Key would also change. If the attacker instead only changed the encrypted characters corresponding to the Data Key (the last value in the file), only the Data Key would be affected.

The value of this attack would not be in programming a specific key. Rather, by inserting a random key, the attacker would be able to prevent an election for occuring. By encoding all of the memory cards with a random key, none of the AV-TSX units would be able to load the ballot definitions. [2]

Prerequisites: An attacker would need access to the `bs-security.cf` file on the EMP server.

Impact: An attacker may delay the running of an election as all memory cards encoded with the incorrect data key would be unusable until properly rewritten.

Procedural Mitigations: The hash value displayed on the Security Setup menu should also be monitored constantly (see Issue 14.1.3 for problems with the displayed hash value).

Verification: This vulnerability was confirmed through source code analysis and penetration testing.

14.1.9 GEMS trusts the EMP to deliver correct ballot definitions and results

GEMS delivers ballot definitions and receives election results from the EMP server. However, there is no need for the EMP server to have unencrypted access to either. Because the EMP server is less protected than the GEMS server (no firewall, virus scanner, etc), it becomes a much easier target.

Description: The GEMS server relies upon the EMP server to perform a number of tasks. Before an election occurs, the EMP encodes memory cards with the assorted files necessary to run an election. After the close of an election, the EMP will tallies and packages results in a format read by the GEMS server. In order to perform these operations, the EMP server is given access to the Data Key. Accordingly, the EMP can decrypt all of the votes cast on AV-TSX machines.

The major difficulty with this model is that a malicious EMP server could easily switch, forge or drop votes. Because the same Data Key is known by the EMP and all AV-TSX devices and the serial number of the AV-TSX associated with each vote is included in the header of the results file (see Issue 13.3.5), the EMP server can therefore perform all of the operations associated with any AV-TSX and use the correct cryptographic keys to "validate" the results. There would be no means of distinguishing where a vote was written. There is therefore no reason for the GEMS server to believe the accuracy of the results reported from the EMP server.

The option to not use of SSL between the GEMS and EMP servers is additionally troublesome. Even if the chain of custody between all precincts and the EMP server is sufficient to protect the memory cards, an adversary might be able to eavesdrop and change election results if SSL is not enabled. Because the EMP

[2] When the encrypted Data Key is tampered with, the AV-TSX complains, "Unable to load the election: the active ballot box is encrypted with the wrong media key."

server removed the encryption protecting the results, the attacker would not need to recover the Data key. Trusting the EMP with the Data Key is therefore an extremely dangerous architectural decision.

Prerequisites: An attacker would need to gain access to the EMP server. This could be achieved by the EMP administrator or remotely by an adversary on the same network using common hacking tools such as Metasploit.

Impact: An attacker would be able to completely control the results of an election.

Procedural Mitigations: SSL must always be used in communication between the EMP and GEMS servers. Additionally, the EMP server should be hardened against compromise to a level at least equal to GEMS.

Verification: This vulnerability was confirmed through source code analysis and penetration testing.

14.1.10 A malicious GEMS server can crash an EMP server

The EMP server trusts that all inputs from the GEMS server are well-formed. Because of a lack of input checking in a number of places, the EMP server is vulnerable to attacks that can crash or gain control of the software.

Description: The EMP server trusts that the election data downloaded from the GEMS server is both well-formed and benign. Accordingly, the EMP becomes vulnerable to attack if the GEMS server or an adversary claiming to be the GEMS server is able to send malformed data. Specifically, the EMP server is vulnerable to a number of format string vulnerabilities. These vulnerabilities occur when the software treats strings of text in an unsafe manner. In this particular example, a malicious administrator can append `printf` format specifiers (e.g., values such as "%s") to the name of the Voting Center and cause the program to crash.

This particular vulnerability has been confirmed to crash the server on ballot download to the EMP server. Such vulnerabilities may also allow an attacker to take control of the EMP server; however, we did not have time to create such an exploit.

Prerequisites: An attacker would need to gain access to the GEMS server. If the attacker could capture the Data Key (See Issue 13.3.5), they would be able to modify the necessary field directly on the memory card.

Impact: An attacker could crash or potentially take control of the EMP server. In so doing, they may be able to delay or alter the results of an election or spread a virus.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through source code analysis and penetration testing.

14.1.11 A compromised AV-TSX can crash an EMP server

The EMP server trusts many of the inputs offered by the AV-TSX. A malicious AV-TSX can take advantage of this trust to launch attacks capable of crashing or gaining control over an EMP server.

Description: The EMP server also trusts that the information sent from the AV-TSX is well-formed and benign. The EMP server is therefore vulnerable to attack by an attacker that has compromised an AV-TSX. As was mentioned in Issue 14.1.10, the EMP server is vulnerable to a number of format string vulnerabilities. These vulnerabilities occur when the software treats strings of text in an unsafe manner. In this particular example, a malicious AV-TSX can add `printf` format specifiers (e.g., values such as "%s") to the name of

the Voting Center and cause the program to crash.

This particular vulnerability has been confirmed to crash the EMP server on when a card containing such a modification is inserted. Such vulnerabilities may also allow an attacker to take control of the EMP server; however, we did not have time to create such an exploit.

Prerequisites: An attacker would need to gain control of a single AV-TSX. If the attacker could capture the Data Key (See Issue 13.3.5), they would be able to modify the necessary field directly on the memory card.

Impact: An attacker could crash or potentially take control of the EMP server. In so doing, they may be able to delay or alter the results of an election or spread a virus.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through source code analysis and penetration testing.

14.2 Premier General Vulnerabilities

14.2.1 Premier may follow insecure media format procedures

When memory cards are returned to Premier, the "erase" procedure may not entirely remove the contents. This issue may allow an attacker from another state to gain sensitive information about citizens to be used for identity theft.

Description: We received a number of media cards for our security analysis of the Premier system. All cards appeared to be blank; however, upon further investigation, we discovered that memory cards had merely undergone a Windows "Quick Format." For one of the ExpressPoll compact flash cards, we were able to recover the polling data for a county in another state. We confirmed that a few names and addresses are valid citizens of the county using public online resources. This leads us to believe that memory cards recirculated back to Premier may not be securely formatted. However, this instance may be an isolated occurrence.

Prerequisites: Access to a new memory card from Premier.

Impact: The ExpressPoll data contains potentially private information such as name, address, race, gender, and date of birth. Such information could be used for identify theft.

Procedural Mitigations: All memory cards should be secure erased *before* returning them to Premier.

Verification: This issue was confirmed via demonstration with provided memory cards.

14.3 Premier GEMS Vulnerabilities

14.3.1 Vulnerabilities in Microsoft Windows provided DLL files affect GEMS

GEMS depends on Microsoft Windows libraries (known as DLL files) in various aspects, such as making windows and accessing the election database. If there are vulnerabilities in this COTS software, GEMS is also affected. For example, a recent vulnerability in the Microsoft DLL used to access election databases could allow an attacker to create a an election database that infects GEMS with a virus.

Description: A recent vulnerability of the Jet Data layer targeted towards Microsoft Access[3] could also affect GEMS. While publicly available malicious `.mdb` files are designed to exploit Microsoft Access, a file could just have easily be crafted to exploit GEMS. We have not crafted such a file; however, we have confirmed that the malicious file designed to exploit Access crashes GEMS.

Note that while this vulnerability was made public only after our study began, it is a stark reminder that vulnerabilities are continually discovered in operating systems and supporting libraries. The election software implicitly trusts the operating system and libraries to function correctly.

Prerequisites: Ability to cause GEMS to open the malicious `.mdb` file.

Impact: For the case of the Microsoft Jet vulnerability, anyone who has access to modify a `.mdb` file can control an election.

Procedural Mitigations: None.

Verification: This issue was confirmed via demonstration with the GEMS server.

14.3.2 Many GEMS servers state-wide use the same BIOS password

Premier maintains a support contract with many of the Ohio counties using Premier equipment. Most counties use the same BIOS password. Knowledge of the BIOS password allows an attacker to boot from external media to subvert the system.

Description: Premier maintains a support contract with many of the Ohio counties using Premier equipment. 40 of the 48 GEMS servers deployed have the same BIOS password. An attacker discovering the BIOS password in one county has a high probability of gaining access BIOS access to the GEMS server in another county. Knowledge of the BIOS password enables an attacker to boot from external media to subvert the system (see Issue 14.7.3).

Prerequisites: Access to the GEMS BIOS password in one county.

Impact: The GEMS server in another county can be compromised.

Procedural Mitigations: Each GEMS server should have a different BIOS password.

Verification: This issue was confirmed through personal correspondence with Ohio Secretary Of State Office employees.[4].

14.4 Premier AV-OS PC Vulnerabilities

14.4.1 Forged ballots can be forced into the ballot box

When the AV-OS PC scanner detects jammed paper, the roller motors loosen their grasp and the ballot can be pushed into the ballot box. While the vote is not recorded, an attacker can fill the ballot box with forged ballots that may be accepted as official upon recount.

[3] *Microsoft JET – Remote hacker manual control*. ⟨URL: `http://www.beskerming.com/security/2007/11/17/73`⟩.

[4] *Personal correspondence with Ohio Secretary of State Office employee*. December 4, 2007.

Description: The AV-OS PC scanner senses when a ballot is not feeding properly through the reader. In order to avoid motor burnout and allow a paper jam to be fixed, the rollers disengage, allowing paper to be unjammed. An attacker can cause the AV-OS to detect a jammed ballot by simply firmly holding on to a ballot. Once the rollers disengage, the ballot can be manually fed through the scanner. As all ballots are printed on card stock, the another ballot is strong enough to push the illegitimate ballot into the ballot box. We were unable to find any reference to the jammed scanner in the audit report.

Prerequisites: An attacker requires access to the AV-OS PC and ballot box while it locked by physical key. Note that the motors are disengaged while the AV-OS PC is powered off. Finally, the ballot box contains a lockable panel to seal the ballot box when the AV-OS PC is not in place. If this panel is in place, this vulnerability cannot be used to force ballots into the ballot box.

Impact: An unsupervised attacker with a few seconds access to a locked AV-OS PC and ballot box can feed forged ballots into the ballot box which may be accepted as official upon recount. Note that ballots must be fed in one at a time, therefore the required time is directly related to the number of ballots forced into the ballot box.

Procedural Mitigations: The ballot feed panel should always be locked when the ballot box is not supervised. Note that simply locking the AV-OS PC in place is not sufficient. This is a *partial* mitigation, as locks may be picked.

Verification: This issue was confirmed via demonstration with the AV-OS PC and ballot box.

14.4.2 The AV-OS PC ballot box collecting votes allows vote order to be reconstructed

The AV-OS PC ballot box sorts ballots based on the existence of a write-in candidate; one bin is kept for each. Within these bins, all ballots lay in the order they were cast. Thus, specific voters may be targeted by noting the order in which they vote and examining the pile of ballots at close of election.

Description: During election day, the AV-OS PC is placed in a special ballot box. The box contains a number of physically locked compartments to keep cast ballots sealed inside. This box also contains a data port that allows the AV-OS PC to control a motor and plastic flap that sorts ballots into one of two bins depending on the existence of a write-in candidate.

When a ballot is cast, it falls into the designated bin. We examined the ballot box and discovered that there are no mechanisms within to inhibit or otherwise change the order in which ballots fall. As a result, at the close of an election, many ballots will be cast at the bottom of the non-write-in bin in the same order they were cast.

We numbered and cast 10 ballots through the AV-OS PC. Upon checking the ballot box, we examined the pile and found that the order was maintained. A watchful adversary who had access to the ballots after they had been cast would be able to report, with non-negligible probability, the way that a given voter had cast their ballot if they knew the order in which that vote had been cast.

Prerequisites: An attacker would need to watch the order in which votes were cast at the AV-OS PC, then be able to access the pile of ballots at the end of the election. A malicious poll worker, for example, could determine the order of a cast ballot and make note of the way in which the voter had voted. The probability of identifying a specific voter increases as the number of ballots containing write-in votes decreases.

Impact: This allows a compromise in voter privacy, as an attacker could determine the way a vote was cast. This attack could be used for the purposes of vote-buying or coercion.

Figure 14.1: The same key works in both the Hart (left) and Premier (right) ballot boxes.

Procedural Mitigations: The election administrator should shuffle the piles of ballots in the ballot box at the close of the election. This will have limited effect if the administrator or any other workers observing the pre-shuffled state of the ballots is corrupt.

Verification: This issue was confirmed via demonstration with the AV-OS PC and ballot box.

14.4.3 The Hart ballot box key works in the Premier ballot box

While the Premier and Hart ballot boxes are significantly different, the keys we received work in both. If what we observed represents a larger trend, an attacker gaining access to the ballot box key in one county can compromise other ballot boxes within the state, regardless of the vendor.

Description: Both Premier and Hart InterCivic have precinct optical scanners using large ballot boxes on top of which a small scanner device is secured. The ballot box appears to be custom made for each vendor and, in the case of Premier, performs special functionality (sorting write-in ballots from non-write-in ballots).

We received keys for the ballot boxes from the Secretary of State and discovered that they are interchangeable between the two boxes. Furthermore, each ballot box has multiple locks, all of which are accessed by the same key. Figure 14.1 shows the same key working in both ballot boxes. If what we observed represents a larger trend, an attacker gaining access to the ballot box key in one county can compromise other ballot boxes within the state, regardless of the vendor.

Prerequisites: If the ballot boxes use the same key, an attacker requires access to the ballot box key in just one county.

Impact: Ballot boxes in other counties within the state, regardless of the vendor, may compromisable.

Procedural Mitigations: Each ballot box should have unique keys.

Verification: This issue was confirmed via demonstration with the ballot box.

14.5 Premier VCEncoder Vulnerabilities

14.5.1 Access to the Voter Card Encoder is not protected by a PIN

With access to a Supervisor Card, a Voter Card Encoder can be enable to create legitimate Voter Smart Cards. Similar actions on the AV-TSX require a PIN. Such an operation should be additionally protected with a PIN to provide protection against stolen cards.

Description: The Voter Card Encoder allows poll workers to create valid Voter Smart Cards during an election. Protection for this device is provided in the form of the Supervisor Smart Card. This card, to which access should be limited, also allows administrative operations to be conducted on AV-TSX units. Furthermore, all supervisor actions performed on the AV-TSX require a PIN.

Even the best procedures may occasionally fail to protect the Supervisor Card. In such an event, there is no protection against an attacker activating a Voter Card Encoder and creating an unlimited number of legitimate voter cards.

Prerequisites: An attacker would need to gain access to a Supervisor Smart Card and a Voter Card Encoder.

Impact: The attacker would be able to create an unlimited number of legitimate voter cards.

Procedural Mitigations: Chain of custody logs are a potential *partial* mitigation to protect Voter Card Encoders. For practical limitations, see Section 3.5.

Verification: This vulnerability was confirmed through penetration testing.

14.5.2 Once the VCE is enabled, no mechanism prevents smart cards from being encoded

Once a Voter Card Encoder is enabled, nothing other than possession prevents it from being used to create legitimate Voter Smart Cards.

Description: When the Voter Card Encoder is enabled using the Supervisor Smart Card, poll workers can use this device to create legitimate Voter Smart Cards. However, after this initial access check, any individual with access to the Voter Card Encoder can create voter cards.

If a poll worker were to place the Voter Card Encoder on a desk, an attacker may "accidentally" drop the contents of their bag on the desk. Without arousing much suspicion, the attacker could then take the Voter Card Encoder into their possession. After leaving the polling place, the attacker could encode legitimate Voter Smart Cards from the parking lot and provide them to colluding adversaries in order to cast extra votes.

Such an attack need not happen on election day. We were able to activate an unmodified Voter Card Encoder, turn the power on and off and were still able to create legitimate voter cards over three weeks later.

Prerequisites: An attacker would need to take an active Voter Card Encoder before or during an election.

Impact: The attacker could create an unlimited number of legitimate Voter Smart Cards.

Procedural Mitigations: The Voter Card Encoder should not be out of the physical control of trusted elections officials/poll workers at any time. Chain of custody logs are a potential *partial* mitigation to protect these devices. For practical limitations, see Section 3.5.

Figure 14.2: A Voter Card Encoder running arbitrary software.

Verification: This vulnerability was confirmed through penetration testing.

14.5.3 Software can be loaded by anyone with access to the VCE

Installing new software on a Voter Card Encoder lacks any authentication mechanisms. Simply by pressing the "Off" button, followed by "Yes" will replace the software stored on the device.

Description: Like the other software components of any system, the Voter Card Encoder may require software updates over the course of its lifetime. In order to facilitate such an operation, the Voter Card Encoder can receive software updates using a 9-pin serial cable attached to a PC. To load new software onto the Voter Card Encoder, a user simply turns the device off. When the user presses the off button again, the Voter Card Encoder asks the user to press "Yes" if they would like new software to be loaded.

The problem with this mechanism is that it lacks any authentication of the new software loaded onto the Voter Card Encoder. As a demonstration of this issue, we created and loaded new software. Where the standard software would require a user to activate the Voter Card Encoder with a Supervisor Smart Card, we allowed any card (even unrecognizable formats) to enable the device. An adversary could therefore steal a Voter Card Encoder, load their own software and then create valid Voter Smart Cards. Figure 14.2 shows an example of such software.

Alternatively, we could have used this vulnerability to encode any type of card we wished. For instance, an attacker could easily create Central Administrator, Security or Supervisor Cards by modifying the software running on the Voter Card Encoder.

Prerequisites: An attacker would need access to a Voter Card Encoder.

Impact: An attacker could create valid Voter Smart Cards without the Voter Card Encoder being enabled by the Supervisor Smart Card.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through penetration testing.

14.5.4 The VCE accepts the default smart card key

When attempting to authenticate an inserted smart Card, the Voter Card Encoder first attempts to see if the card contains the publicly known default password.

Description: The Voter Card Encoder authenticates inserted smart cards as a means of access control. If the inserted smart card does not contain the proper password, the card is rejected. However, like the AV-TSX, the Voter Card Encoder accepts smart cards with the hardcoded default password (`0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08`). Moreover, the Voter Card Encoder checks for this password first, only attempting to see if the password set by a previous Security Smart Card is present after the default check fails.

Because of the obviousness and public nature of this password, at attacker can easily forge the smart cards needed to gain access to the Voter Card Encoder.

Prerequisites: The attacker needs access to the Voter Card Encoder and the ability to write their own smart cards.

Impact: An attacker can gain complete administrative access of the Voter Card Controller and use it to create legitimate (to cast fradulent ballots) and illegitimate (to delay or prevent an election) smart cards.

Procedural Mitigations: Chain of custody logs are a potential *partial* mitigation to protect Voter Card Encoders. For practical limitations, see Section 3.5.

Verification: This vulnerability was confirmed through penetration testing.

14.6 Premier ExpressPoll Vulnerabilities

14.6.1 The ExpressPoll runs a webserver

If an attacker is able to gain control of the ExpressPoll, the existing webserver can be modified to create an undetectable back door for future accesses. We were unable to determine why the ExpressPoll is running a webserver.

Description: The ExpressPoll device replaces paper poll books at the polling place. Because precincts may have multiple ExpressPoll units, each device has a Ethernet NIC, which allows devices to synchronize poll book data, e.g., who has voted. We discovered that the ExpressPoll is running a webserver on port 80 (port 443 was also open; however, it did not respond to a web browser). The server displays a default webpage for the Microsoft Platform Builder. We were unable to determine why the ExpressPoll is running the webserver.

The existence of the webserver poses a significant threat to the Express Poll. Once an attacker gains access to the ExpressPoll, possibly exploiting other vulnerabilities mentioned in this report, the webserver can be patched to provide a back door. More importantly, the attacker can revert attacks that visibly change the system, e.g., Issues 14.6.3 and 14.6.4, and maintain access under the guise of the seemingly legitimate webserver.

Prerequisites: An attacker requires the ability to connect to the Ethernet port on the bottom of the ExpressPoll.

Impact: The existence of the legitimate webserver may hide the existence of a persistent back door created after exploiting other vulnerabilities.

Procedural Mitigations: None.

Verification: This vulnerability was discovered through experimentation with the ExpressPoll.

14.6.2 The ExpressPoll will install an unauthenticated bootloader and operating system

The ExpressPoll will automatically load a new bootloader or operating system if files with the appropriate name are present on a memory card. Because these files are not authenticated, anyone with access to a memory card can place any software they wish on the ExpressPoll.

Description: In order to allow updates to the bootloader and operating system, the ExpressPoll scans all inserted memory cards (both PCMCIA and CF) on boot. If the bootloader finds an update to either itself (`EBOOT.BIN`) or Windows CE (`NK.BIN`), it erases the previous version of the software and loads the new version from the above file(s). The difficulty, however, is that at no time is the source of these files authenticated. Accordingly, any file with these names placed on the memory card by any party will automatically be loaded an executed by the ExpressPoll. This weakness is similar to Issue 13.3.1 on the AV-TSX.

Finally, there is a chance that placing the Windows CE file (`NK.BIN`) will not replace the current operating system, but rather only boot from it. Due to the potentially destructive nature of the testing, we verified that the files are accepted, but did not allow the process to be completed. However, either functionalities, booting as runtime image or direct flashing, offer the same potential. Once booted, the runtime image can flash itself to permanent memory.

Prerequisites: An attacker would require approximately 30 seconds and an ExpressPoll to execute this attack.

Impact: An attacker could gain control of an ExpressPoll and execute arbitrary software.

Procedural Mitigations: Tamper evident seals are a potential *partial* mitigation to denote access to the memory card slot has occurred. For practical limitations, see Section 3.5.

Verification: This issue was confirmed via demonstration with the ExpressPoll.

14.6.3 The ExpressPoll uses an unprotected database for poll data

The ExpressPoll database contains information about all registered voters in an area and creates Voter Access Cards for use in the AV-TSX on election day. An attacker can modify the database to include additional names to allow extra voters in a precinct and/or remove names to prevent voters from voting.

Description: The ExpressPoll uses a single DB3 database file to store all the list of all legitimate voters and their designated precincts. The database file is not limited to just one precinct and can span large regions. The file has no protection to prevent malicious modification. An attacker with access to the database file can append new voter records. On election day, these records will allow illegitimate voters to vote in a precinct. An attacker can also remove voter records, thereby disenfranchising voters from their right to vote.

The database file also includes user and supervisor passwords to access various administrative options within the software. Therefore, an attacker with access to the poll database can obtain or change the user and supervisor passwords. The attacker can even disable the need for a supervisor smartcard. These changes allow an attacker to use the ExpressPoll to manipulate poll data.

Prerequisites: An attacker requires 30 seconds of access to the ExpressPoll memory card.

Impact: An attacker has complete control over who can and cannot vote in an election. An attacker can add voters to a precinct and/or prevent specific voters from voting.

Procedural Mitigations: Tamper evident seals are a potential *partial* mitigation to denote access to the memory card slot has occurred. For practical limitations, see Section 3.5.

Verification: This issue was confirmed via demonstration with the ExpressPoll.

14.6.4 The ExpressPoll uses an unprotected resource file

The ExpressPoll interface is defined by a single resource file contained on the memory card. An attacker with access to the memory card can modify this file to extend the functionality of the ExpressPoll application and install malicious software.

Description: The ExpressPoll software uses a .NET resource file to describe the layout and functionality of options in the user interface. The file's authenticity is not verified, and an attacker with access to the file can manipulate the feature functionality of the user interface. Buttons can be rebound to different functionality, and new buttons can be created. For example, a button can be added to launch `explorer.exe`, thereby giving the attacker full access to the system.

Prerequisites: An attacker requires 30 seconds of access to the ExpressPoll and its memory card. Depending on the attack, access to the memory card may be sufficient.

Impact: The attacker can install malicious software on the ExpressPoll and/or make arbitrary modifications to any software on the system.

Procedural Mitigations: Tamper evident seals are a potential *partial* mitigation to denote access to the memory card slot has occurred. For practical limitations, see Section 3.5.

Verification: This issue was confirmed via demonstration with the ExpressPoll.

14.6.5 The ExpressPoll can be rendered useless in transit

The ExpressPoll can be charged while locked in the carrying case (for the case we were provided). An attacker can connect the case to an incorrect power supply and possibly physically destroy the device.

Description: The ExpressPoll carrying case we were provided contains an external power socket presumably used to keep the batteries charged while in storage. If the power cable connecting the external socket to the ExpressPoll is connected during transit, an attacker can destroy the device by connecting an incorrect power source. Tamper evident seals and locks may prevent election officials from testing the device before election day. Note, the ExpressPoll case may vary by precinct, therefore this attack may not generally apply.

Prerequisites: An attacker must have access to the ExpressPoll while locked in a tamper evident sealed case with the power cable connected.

Impact: Poll workers will not be able to verify voters and make Voter Access Cards until a replacement ExpressPoll arrives.

Procedural Mitigations: Test the ExpressPoll and disconnect the power cable before it leaves the county office.

Verification: This issue was confirmed via demonstration with the ExpressPoll; however, we did not attempt to connect an incorrect power supply, as it would destroy the device.

14.6.6 The ExpressPoll audit logs are not protected

The ExpressPoll audit log is contained in two unprotected files on the memory card. An attacker can modify these files to hide malicious activities.

Description: The ExpressPoll is used to authenticate voters on election day. It logs all activities, including login attempts and modification of voter information, to an unprotected DB3 database file. System exceptions are stored in a separate `.xml` text file, also unprotected. Additionally, if either file is deleted, the ExpressPoll creates a new file without indicating an error to the user.

An attacker with access to the log files could remove all traces of malicious activity. For example, if a malicious poll worker changes a voter's status back to "unvoted," the corresponding log entry can be removed or changed to an otherwise benign event.

Prerequisites: An attacker requires 30 seconds of access to the memory card.

Impact: An attacker could modify or delete log entries to hide malicious activity.

Procedural Mitigations: Tamper evident seals are a potential *partial* mitigation to denote access to the memory card slot has occurred. For practical limitations, see Section 3.5.

Verification: This issue was confirmed via demonstration with the ExpressPoll.

14.6.7 ExpressPoll audit logs violate voter privacy

The ExpressPoll audit log records a voter identifier when a voter is given a Voter Access Card. This identifier corresponds to a field in the voter information database and can be used to determine the order and time in which voters enter the polling place.

Description: The ExpressPoll is used to authenticate voters as they arrive at the polling place. Once authenticated, the voter is given a Voter Access card, and the voter information database (discussed in Issue 14.6.3) is updated to indicate the voter has already entered the polling place. Along with this update, the ExpressPoll appends the activity audit log (discussed in Issue 14.6.6) indicating the `voterId`. The `voterId` field recorded in the audit log matches the field in the voter information database. As the audit log is appended, the order voters enter the polling place is captured with a sequence number, and while a timestamp is not recorded for these entries, other entries, e.g., power-on, include a timestamp. Hence, an attacker can derive approximate times for voter entries.

The memory card we received with the provided ExpressPoll was blank. However, due to the erase method used on the memory card (see Issue 14.2.1), we were able to retrieve a legitimate voter information database. Using this we were able to analyze the audit log and correlate the `voterID`.

Prerequisites: An attacker requires access to the voter information database and the audit log database. For additional time information, the attacker requires the ability to power cycle the ExpressPoll multiple times during the election day.

Impact: An attacker can determine the order in which voters enter the polling place, possibly an approximate time.

Procedural Mitigations: Chain of custody logs are a potential *partial* mitigation. For practical limitations, see Section 3.5.

Verification: This issue was confirmed via demonstration with the ExpressPoll.

14.7 Premier Digital Guardian Vulnerabilities

14.7.1 Users with administrative access can circumvent boot restrictions

If an attacker can boot from a different operating system, e.g., a "live CD," the system can be modified without Digital Guardian protections. Therefore, Digital Guardian contains explicit policy to keep the Windows boot time selection from modification. Due to a vulnerability in the Digital Guardian policy, a user with administrative access can circumvent protections to modify the boot time selection and load an operating system from CD.

Description: Digital Guardian cannot provide protection to the file system if it is accessed by another operating system. For example, if an attacker boots the system off of a "live CD," files can be modified without restriction. The Digital Guardian policy explicitly denies all users access to the Windows `boot.ini` file. This file is used by Windows to select from multiple operating systems that may be installed on the machine. It is also commonly used to allow the user to boot to the "Recovery Console" without booting from the Windows installation CDROM.

While the policy denies access to `boot.ini`, it does not deny access to `ntldr`. The `ntldr` file is a executable used by Windows to bootstrap loading the core operating system, e.g., the Windows kernel. `ntldr` has a hardcoded string to look for `boot.ini`. The `boot.ini` determines which operating system to load. An attacker and use a file (hex) editor to modify `ntldr` to look for a different file, e.g., `b00t.ini`, that is controlled by the attacker.

Controlling the boot entries provides an attacker great flexibility. If the recovery console software is installed, an entry can be created that allows a user with the administrator password to disable Digital Guardian. However, even more powerful, the adversary can use well known open source bootloaders that run from the `boot.ini` file (e.g., Grub4DOS) to boot from a CDROM. This is particularly useful when the system BIOS is configured to not allow these boot options. Once the attacker boots to this "portable operating system," he has complete control over the Windows file system. Possible attacks include modifying files (e.g., the election database), retrieving system passwords, and disabling Digital Guardian. See Issue 14.7.3 for more information about disabling Digital Guardian from removable boot media.

Prerequisites: The attacker requires a few minutes time and access to a user account with administrative privileges. Additionally, the attacker requires a way to copy necessary files and tools to the GEMS server, e.g., CDROM, USB drive, network. The CDROM drive is required if the attacker wishes to boot from a CDROM.

Impact: The attacker can boot a "portable operating system" to gain complete control over the system. Possible attacks include modifying files (e.g., the election database), retrieving system passwords, and disabling Digital Guardian. Note that *GEMSUser* has administrative access and can perform this attack (see Issue 14.7.2).

Procedural Mitigations: None.

Verification: This issue was confirmed via demonstration with the GEMS server.

14.7.2 The *GEMSUser* account is in the *Administrators* group

The *GEMSUser* is configured as a system administrator to the GEMS server. This allows the user to exploit other vulnerabilities, possibly disabling Digital Guardian.

Description: Along with the Digital Guardian policy, three Windows system users were created: *Administrator*, *GEMSadmin*, and *GEMSUser*. The *GEMSUser* account is a member of the system *Administrators* group. This allows the user to perform any administrative action on the machine. While Digital Guardian protects the system from some commands, this configuration allows normal users to exploit vulnerabilities they would otherwise be unable to perform. For example, due to this vulnerability, *GEMSUser* can completely subvert the system using Issue 14.7.1.

Prerequisites: Access to the GEMSUser password and the ability to log on to the GEMS server either via physical access or network connection.

Impact: *GEMSUser* can perform any administrative action on the system, possibly disabling Digital Guardian by exploiting other vulnerabilities (e.g., Issue 14.7.1).

Procedural Mitigations: Remove *GEMSUser* from the *Adminstrators* group.

Verification: This issue was confirmed via demonstration with the GEMS server.

14.7.3 Digital Guardian can be disabled by booting from external media

An attacker with the ability to boot from external media, e.g., a CDROM, can access the GEMS file system to disable and later re-enable Digital Guardian.

Description: If an attacker can boot from external media, e.g., a CDROM drive, Digital Guardian can be disabled. Booting to this media can be done either via BIOS boot menus or the operating system loader (see Issue 14.7.1).

Various "portable operating systems" (also know as "live CDs") can be used to disable Digital Guardian. If the *Administrator* account password is known, the attacker can boot from a Windows installation CD and select the recovery option. If the *Administrator* account password is not known, the attacker can boot to one of many well known password recovery/reset utility CDs to obtain/reset the *Administrator* password. The recovery console prompts for the administrator's password; however, once entered, the attacker can disable the Digital Guardian device drivers. When the system reboots, Digital Guardian is disabled. The same process can be used to re-enable Digital Guardian after an attack occurs.

If the attacker prefers not to use the *Administrator* account password, one of many Linux "live CDs" with a command line Windows registry editor can be used to keep the Digital Guardian drivers from loading. Note that while Windows is running, Windows protections do not allow modification of the part of the registry where driver settings reside. While this option requires slightly more sophistication, it does not require a password.

Finally, there exist ways to install Linux from within Windows. Therefore, it may be possible to mount this attack without booting from external media; however, due to time restrictions, we have not confirmed this.

Prerequisites: The attacker requires either BIOS password or Issue 14.7.1 and access to external media device, e.g. CDROM.

Impact: Digital Guardian will be disabled and can be re-enabled after an attack.

Procedural Mitigations: None.

Verification: This issue was confirmed via demonstration with the GEMS server.

14.7.4 An administrative user can disable the Digital Guardian device drivers once

The Digital Guardian policy is enforced by a set of system device drivers. We found that the device drivers can be disabled once after Digital Guardian is installed; however, after re-enabling the drivers, subsequent attempts to disable the drivers fail. However, the ability to disable Digital Guardian even once could allow an attacker to replace Digital Guardian with a malicious version.

Description: Digital Guardian implements its reference monitor using four device drivers that hook into various hardware and software interrupts in the OS kernel. The drivers are "hidden," but can still be viewed in "Device Manager." An attacker can easily identify the specific drivers that relate to Digital Guardian by looking at the "Provider" string for all drivers. Once identified, the attacker uses GUI controls to disable the drivers.

Through experimentation, we found that these drivers can be disabled only one time. After the drivers are re-enabled, attempts to disable the drivers fail. However, we were able to repeat the attack by restoring the hard drive disk image to the systems original state. Without access to the Digital Guardian source code we cannot understand why this occurs. We additionally tried removing and reinstalling the agent using the Digital Guardian console; however, in our tests this did not allow us to disable the drivers a second time.

In many cases, an attacker only needs to disable Digital Guardian one time. Once it is disabled, the attacker can replace the drivers with ones containing a backdoor. Note that any user with administrative access can perform this attack; this includes *GEMSUser* (see Issue 14.7.2). Finally, we believe this problem cannot be fixed by changing the Digital Guardian policy and should be considered a flaw in the Digital Guardian software.

Prerequisites: An attacker requires access to a user account with administrative access on a GEMS server where Digital Guardian has not already been disabled via this issue.

Impact: Users with administrative access, including *GEMSUser* can easily disable Digital Guardian.

Procedural Mitigations: None.

Verification: This issue was confirmed via demonstration with the GEMS server.

14.7.5 Users with administrative access can circumvent many Digital Guardian controls

Digital Guardian is used to restrict even users with administrative access. However, within Windows, there exists a user with even more privileges than *Administrator*. While no one can log in as this user, a user with administrative access can execute commands to gain these special privileges. Many of the Digital Guardian policy rules do not restrict a user with these privileges.

Description: The Digital Guardian policy explicitly controls what the *Administrator* user can do on the system. However, within Windows there exists a user with even more privileges than *Administrator*: *SYSTEM*. Many system services run as *SYSTEM*. While a user cannot log in as *SYSTEM*, there are well known ways for a user with administrative access to open a command line as *SYSTEM*. We used the command line task scheduler to gain such a shell. We stress that while simply keeping users from executing this task sched-

ule stops this specific method of gaining *SYSTEM* access, it does not solve the problem. See Issue 14.7.6 for further discussion on the drawbacks of protecting a system by denying execution of a specific set of applications.

With the current system configuration, an attacker logged in as either *Administrator* or *GEMSUser* can easily create a new *SYSTEM* shell. Once the attacker gains this access, many Digital Guardian restrictions can be circumvented. We were able to open the main system management console, which is explicitly denied by the policy. Digital Guardian opened a window to tell us the action was denied; however, the console opened anyway.

The Digital Guardian policy denies access to files by specifying the path name in the form of a regular expression. We found that this shell could get around any Digital Guardian restrictions that look purely at the path name, e.g., we could rename files with the `.mdb` extension, or modify them directly.

The Digital Guardian policy keeps applications from executing by specifying either the filename, or the cryptographic hash (MD5) of the application's binary. We found that the rules specifying hashes still applied; however such rules that explicitly deny execution can be circumvented via other means (see Issues 14.7.6 and 14.7.7).

Prerequisites: An attacker requires access to a user account with administrative access and less than one minute to gain *SYSTEM* access.

Impact: Digital Guardian protections can be circumvented, and the election database can be compromised.

Procedural Mitigations: None.

Verification: This issue was confirmed via demonstration with the GEMS server.

14.7.6 The Digital Guardian policy denies only specific known unwanted applications

The Digital Guardian policy protects the system from modification by denying specific applications from executing. An attacker can make simply modifications those applications to allow them to run.

Description: The Digital Guardian policy protects the system from modification denying specific applications from execution. This technique is commonly know as blacklisting. The policy covers many "dangerous" applications such as registry editors and system management control panels that administrators typically use to reconfigure a system. Fundamentally, many recommend against the use of blacklists, because enumerating *every* potential bad item, in this case application, is often impractical. Issue 14.7.5 describes such an application that was missed by the policy designers. Furthermore, using a blacklist in this way cannot stop applications copied to the machine by an attacker.

The current Digital Guardian policy uses two techniques to specify blacklisted applications. The first technique specifies a filename. Whenever the filename appears in a file system operation, e.g., read, copy, and execute, the operation is denied. Unfortunately, an attacker wishing to overcome this restriction can bring along the same application with a slightly different name.

The second technique specifies the cryptographic hash (MD5) of the application binary. Applications matching the blacklisted MD5 hashes are denied execution; however, the Digital Guardian implementation does not operate exactly like this, see Issue 14.7.7. Again, an attacker can overcome this restriction. The rule does not keep the attacker from copying the executable and then slightly modifying internal text fields with a file (hex) editor. This will change the MD5 hash and the application will no longer match the blacklisted

value.

Circumventing the blacklist restrictions can manifest itself in many ways. The blacklist exists to maintain the system's configuration. By changing the system configuration an attacker changes the way the Digital Guardian policy is interpreted. For example, election procedure can dictate that the *GEMSAdmin* account (used to manipulate election databases) is disabled to restrict how the election database is modified. An attacker can bypass blacklisted applications to temporarily re-enable the account long enough to make desired changes.

Prerequisites: Depending on applications existing on the GEMS server, an attacker may require the ability to copy files to machine, e.g., CDROM, USB, network. With the proper tools, the Digital Guardian policy can be circumvented within seconds.

Impact: Digital Guardian policy can be circumvented, potentially enabling an attacker to disable Digital Guardian or modify the election database.

Procedural Mitigations: None.

Verification: This issue was confirmed via demonstration with the GEMS server and provided documentation of the Digital Guardian policy.

14.7.7 Digital Guardian execution restrictions are circumventable by copying files

The Digital Guardian policy protects the system from modification by denying specific applications from executing. Flaws in the way Digital Guardian identifies applications allows an attacker to simply copy a restricted application in order to run it.

Description: As discussed in Issue 14.7.6, the Digital Guardian policy blacklists a number of potentially dangerous applications by specifying the cryptographic hash (MD5) of the binary executable. While Issue 14.7.6 discussed how this technique is trivially circumventable using a file (hex) editor, there is a flaw with the way Digital Guardian matches an application to a blacklisted MD5 hash value. By specifying MD5 hash values in the Digital Guardian policy, one would expected the hash value to be calculated for every application that executes; however, this is not the case.

When a blacklisted application is copied to a local directory, Digital Guardian allows the program to execute. We found that after the system ran for some duration and was rebooted, possibly multiple times, Digital Guardian spontaneously began restricting the execution of the application copy. We cannot be sure of exactly how Digital Guardian functions without access to the source code.

While we did not fully investigate the situations under which Digital Guardian begins restricting the application copy, we performed a few experiments to gain a better understanding. We found that modifying the application copy with a hex editor (as described in Issue 14.7.6) keeps Digital Guardian from identifying the copy, and its execution is never restricted. We found that modifying an application copy that was already found by Digital Guardian with a hex editor allows it to execute.

Prerequisites: Access to a user account on the GEMS server.

Impact: Users can execute applications that are denied by the Digital Guardian policy.

Procedural Mitigations: None.

Verification: This issue was confirmed via demonstration with the GEMS server.

14.7.8 *GEMSUser* can use the CD burning application to modify the election database

One goal of the Digital Guardian policy is to only allow GEMS to modify election databases. A policy decision was made to allow *GEMSUser* to make CD backups of the election database. The *GEMSUser* can use the CD burning application to rename the election database, modify it using an arbitrary application, and restore the file to its original name.

Description: One goal of the Digital Guardian policy is to only allow *GEMSUser* to interact with election database files using GEMS. The only purpose of the *GEMSAdmin* account is to manage database files with `explorer.exe`. However, on election day, *GEMSUser* must frequently backup the election database. The current Digital Guardian policy allows *GEMSUser* to interact with `.mdb` and `.gbf` files using the Nero Burning ROM application to back up the files to a CD-R. Note, `.mdb` files contain the election database modified by GEMS, and `.gbf` files contain an encrypted copy of an election database; however Digital Guardian policy treats both file types identically.

Unfortunately, the file browser included with the Nero Burning ROM allows GEMS to copy and rename files. Using Nero, an attacker with access to the *GEMSUser* account can rename `.mdb` files to a different file extension to gain unmediated access. Additionally, note that the Digital Guardian policy has an explicit rule to keep GEMS from deleting `.mdb` files; however, Nero is not included in this rule.

Prerequisites: Access to the *GEMSUser* account.

Impact: The *GEMSUser* can gain unrestricted access to election databases.

Procedural Mitigations: None.

Verification: This issue was confirmed via demonstration with the GEMS server.

14.7.9 The election database protections only apply to files on the local hard drive

One goal of the Digital Guardian policy is to only allow GEMS to modify election databases. However, the current policy does not apply to files that are not on the local fixed hard drive. For example, if an election database file is copied to a USB flash drive, arbitrary applications may modify it. This allows an attacker to modify copied election databases with an application the depends on the file extension.

Description: One goal of the Digital Guardian policy is to only allow *GEMSUser* to interact with election database files using GEMS. The Digital Guardian policy specifies which applications may access the `.mdb` and `.gbf` election database files. However, the rule only applies when the files reside on the local fixed hard drive (explicitly stated in the policy definition). As a result, the policy allows unrestricted access to `.mdb` and `.gbf` files on external media, e.g., a USB flash drive or CDROMs. Combining this issue with Issue 14.7.8, an attacker can use Nero to copy a `.mdb` file to a USB flash drive, and then use Issue 13.1.2 to modify the file before copying the file back to the GEMS directory with Nero. While this issue may appear to provide only minimal advantage over Issue 14.7.8, we note that the attacker may wish to use a database tool that requires a file to be of a known file extension.

Prerequisites: An attacker requires a few minutes of access to the *GEMSUser* account, an attached USB flash drive, and Issue 14.7.8 or some other way of copying the `.mdb` files, e.g., Issues 14.7.2 and 14.7.5.

Impact: The *GEMSUser* can modify `.mdb` files copied to a USB flash drive.

Procedural Mitigations: None.

Verification: This issue was confirmed via demonstration with the GEMS server.

14.7.10 Digital Guardian logging is disabled

The instructions for installing Digital Guardian specify that logging should be disabled. As such, Digital Guardian does not report attempts to circumvent it, despite the displayed message indicating an infraction has been logged.

Description: Digital Guardian provides additional protection of the GEMS server. When a user performs an action that is denied by the Digital Guardian policy, that action is denied and a dialog box informs the user that the action has been blocked and recorded. However, installation guides provided for setting up Digital Guardian on GEMS servers explicitly indicates that the "Enable Activity Detail Logging" option should be unchecked. This guide corroborates discussions with state employees indicating Digital Guardian logging is disabled due to storage concerns. Furthermore, we were unable to view policy violations in the Digital Guardian console laptop used to administer the GEMS servers. Finally, we note that system security logs often contain many false positives. We cannot speak to the usefulness of the logs produced by Digital Guardian (having never seen them).

Prerequisites: None.

Impact: Digital Guardian does not report policy violations or other attempts to circumvent restrictions.

Procedural Mitigations: None.

Verification: This issue was confirmed by document analysis and via demonstration with the GEMS server.

14.7.11 Digital Guardian does not immediately detect if GEMS is replaced

Flaws in the way Digital Guardian identifies applications allow an attacker to replace the GEMS application and gain unrestricted access to election databases. We were unable to conclusively confirm the duration before Digital Guardian detects the change, if ever. It is possible that GEMS could be replaced with a malicious application for the duration of an election.

Description: One goal of the Digital Guardian policy is to only allow *GEMSUser* to interact with election database files using GEMS. As discussed in Issue 14.7.7, there is a flaw in the way Digital Guardian identifies applications. Issue 14.7.7 described how the Digital Guardian policy identifies applications that may not run by specifying the cryptographic hash (MD5) of the application's binary executable. The vulnerability in Issue 14.7.7 is that an attacker can copy restricted applications in order to run them.

The Digital Guardian policy uses the same hash value identification in order to specify that `GEMS.exe` is one of the only applications that may access election database files (`.mdb` and `.gbf` files). In the system's current configuration, an attacker can copy a malicious program to the exact file path of the GEMS executable, and even though the MD5 hash of the malicious application is different from that specified in the policy, Digital Guardian treats it as if it was GEMS.

We were unable to conclusively confirm if Digital Guardian will eventually recognize the new application has a different hash value. In our limited testing, Digital Guardian did not recognize the application replacement. However, regardless of whether or not the malicious application is recognized, the issue gives an attacker a sizeable window to mount an attack.

Prerequisites: An attacker requires access to a user account on the GEMS server that has permission to modify or rename the GEMS executable.

Impact: Any administrative user, including *GEMSUser*, can replace the GEMS application with a malicious application to gain unrestricted access to election databases. We were unable to conclusively confirm how long the GEMS application can be replaced before Digital Guardian detects the change. It is long enough to mount an attack that immediately modifies the database; however, there is a chance Digital Guardian will not detect the change long enough to allow the malicious application to remain for the duration of an election.

Procedural Mitigations: None.

Verification: This issue was confirmed via demonstration with the GEMS server.

14.8 Premier AV-TSX Vulnerabilities

14.8.1 The wires connecting the VVPAT printer can be cut without opening the AV-TSX

The power and data cables connecting the printer to the AV-TSX are easily accessible without opening the casing.

Description: In the state of Ohio, the paper receipt produced by the *Verifiable Voter Paper Audit Trail* (VVPAT) printer is the official record for an election. Accordingly, a machine may not be used in an election if its printer is not properly functioning.

The plastic enclosure surrounding the printer is highly flexible. By simply pushing the bottom edge of this enclosure, the data cables and power supply connecting the printer to the AV-TSX become exposed. The space created by pushing the plastic can easily fit a number of fingers or a small cutting device. An attacker could therefore successfully disable the printer in a matter of seconds.

Prerequisites: A voter or poll worker with 30 seconds of access to the machine could exploit this vulnerability.

Impact: If voters were not aware that a paper receipt was supposed to be printed, an unverifiable election may be run on the attacked unit. Should an audit expose that the printer was not working, the votes cast before this discovery may need to be thrown out. The attack could also be used to launch a Denial of Service, preventing this machine from being used in a polling place. Polling places with a small number of machines may be completely shut down by one or a few attackers.

Procedural Mitigations: Tamper evident seals are a potential partial mitigation to alert poll workers that an attack has occurred. For practical limitations, see Section 3.5. This particular attack, however, is obvious by its nature as the printer will no longer work.

Verification: This vulnerability was confirmed through penetration testing.

14.8.2 The plastic housing protecting the printer can easily be removed

The printer is attached to the AV-TSX by a protective plastic enclosure. However, this cover is easily removable, giving an attacker access to the records of previously cast ballots.

Description: The VVPAT printer is enclosed in a plastic housing. When set properly over the printer, the enclosure is secure to the voting machine using two mechanisms. The first, a 1/8 inch plastic latch, sits below the spool of cast votes. Access to this latch is protected by a lock which keeps the cover to the printer closed.

Without opening the cover and accessing the latch, the enclosure protecting the printer can be removed. An attacker would then have direct access to the printer, all of the paper spools and the wiring inside.

Prerequisites: A voter or poll worker with 30 seconds of access to the machine could exploit this vulnerability.

Impact: Like the previous exploit, this attack would allow an attacker to render the printer useless. More critically, the attacker could gain access to the unused paper (potentially modifying it to cause jamming errors) or even gain access to cast ballots. A voter or poll worker could successfully carry out this attack with under 30 seconds of access to an AV-TSX.

Procedural Mitigations: Tamper evident seals are a potential partial mitigation to alert poll workers that an attack has occurred. For practical limitations, see Section 3.5. This particular attack, however, is obvious by its nature.

Verification: This vulnerability was confirmed through penetration testing.

14.8.3 An adversary can destroy paper copies of cast ballots without opening the AV-TSX

The plastic enclosure protecting the printed paper ballots us not adequately sealed. An adversary can press against or inject a number of common household chemicals into the enclosure and destroy all paper records.

Description: The plastic enclosure protecting the VVPAT printer allows a voter to see the paper copy of their ballot without allowing them to directly access that paper. However, both the clear plastic window and the enclosure itself are not sufficiently sealed. An attacker can exploit this weakness by using a syringe to inject any number of compounds known to degrade/destroy information written to thermal printer paper.

Because of the design of the AV-TSX, it may be possible to damage the ballot paper trail in multiple ways. First, such a compound could be inserted in multiple ways such that all previous paper ballots stored in a machine would become unreadable. Alternatively, all of the unused paper in the machine could be attacked, preventing all future votes from being tallied. An example of this attack is shown in Figure 14.3.

A similar vulnerability was discussed in the California Red Team report (Issue 4.f); however, the details of this vulnerability were not listed in the public report.

Alternatively, the printer can be caused to jam simply by pressing against the enclosure. All results and logs sent to the printer after this occurs print over each other and are therefore unreadable.

Prerequisites: An attacker could successfully execute this attack with a few seconds of access to the AV-TSX.

Impact: An attacker would be able to damage or destroy the current and future votes cast on a roll of paper stored in an AV-TSX.

Procedural Mitigations: None.

Verification: This vulnerability was discovered through red teaming analysis.

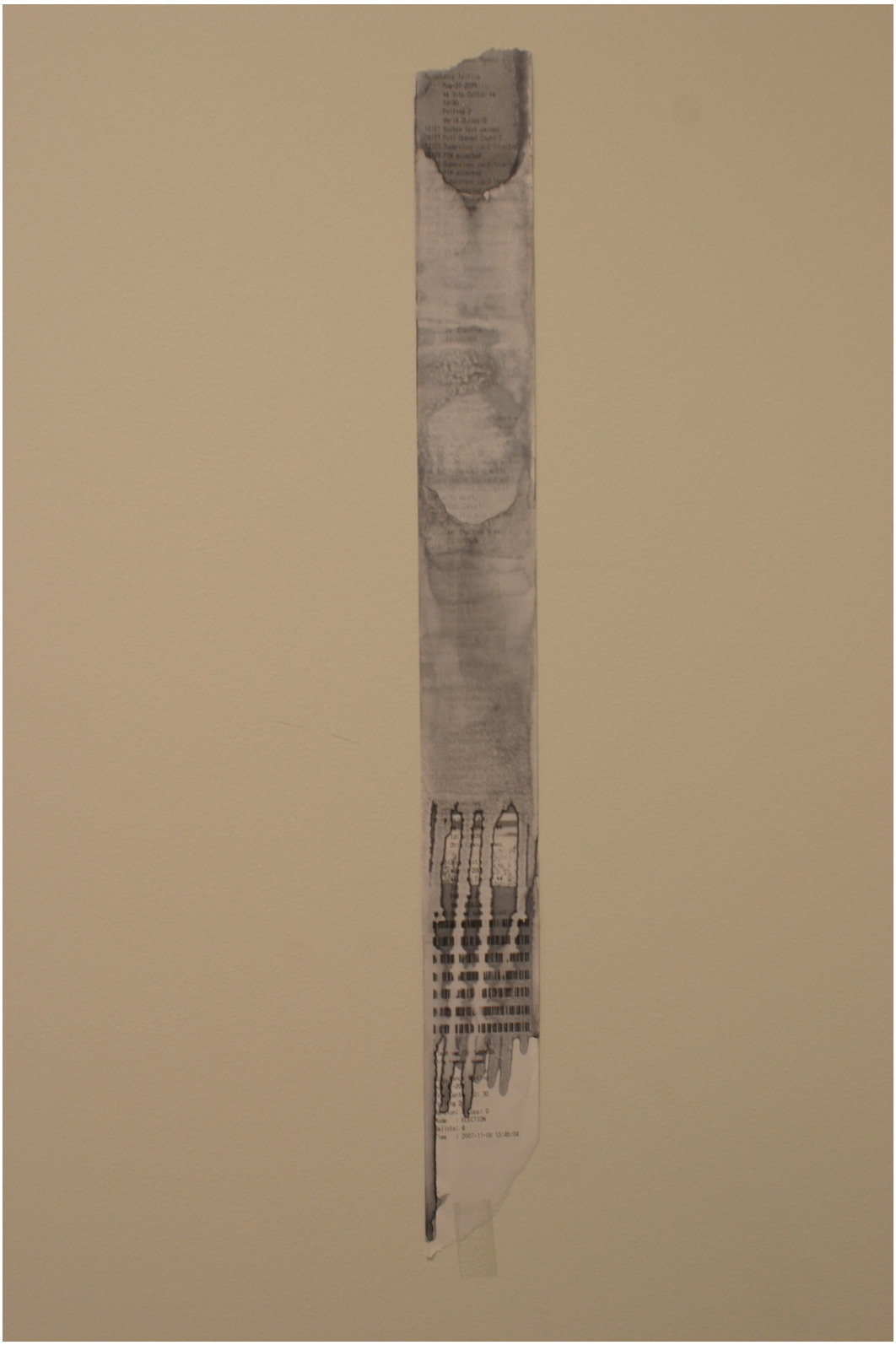

Figure 14.3: A paper copy of the system log destroyed by injecting a household chemical into the AV-TSX. No seals were broken in this attack.

14.8.4 The power button is accessible when all panels on the AV-TSX are closed and locked

The AV-TSX power button is located behind the same locked door protecting the memory card. Because this door is not sufficiently sealed, an attacker can easily turn the power on and off.

Description: The power button is located next to the PCMCIA memory card slot on the upper left hand side of the machine. Both of these parts of the machine are protected by a locked plastic shield when the AV-TSX is used in an election. However, the placement of the power button and the gapping between the shield and the main chassis allow an attacker to turn of an AV-TSX without removing the shield.

Prerequisites: A voter or poll worker with 30 seconds of access to the machine could exploit this vulnerability. Members of our team were regularly able to exploit this issue in under 10 seconds.

Impact: An adversary could cause a machine to shut down unexpectedly. This is a necessary prerequisite to a number of other attacks.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through penetration testing.

14.8.5 The bootloader can be manipulated to give access to the file system on the AV-TSX

The bootloader will allow an attacker to run Windows Explorer on the AV-TSX if a conditional statement evaluates to "True". This value is set to false in commercial releases; however, simply making one change to the binary file will allow Windows Explorer to run.

Description: Previous versions of the bootloader software allowed an attacker to run Windows Explorer instead of BallotStation simply by including the file `explorer.glb` on a memory card. This method of running Windows Explorer was removed from the version of the code we examined.

However, it is still possible for the AV-TSX to boot Windows Explorer instead of BallotStation. By setting a variable system, an attacker with access to a binary of the bootloader can use software called a "hex editor" to gain access to Windows Explorer simply by changing the value of the variable. This attack is possible because the code to run Windows Explorer is included in commercial releases of the build.

An attacker could gain access to the bootloader in any number of ways. As an insider, they may be able to take the binary from elections headquarters. A poll worker with access to a machine during a "sleep-over" may be able to read the binary from the AV-TSX by exploiting Issue 13.3.4. States may also potentially sell older or surplus voting units, giving the general public access to the machine and a valid bootloader.

Having acquired a copy of the binary bootloader, the attacker could then change the value of the variable to TRUE using a hex editor. This change would be relatively quick to make. The attacker could load the modified bootloader onto memory cards and, because of Issue 13.3.1, cause any machine to load Windows Explorer.

Note that there is no need for Windows Explorer to be run on a commercial build of the software. Premier has built in a number of software update mechanisms into the memory card itself (all of which are insecure - see Issues 13.3.1 and 13.3.2). At no time should anyone be able to run Windows Explorer in the polling place.

Prerequisites: An attacker would need access to a copy of the bootloader binary and a hex editor.

Impact: An adversary could use Windows Explorer to change arbitrary settings and install malicious software on the AV-TSX. This would provide a helpful user interface such that the attacker creating the modified bootloader could rely on less sophisticated attackers to exploit the vulnerability in polling places.

Procedural Mitigations: No election official should be given unsupervised overnight access to equipment such as the AV-TSX (known as a "sleep-over"). Tamper evident seals are a potential *partial* mitigation to denote access to the memory card slot has occurred. For practical limitations, see Section 3.5.

Verification: This vulnerability was confirmed through source code analysis.

14.8.6 The memory of an AV-TSX can be erased by a memory card

While booting, the AV-TSX checks for the presence of a number of files. If these files are present, it will delete the contents of its registry and/or entire memory. The machine would not be usable until a legitimate update was reinstalled by election officials.

Description: Before BallotStation is started, the bootloader searches a memory card for a number of files. As discussed in Issue 13.3.1, this mechanism is designed to allow software updates to propagate to the AV-TSX. When checking to see if a new bootloader or operating system are present on the memory card, the bootloader looks for two additional files: `ERASEPSM.STL` and a second previously unnamed file (see the Private Appendix). If files with these names are present on the memory card, the AV-TSX will delete the contents of its entire flash memory or just the contents of the registry, respectively. This is achieved by a call to `FlashErase`, which overwrites the contents of the memory with `0xFF` (a string of ones).

Part of the vulnerability was identified by Hursti[5]. In previous versions of the software, Hursti noted that the presence of the file `ERASEPSM.STL` on a memory card caused the AV-TSX to delete the contents of its flash memory. While the current version of the bootloader contains a similar issue (with the filename slightly modified), it now requires the DEBUG jumper (Issue 13.3.4) to be set in order to delete the entire memory. As shown earlier, however, this would do little to stop a determined adversary. The registry, however, can be deleted without any requirements beyond the presence of the second previously unnamed file.

Neither file is authenticated in any way. Accordingly, anyone that can access a memory card for a few seconds can write the necessary files.

Part of this vulnerability (flashing the contents of the entire memory) was previously reported by Hursti[6]. Flashing the registry alone has not previously been reported.

Prerequisites: An attacker would need access to the memory card for both attacks. To delete the entire memory, the attacker would also need to be able to set the DEBUG jumper (see Issue 13.3.4).

Impact: An attacker would not need to delete the entire contents of the memory in order to prevent the AV-TSX from operating. Because so many critical values are stored in the registry (e.g., cryptographic keys, machine serial number), deleting the registry would prevent the AV-TSX from operating correctly. Because a new registry is included with the BallotStation binary, loading a new `.ins` file could reverse this attack if it were launched at the polling station.

Allowing poll workers to have access to a memory card and to the memory card slot in public may be more dangerous. This card may easily be misplaced or stolen by an attacker, giving them the ability to develop more sophisticated attacks against the system. However, in the absence of a card with a valid build of

[5]Harri Hursti, *Supplemental report, additional observations*. Black Box Voting, Unredacted release July 2, 2006, May 22, 2006.
[6]Hursti, 'Supplemental report, additional observations' (as in n. 5).

BallotStation, the exploited AV-TSX would be unusable.

Procedural Mitigations: Chain of custody logs are a potential *partial* mitigation to protect memory cards. For practical limitations, see Section 3.5.

Verification: This vulnerability was confirmed through source code analysis.

14.8.7 A voter can gain administrative access to the AV-TSX

Any voter can discretely gain access to the administrative screens of an AV-TSX. Among other options, this attacker can unload or delete the results of an election.

Description: Voters necessarily have limited capabilities on the AV-TSX voting machine. Many operations, such as unloading an election, printing reports and manipulating settings must only be accessible by trusted elections officials with access to the correct smart cards. However, it is possible for a voter to gain central administrator-level access to a machine during the course of an election. Most critically, votes can delete all cast ballots stored on both the memory card and internal memory without any special tools.

We discuss the details of this attack in the private appendix.

This attack was independently discovered; however, at the end of our analysis, we were given access to the private reports from the California TTB review. These reports indicate a similar attack yielding the same outcome.

Prerequisites: With any smart card and the ability to turn the power on and off, the attacker can achieve this elevated level of access.

Impact: An attacker could delete the results of an election or make an AV-TSX unusable.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through penetration testing.

14.8.8 A voter can cast an unlimited number of votes without any tools or knowledge

A voter with no special tools can program an infinite number of smart cards and therefore cast an unlimited number of votes. The attacker does not need to break any protocols or intercept any communications.

Description: A Voter smart card allows each voter to cast a single ballot when they interact with the AV-TSX. However, without any special tools or the Supervisor smart card, it is possible for a voter to create an unlimited number of voter cards using the AV-TSX. Alternatively, the attacker can reprogram a single smart card an unlimited number of times.

This attacks causes the machine to incorrectly enter the Supervisor Menu, from which the user is given the option to create voter cards.

Prerequisites: The attacker simply needs access to the AV-TSX during an election. Further details are contained within the Private Appendix.

Impact: An attacker can cast an unlimited number of votes without any special tools or previous interaction with any AV-TSX.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through penetration testing.

14.8.9 The smart card reader can be jammed by an attacker

An attacker can cause the card reader on the AV-TSX to hold a smart card, thereby blocking anyone else from using the machine. This attack is temporary and can be mitigated by rebooting the machine.

Description: When the smart card reader on the AV-TSX is finished reading a valid card or recognizes that an invalid input has been inserted, it attempts to physically eject the card from the slot. This allows other voters or administrators to apply valid cards and gain appropriate access to the AV-TSX.

It is possible, however, to cause the AV-TSX to refuse to return a smart card to a user. In so doing, subsequent voters and or administrators will not be able to use the machine as intended as they can not insert their valid cards. Because the majority of the card is kept inside the machine, even a reasonable amount of force will not dislodge the card. The card can be removed with a pair of pliers; however, this causes the entrance of the card reader to close, thereby preventing other cards from being inserted.

The only means of freeing the card is by rebooting the machine. This will force a precinct administrator to unlock the compartment containing the power button and memory card during an election. Any tamper evident seal attached to this compartment will also have to be voided.

Prerequisites: An attacker requires no special tools or previous access to the AV-TSX.

Impact: An attacker can create a number of problems with such an attack. By jamming a card in the slot, an AV-TSX may become unusable for the remainder of an election. If this tactic is taken against a large number of machines in a precinct, administering an election may become difficult in the allotted time. Alternative, the attacker may wish to have an elections official intentionally void the tamper evident seal on the memory card compartment to remove any suspicion of subsequent attacks.

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through penetration testing.

CHAPTER 15

PREMIER COMBINED IMPLICATIONS AND ATTACK SCENARIOS

15.1 Casting an Unlimited Number of Ballots

Premier Election Systems use smart cards to ensure that each voter is only able to cast a single ballot per election. After casting their ballot on an AV-TSX, the card reader marks the card as "Cast". If this card is reinserted into an AV-TSX before it is re-enabled by a poll worker (using either the ExpressPoll or Voter Card Encoder), the voting machine ejects the card and alerts the user that it has already been used. Implemented correctly, this mechanism should prevent a single user from casting more than their allotted single ballot.

Using multiple vulnerabilities discovered during this evaluation, it is possible to enable a voter to bypass these mechanisms and cast an unlimited number of votes. Moreover, the evidence that such an attack has been launched can also be erased. Worse still, this attack requires no special tools or private knowledge of the system.

We assume that our attacker approaches the voting booth during an election under normal circumstances. The attacker brings with them a stack of smart cards containing the default Smart Card Key (published on the Internet). After approaching the AV-TSX, the attacker begins by covering his/her tracks. Because the AV-TSX notes in its audit logs when cards have been encoded, the attacker accesses the Central Administrator by exploiting Issue 14.8.7. Here, the attacker can delete the contents of both the memory card and the AV-TSX, thereby erasing most evidence of the attack. To hide the card creation operations, the attacker then simply changes the time and date of the AV-TSX to a period before the election. This portion of the attack can be accomplished in just over one minute. Moreover, deleting the contents of the memory card and changing the time/date are not logged. Should the attacker also worry about the log information encoded on the VVPAT, they can exploit either weakness discussed in Issue 14.8.4.

To encode voter cards, the attacker gains access to the Supervisor Menu by exploiting the vulnerability described in Issue 14.8.8. Access to this menu can be achieved consistently in under one minute. The attacker then encodes the stack of smart cards smuggled into the voting precinct as valid Voter Cards, each of which takes a few seconds. There is no limit to the number of cards that can be programmed.

The attacker can then walk away from the machine and give the cards to colluding adversaries in the parking lot. These adversaries can use all of the cards to cast extra votes.

15.2 Pre-Stuffing an Election

The concept of "pre-stuffing" an election using the AV-OS PC memory card is not new; however, the vulnerabilities that enable this attack are still present. Since the idea was first suggested by Hursti[1], subsequent researchers have demonstrated the attack and even proposed slight variations.

The AV-OS PC is the primary means of reading paper ballots at the polling place. The most critical component of the AV-OS PC is the memory card, which contains the number of races, the location of the ovals for candidates, the current number of votes for each candidate, and even "live code" used to tell the AV-OS PC how to print the Election Zero report.[2]

AV-OS memory cards have no integrity protection mechanisms (see Issue 13.2.1). An attacker that physically possesses the memory card can therefore modify it in arbitrary ways. For instance, an attacker gaining access to the memory card after an election can submit forged results. Before an election, an attacker can update the ballot coordinates for ovals and completely change a voters intention or even neutralize candidates such that they receive no votes. Researchers working for the Connecticut Secretary of State provided an extensive scenario under which such an attack may occur[3]. While this attack does not exactly "pre-stuff" an election, it definitely has significant impact on the outcome.

The original "pre-stuffing" attack proposed by Hursti makes use of lack of memory card data protection, the existence of "live code" on the memory card, and a flaw in the way the AV-OS PC counts and stores votes (see Issue 13.2.7). The AV-OS PC does not store individual votes; rather, it stores the aggregate votes for each candidate. Like personal computers, the AV-OS PC store numbers in fixed size storage, which means if a number is incremented past a maximum value, it will reset to zero. In the AV-OS PC, the maximum value is 65,535, and it does not check to see that value is ever reached. While 65,535 votes may seem sufficiently large for any polling place, an attacker can use the "roll-over" side-effect to "pre-stuff" an election such that one candidate has a fixed advantage over another. An attacker with physical access to the memory card before an election can adjust the vote counter for *Candidate A* to 65,526 (-10) and the counter for *Candidate B* to 10. This gives *Candidate B* an advantage of 20 votes, since after *Candidate A* receives 10 votes, the counter will hold the value of zero. Furthermore, at the end of the election, the total number of votes on the memory card will match identically with the number of ballots in the ballot box.

When the election official prints the Election Zero report, however, *Candidate A* and *Candidate B* will not be zero. With access to the memory card, overcoming this limitation is trivial. Because the AV-OS PC is instructed how to print the Election Zero report by running the "live code" on the unprotected memory card, an attacker can simply modify the "live code" for the Election Zero report to always print zero. See the California Diebold source code TTBR report (Issue 5.1.11) for a step by step walk through of how the vote counter "roll-over" allows this attack to occur.

Some claim that this "pre-stuffing" attack can be detected by using proper election procedure. The memory card is always in one of several modes (e.g., pre-election, election mode, post-election) as indicated by a value stored on the card itself. It has been suggested that waiting until the card is in the polling place before changing it to election mode will provide defense against this attack. However, in a study for the California

[1] Harri Hursti, *The Black Box Report: Critical Security Issues with Diebold Optical Scan Design*. Black Box Voting, July 4, 2005 ⟨URL: `http://www.blackboxvoting.org/BBVreport.pdf`⟩.

[2] The Election Zero report is an important piece of documentation that assures poll workers and election officials that no votes have taken place before the election begins.

[3] A. Kiayias et al., *Security Assessment of the Diebold Optical Scan Voting Terminal*. UConn Voting Technology Research (VoTeR) Center, October 30, 2006 ⟨URL: `http://voter.engr.uconn.edu/voter/OS-Report_files/uconn-report-os.pdf`⟩.

Secretary of State, Wagner et al.[4] determined that the attack can occur no matter memory card state during shipment (Finding 10).

Finally, a recent study for the Florida Department of State[5] reviewed newer versions of the Premier source code than we were given. The study indicates that the "live code" now includes protections to keep an attacker from modifying it in transit. Without changing the "live code" on the memory card, the "pre-stuffing" attack can be detected before the election begins. However, the Florida study reported a flaw in the method used to protect the "live code," making the attack still possible.

The only known detection for this "pre-stuffing" attack is to recount all of the paper ballots.

15.3 Voting Machine Virus

A virus is a malicious computer program with the ability to automatically move from machine to machine. Upon infection, such programs can change the contents of files, record a user's operations or make a computer unusable. Because they can spread silently even in the presence of procedural safeguards, viruses represent a significant and realistic threat to the integrity of an election. By exploiting one of many of the issues presented in this report, an attacker could inject a virus into any AV-TSX, AV-OS, EMP or GEMS server and gain control of an election in a county.

We are by no means the first to suggest such vulnerability to viruses. Feldman et al[6] were in fact able to build their own virus targeting an older version of the Premier's touchscreen voting machine (AccuVote-TS). As the memory card from the infected machine was inserted into other machines, these machines automatically became infected. Such a virus could easily be replicated by an attacker with moderate programming abilities and access to a voting machine.

Our review of the source code and penetration testing of the range of Premier machines indicates that such attacks are highly plausible. This report has not only confirmed many of the previously known vulnerabilities enabling such attacks, but it has also uncovered a number of new serious new avenues through which viruses could infect voting machines. We briefly discuss a few of the points through which such an attack could occur.

15.3.1 Infecting the GEMS Server

The GEMS server should be the best protected element of an election system. These systems run antivirus software, rely on Digital Guardian to restrict the operations a user can execute and are typically kept physically secure within a county's election headquarters. However, none of these protections are sufficient to stop a virus targeting a voting machine.

Antivirus software can only recognize viruses that have been previously identified. Because voting machine viruses are not commonly found on the Internet, such virus "definitions" are not part of any known antivirus

[4]David Wagner, David Jefferson and Matt Bishop, *Security Analysis of the Diebold AccuBasic Interpreter*. Voting Systems Technology Assessment Advisory Board (VSTAAB), February 14, 2006 ⟨URL: `http://www.ss.ca.gov/elections/voting_systems/security_analysis_of_the_diebold_accubasic_interpreter.pdf`⟩.

[5]Ryan Gardner et al., *Software Review and Security Analysis of the Diebold Voting Machine Software*. Security and Assurance in Information Technology (SAIT) Laboratory, Florida State University, For the Florida Department of State, July 27, 2007 ⟨URL: `http://election.dos.state.fl.us/pdf/SAITreport.pdf`⟩.

[6]Ariel Feldman, J. Alex Halderman and Edward Felten, 'Security Analysis of the Diebold AccuVote-TS Voting Machine'. In USENIX/ACCURATE Electronic Voting Technology Workshop (EVT). 2007.

software. Accordingly, such software offers little protection against viruses targeting the GEMS server. The Digital Guardian software is also circumventable. An attacker able to exploit the vulnerabilities detailed in this section could easily circumvent the protections provided by this software. Moreover, while physical security certainly eliminates a significant number of attacks, it simply does not prevent a virus from infecting the GEMS server. An insider, for instance, can easily bypass all physical protections and secretively inject a virus onto the GEMS server.

15.3.2 Infecting the EMP Server

The EMP server is used to encode the memory cards sent to precincts and tally the results on a card when an election is completed. As the intermediary between the GEMS server and AV-TSX units deployed in precincts, the EMP server has significant access to critical pieces of election materials. For instance, while the communications between the GEMS and EMP servers can be protected by encryption, an infected EMP server could change ballot definitions before loading them onto memory cards. At the end of an election, an infected EMP server could change the results of ballots and then forge the authenticity of all results using or generating the necessary keys (Issues 13.3.5 and 14.1.7). Many of the vulnerabilities found in the EMP server, some of which execute immediately upon the insertion of an infected memory card, would allow a virus to gain access to perform such attacks.

15.3.3 Infecting the AV-TSX

An attacker could load a virus onto an AV-TSX during multiple phases of an election using vulnerabilities such as Issues 13.3.1 and 13.3.2. If memory cards are placed in voting machines before delivery to a precinct, an attacker could use a number of known techniques to remove tamper evident seals and unlock the door protecting the memory card before the voting machine was delivered. Alternatively, a poll worker could access the memory card before an election as the power button is located behind the same locked door. In both of the above cases or if cards are delivered to precincts independently of machines, the memory card could be replaced by one containing a virus. When an election is finished and the memory card is inserted into the EMP server, the virus could continue to spread throughout the system.

15.4 Denying Voters the Ability to Vote

In many situations, preventing specific or all voters from participating in an election may be valuable to an attacker. For instance, the expense of attempting to run multiple elections may put significant financial strain on a county. Alternatively, by repeatedly "canceling" elections, an attacker may disenfranchise some portion of likely-voters. Vulnerabilities in every component of these systems allow such attacks to be successfully executed. We highlight a few of these in various devices.

Weaknesses in the ExpressPoll may allow an attacker to load their own modified version of the voter database (Issue 14.6.3). AV-TSX units can be crashed or made unusable though vulnerabilities including but not limited to Issues 13.3.18, 14.8.6 and 14.8.7. The EMP server can made to crash or unusable through vulnerabilities such as Issue 14.1.4 and 14.1.10. GEMS can also be made to crash through weaknesses such as Issue 14.3.1.

Part IV

Analysis of the Hart InterCivic, Inc. Voting Systems

CHAPTER 16

HART EXECUTIVE SUMMARY

This study evaluates the ability of the Hart voting system to conduct a trustworthy election. The review team was provided access to the Hart source code and election equipment. The reviewers studied these materials in order to identify any security issues that can be exploited to affect an election. As part of that analysis, the reviewers were asked to identify best practices that may limit or neutralize the impact of discovered issues.

Our evaluation suggests that the Hart system lacks the technical protections necessary to guarantee a trustworthy election under operational conditions. The vulnerabilities and features of the system work in concert to provide numerous opportunities to manipulate election outcomes or cast doubt on legitimate election activities. Such vulnerabilities are exploitable under election conditions, and often require minimal physical access to equipment or information. These vulnerabilities are a result of the following failures of the Hart system's design, implementation, and practices:

- *Failure to effectively protect election data integrity* - Virtually every ballot, vote, election result, and audit log is forgeable or otherwise manipulatable by an attacker with even brief access to the voting systems. These vulnerabilities place enormous burdens on physical procedures.
- *Failure to eliminate or document unsafe functionality* - There are a number of largely undocumented features in the system that are highly dangerous in a production election system. For example, existing features allow an attacker to remotely "script" DRE voting machines to cast votes as the attacker chooses, to allow a single (or photocopied) voter ballot to be counted many times, and to print pre-voted ballots that will be accepted by voting equipment. Note that all of these activities are not attacks *per se*, but are the apparent intended use of existing Hart system features. These features are available during a live election.
- *Failure to protect election from malicious insiders* - The protections in the Hart system that are intended to prevent election officials, poll workers, and vendor representatives from using dangerous features or modifying election data are circumventable. Attackers with access to the system can quickly recover critical system passwords, extract cryptographic keys, and reproduce security hardware. These artifacts are the "keys to the kingdom" that can be used to forge election data and compromise nearly all of the Hart election equipment.
- *Failure to provide trustworthy auditing* - The auditing capabilities of the Hart system are limited. Those features that are provided are vulnerable to a broad range of attacks that can corrupt or erase logs of election activities. This severely limits the ability of election officials to detect and diagnose attacks. Moreover, because the auditing features are generally unreliable, recovery from an attack may in practice be enormously difficult or impossible.

One of the critical discoveries in this study is that the full functionality of the Hart system is currently unknown, and in some cases may never be known. We found many functions and system configurations that were not documented and whose purpose was non-obvious. The vast majority of these remain unstudied for lack of reviewer time. Furthermore, certain interfaces–in particular in the election tally software–were designed to be augmented at run time with additional software. Because these interfaces were apparently designed to allow previously unknown software to be introduced into the live system, they represent a source of potential vulnerability.

Our findings are consistent with those of previous studies. The lack of protections leave the system vulnerable. Thus, the security of an election is almost entirely reliant on the physical practices. The technical limitations of its design further show that when those practices are not uniformly followed, it will be difficult to determine if attacks happened and what they were. Even when such attacks are identified, it is unlikely that the resulting damage can be contained and the public's confidence in the accuracy and fairness of the election restored.

CHAPTER 17

HART STUDY OVERVIEW

17.1 Part Structure

The following chapters detail the Hart portion of the EVEREST review. Readers are directed to Part 1 of this report for a discussion of the review's goals and methodologies. Those readers not experienced in information security or Ohio election practices are also encouraged to read the threat model in Chapter 3. The content in that chapter is instrumental in gaining a detailed understanding of the substance and impact of the issues identified throughout.

The issues and commentary described in this section of the report are *reflective of our analysis of the Hart system only*. None of the included material should be considered to apply to either the ES&S or Premier systems, nor should any statement be construed to be making any quantitative or qualitative comparisons of the different systems. Such comparisons are expressly outside the scope of the review, and we have not developed any opinions on the relative merits of any one system with respect to the others. Nothing in this review should be considered as a positive or negative commentary on electronic voting in general.

The remainder of this part of this report is structured as follows. We begin in the next section by giving a brief overview of the Hart system and its use in Ohio elections. Chapter 18 broadly characterizes the security and reliability of the Hart system by highlighting several architectural and systemic issues encountered in the review. Chapter 19 provides detailed technical information on the confirmation of previously reported issues and Chapter 20 provides similar detail on new issues uncovered by our analysis. Chapter 21 describes a number of attack scenarios that demonstrate how the identified issues may be used in concert to subvert an election.

17.2 Architecture

This chapter provides an overview of the architecture for the Hart InterCivic Voting System, including the Election Management System (EMS), the devices comprising a voting setup, and some of the back-end software in use. This information is compiled from public reports and experience of the Ohio EVEREST academic team with the system.

Figure 17.1 provides an overview of the setup and interconnected components within the Hart InterCivic voting system. The Hart system is comprised of a number of components, and can be delineated mainly between components found at county election headquarters versus those found in polling locales.

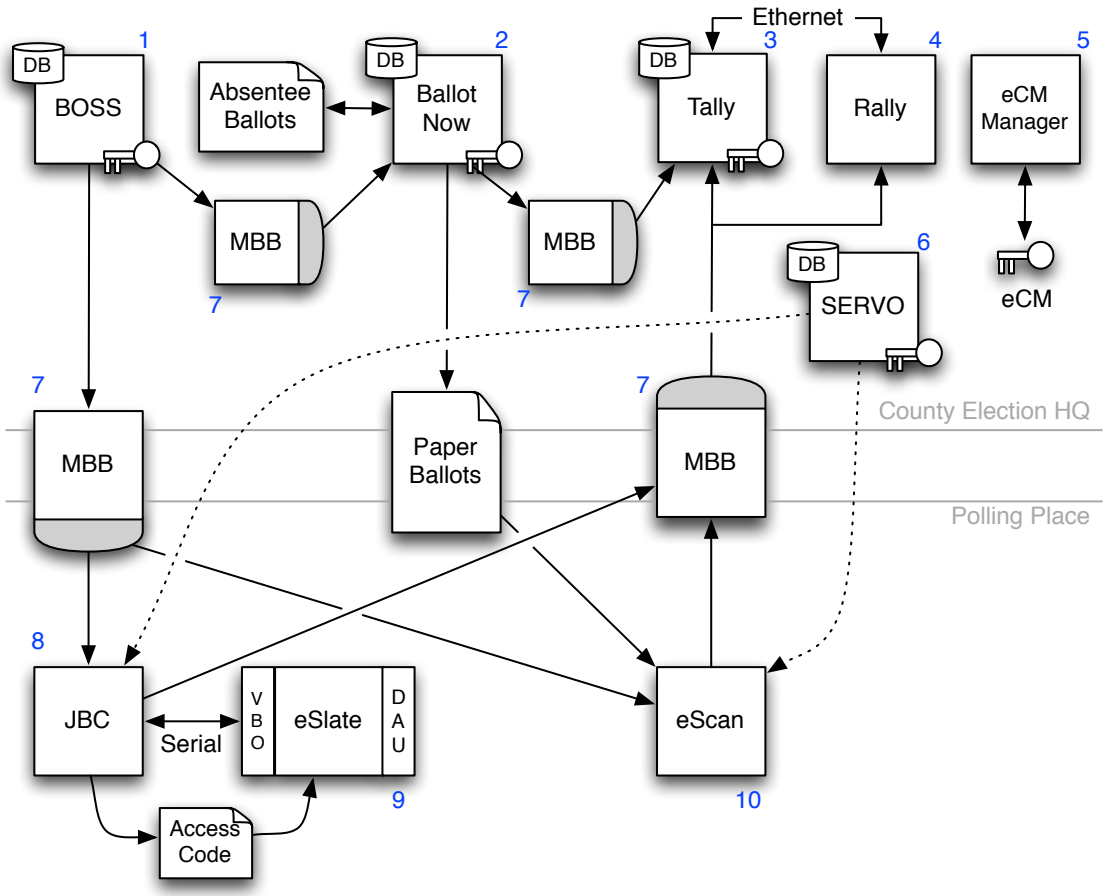

Figure 17.1: The major portions of the Hart system architecture and some of the connected components. Unless indicated otherwise, solid lines indicate a physical relationship between components, i.e., components are either physically connected or must be physically transported to interact with each other. Dashed light lines represent connections that take place between components outside the course of an operational election in progress.

17.2.1 Components at County Election Headquarters

A number of components within the Hart system reside at the county election headquarters. Refer to Figure 17.1 for a pictorial representation of component interaction.

1. **BOSS:** The Ballot Origination Software System is used to set up an election. It creates an election database (i.e., lists of candidates running for each office, precincts, etc.) and is used to define ballot templates, with information sourced from a Sybase database. In addition, passwords for the polling place voting equipment (the JBC and eScan) are defined in BOSS. BOSS exports election definitions to Mobile Ballot Boxes (MBBs). BOSS runs on a Windows 2000 server, and apart from the MBBs it generates, is disconnected from the rest of the election process.

2. **Ballot Now:** This software is used to generate paper and electronic ballots for elections. An MBB is created for Ballot Now by BOSS. With the information on this MBB, Ballot Now generates ballot

images for all districts, creating ballot images electronically for use in the eSlate machines and in a printer-ready form for use in eScan machines and for absentee ballots. In addition, it may be used with a commercial central count optical scanner to process absentee ballots, writing out an MBB containing all vote totals. Ballot Now runs on a server running the Windows 2000 Server operating system.

In reality, Ballot Now is a client-server architecture. The server attaches to the scanner used for counting absentee ballots. The clients are used for manual control of this counting process. We did not have the scanner, nor the networking elements to evaluate. Thus, there may be any number of additional issues that we could not identify for lack of equipment.

3. **Tally:** Tally accumulates and generates a tally of all votes for the electoral district. It runs on a Windows 2000 Server machine and is used to compute total votes based on the MBBs loaded into it from the eScan and JBC devices, and from Ballot Now. The database defining the election in BOSS is imported to Tally and used to provide candidate information and other information, including how many MBBs were generated and how many should be read. Once all MBBs have been tallied, Tally can be used to generate reports about the election outcome. Tally may also be used to manually adjust vote totals. MBBs that are not loaded directly into Tally are retrieved remotely from *Rally* (described below).

4. **Rally:** Rally is used to process MBBs from precincts. Its main purpose is to generate vote totals to communicate to Tally. In Ohio, Rally must be used at election headquarters and may only communicate with Tally over an internal private computer network; however, in other jurisdictions, Rally may be used remotely and communicates with Tally over a private network or modem connection with Tally, rather than physically transporting MBBs to the central locale. An SSL connection is established between Rally and Tally to secure the communication between the two processes. It runs on a Windows 2000 Server machine.

5. **eCM Manager:** This software is responsible for generating signing keys for eSlate Cryptographic Module (eCM) tokens. An eCM is a Spyrus Rosetta USB cryptographic token that stores a key. The token must be inserted for certain sensitive functions within BOSS, Ballot Now, Rally, Tally, and SERVO to operate. eCM Manager writes a key identifier and the generated signing key to each eCM token, and sets up a 6-digit personal identification number (PIN) to access the token. eCM Manager operates on a PC running the Windows 2000 Server operating system.

6. **SERVO:** The System for Election Records and Verification of Operations tracks inventory for all election devices used within a county; all serial numbers for devices are stored in this application, as is the firmware version associated with the device. In addition, SERVO installs cryptographic keys on JBC and eScan devices and resets their memory prior to the beginning of an election, backing up the memory, CVRs, and audit logs for these devices to a database. SERVO operates on a PC running Windows 2000 Server and connects to an eScan over Ethernet, and to the JBC with a parallel cable.

17.2.2 Components in Use at Polling Locations

The following components make up the Hart precinct voting system. Please refer to Figure 17.1 for a pictorial representation for how the components interact.

7. **MBB:** A Mobile Ballot Box is a PCMCIA card that stores information such as ballot definitions and a set of cast ballots. It is the primary means of transporting election information between components

within the Hart system. Ballot definitions are transported through an MBB from BOSS to Ballot Now, where it is used to generate ballots. MBBs generated from BOSS are also used in the JBC and eScan devices for collecting vote information, which are subsequently tallied using the Tally server.

8. **JBC:** The Judge's Booth Controller is a console that can control up to 12 eSlate DREs attached to it. The JBC generates voter access codes and distributes ballot configurations to the eSlates; it also records the Cast Vote Records (CVRs) and stores eSlate ballots to internal memory. In addition, ballots are written to an MBB. The JBC is capable of accumulating and reporting vote results. It connects the eSlates in a dumb-terminal fashion with an RS-485 cable connecting the HD15 interface on the eSlate to the DB9 port on the JBC.

9. **eSlate:** This is the direct-recording electronic (DRE) voting unit used in a Hart-run precinct. It attaches to a *Verified Ballot Option* (VBO) printer known as the *VBOx* to produce a voter-verified paper audit trail (VVPAT). The eSlate may also contain a Disability Access Unit (DAU) with magnified text rendering for vision-impaired voters and oversized "jelly switches" for voters with tactile impairments. In all units, instructions and confirmations are given through an audio track as well as visually on screen.

10. **eScan:** This is a precinct-based optical ballot scanner for voters who choose to vote using paper ballots (currently all non-disabled voters in Ohio). The eScan scans and tabulates votes based on the ballots deposited into it by the voters, and contains an MBB used to store tabulated vote results.

17.3 General Election Procedure

Here are the general steps used to conduct an election using the Hart system:

17.3.1 Pre-Election Procedures

- The eCM manager generates a cryptographic master key, and stores a copy of this key on each eCM to be used during the election. A PIN is required to store the key.

- BOSS creates an election database based on data sourced from the back-end Sybase database. This includes precinct and race definitions, ballot definitions, and other options, such as party primaries, to create the right number of ballot variants. An eCM token must be plugged into the machine where BOSS runs for the generated ballots to be accepted.

- BOSS is used to "burn" or write MBBs. The number burned is dependent on the county size and the number of JBCs and eScans in the area. An audio card (also a PCMCIA card) for each eSlate with a DAU is also burned by BOSS at this time. An MBB is burned for specific use in Ballot Now; it cannot subsequently be used in any other component or an error message will result. BOSS tracks the number of burned MBBs. Once all MBBs have been burned, the election may be finalized. After this time, no more MBBs can be burned, and no election configurations can be changed.

- Ballot Now takes an MBB generated by BOSS and generates electronic images of the ballots. These can be printed with a standard laser printer at county election headquarters or the images may be sent to commercial printers for ballot generation.

- In the warehouse where voting equipment is kept, SERVO is used to reset the memory of all JBCs and eScans and to reset the vote count to zero. When an eCM token is plugged into the machine running SERVO, the cryptographic master key for the country may be installed into the JBCs and eScan devices.
- MBBs are installed into the JBCs and eScans and are now ready for polling. The devices are tamper-sealed at the warehouse. The MBBs may also be transported to the polling locations to be installed into the eScans and JBCs there.

17.3.2 Election Day Procedures

- Election administrators put passwords into JBCs and eScans to start them. A separate password is used to open the polls, after a zero-tape is printed from each device.
- During an election, voters using eScan machines who have registered receive paper ballots, to be filled out with blue or black ink. These ballots are placed in the eScan machine. If the ballot is over-voted (i.e., two candidates are chosen in a race where only one is allowed), the ballot will be returned to the voter and invalidated. The voter can then fill out and submit a new ballot.
- In Ohio, disabled voters are able to use eSlate DRE machines. When they have registered, they will proceed to the poll workers staffing the JBC management station. Each voter will receive a 4-digit code. The voter proceeds to the eSlate where they enter the code and choose candidates, using the on-screen instructions for guidance. eSlate machines with a DAU installed will provide audio narration for the voter. A VBO printer is attached to the eSlate, which provides a paper confirmation of the voter's choices. When the voter casts their ballot, the paper is advanced out of their view. A voter has three chances in Ohio to cast their ballot in case they cancel their ballot choices.
- At the close of the election, election administrators enter passwords into JBC and eScan machines to close the polls.

17.3.3 Post-Election Procedures

- After the election, MBBs from the JBC and eScan units are either physically taken to central headquarters where they are either directly loaded into Tally or loaded into a machine running Rally, which is set up in Ohio to communicate with Tally over an internal private network. (Rally requires an eCM to be inserted for MBBs to be read.)
- Paper ballots, such as absentee ballots [1] precinct, are transported to a central facility and read by a specified central count optical scanner, with the image recognition done by Ballot Now. The results are written to an MBB and sent to Tally for tabulation.
- An eCM token is inserted into the machine where Tally runs, which upon receiving all of the MBBs, tabulates the votes, using the token to ensure no tampering took place with the results. Tally produces an election result database and a variety of reports.
- SERVO backs up the CVRs and audit logs from the JBC, eScan, and eSlate units. These can be used to produce recount MBBs to be recalculated in Tally if necessary. The firmware of all devices can also be verified by SERVO to ensure it is the same as when the machines were sent out.

[1] It also is possible that paper ballots may be used in a precinct without being processed by an eScan. In this case, the ballots will be transported to a central facility and processed by Ballot Now.

17.4 Data Security

The central mechanism used in the Hart system to provide data security is a single county-wide cryptographic key (a secret value, similar to a password, shared throughout the county). This key is generated by eCM Manager and is later programmed into eCM hardware tokens (similar to USB "thumb drives") and all precinct devices, prior to sending them to the field. The back end systems (BOSS, Ballot Now, Tally, and Rally) require an eCM token be present before allowing access to critical functions. The key is read from the token and into the computer's main memory.

During the preparation for an election, the key is read by SERVO and placed in the memory of the eScan and JBC during the "reset" operation (in preparation for an election). BOSS prepares MBBs for use in an election by pushing ballot data onto the memory card and digitally signing[2] it. The MBBs are either sent to the precints on the day of the election, or placed in the JBCs and eScans for use in the election and shipped to the local precincts—sometimes being stored in insecure locations overnight.

The digital signatures of the MBBs are checked at the time that the parent JBC or eScan is booted (typically the morning of the election), and rejected if they are incorrect. The polls are opened and the MBBs begin receiving voting and audit data. When the polls are closed, the MBB is signed using a copy of the key in the parent device's memory. The JBC and eScan keep counters of the number of votes cast in internal flash memory. One of these counters is maintained over "reset" operations.

The MBBs and associated VVPATs are returned to the county election headquarters after the polls have closed. In Ohio, Tally is used to tally all of the votes on all of the MBBs whose digital signatures are valid. Recounts are mandated by statute in extremely close races. The tabulated votes are inserted into a database on the Tally host. Election result reports are generated from the Tally database. 11 to 15 days after the election, a canvass is held to validate and verify the results. [3]

17.4.1 Other Components in the Hart System

Hart has written a number of additional utilities that may be used in a jurisdiction. They include the following:

- **Fusion:** This utility is used to consolidate multiple sources of tallied report information. Descriptions from the Hart website state that "The Fusion software application is a flexible tool that provides election results integration and enhanced reporting options. Fusion can be used to combine election results from different Tally databases, or to combine Hart Voting System election results with results from third-party systems. In addition, Fusion can import results from election databases to provide enhanced reporting options." [4] It has not been certified for use in Ohio, but appears to be used in some jurisdictions. We were not given access to either the application or its source code.

- **InFusion:** This utility is used to retrieve information from multiple sources when creating ballot definitions. It is not certified for use in Ohio and we are unclear as to whether it is used in any jurisdictions. We were not given access to either the application or its source code.

- A utility called **Bravo** exists but we do not have any information about it. The only reference to it is

[2] In reality, this is not technically a digital signature, but a hashed message authentication code.

[3] See ORC Ann 3505.32 for more information on the canvassing procedure in Ohio.

[4] `http://www.hartinc.com/pages/171`

in the *eSlate Reference Manual* but it is not described there nor anywhere else. This software is not certified for use in Ohio and we are unaware of whether it is used in any jurisdictions.

- The Hart JBC is able to connect with **Voter Registration** systems, but there is no information about what these may be, whether there is a system used by Hart, or any other information. No Hart-branded voter registration systems are certified for use in Ohio and we are not aware of any jurisdictions where any of these systems may be used.

CHAPTER 18

HART SYSTEMIC AND ARCHITECTURAL ISSUES

This report focuses on the ability of the studied system to conduct an election without fear of manipulation of its outcomes, processes, or public perceptions. As stated in the executive summary, we have found substantial flaws that inhibit the ability of the Hart system to meet these goals. Many of the issues presented in the following chapters relate to functional or procedural problems embodied in the Hart design and implementation. Such problems undermine the security of the system and provide attacker-accessible avenues to critical election data and systems. This chapter broadly characterizes the report's findings by highlighting the four areas of most concern:[1]

- **Ineffective data, software, firmware, and communication protections** - the majority of security mechanisms used to ensure data and software integrity can be bypassed by an attacker.
- **Ineffective authentication** - all user and system authentication methods (passwords, hardware tokens, cryptographic keys, etc.) can be bypassed or otherwise subverted, often trivially, by an attacker.
- **Undocumented, unprotected, and unsafe features** - the Hart system contains many unsafe and frequently undocumented features that are enabled through simple and available configuration interfaces.
- **Design, development, and maintenance Issues** - The pervasive lack of sound software and security engineering practices is the source of ubiquitous security, reliability, and maintenance issues.

Our findings are fully consistent with previous studies. In particular, we found the architectural and systemic findings of the Hart Source Code Report of the California TTBR study to be universally true. Here, we extend those assertions with our own, and attempt to more generally characterize the specific classes of security issues within the Hart system.

Note that the descriptions below are simply a sampling of the problems that arise from confirmed and newly identified issues. There are many other attacks resulting from these and other areas that can have serious negative consequences for perceived and real election validity.

[1] **Style note**: This chapter references numerous issues raised in the following chapters as demonstrative or enabling of some broader concern. These issues are denoted by reference to the chapter/section in which they are defined. For example, issue 19.2.1 discusses the use of a county-wide eCM key, and is be referenced as "(19.2.1)". Readers are directed to the referenced section for in-depth detail of the issue, its impact on the system, and procedural or technical mitigations, if any exist.

18.1 Ineffective Data, Software, and Firmware Protections

The mechanisms which Hart uses to protect data and software are frequently based on absent or flawed security models. The failure of these mechanisms to address important threats results in broad exposures of election data and features to an attacker. Such failures are a result of unrecognized requirements and other design and development issues. In most cases these issues cannot be addressed via software upgrades, but call for rethinking of both technical designs and procedural practices. We detail some of these central protection issues below.

18.1.1 Data

Much of data security in the Hart system flows from the single 32-byte key that is distributed throughout a county (19.2.1). This is poor security design–compromise of any one of potentially hundreds of devices is sufficient to subvert an entire county. Here the Hart design violates a basic *isolation* tenet of security engineering: compromise of a single precinct provides materials to compromise any precinct and the election headquarters. Further, if such compromise occurs, it will be impossible to identify which precinct is responsible for the breach.

In truth, obtaining the county key is trivial to an attacker with only a few seconds of physical access to precinct or back-end equipment–they can download the key from the eCM manager into a file (19.2.2) or extract the key from an eScan's memory via the unprotected Ethernet port (19.1.7), or by other means.

Once an attacker has the county key, they can forge any election data they want (19.1.7)–such modification would only be detected by careful comparison with the relevant VVPATs or physical ballots. However, as the VVPAT is also forgeable (20.5.6), this countermeasure is only of limited use. If both the MBB and VVPAT/completed ballots are forged, then there will be no way of detecting the forgery short of studying the internal audit information in each eScan and JBC used. This audit information itself can be erased (20.4.2) by an attacker with physical access to the device. These issues can lead to undetectable precinct-level forgery of an election (see Chapter 21.3).

The integrity and secrecy of election and audit data is protected on the back-end EMS servers in databases. Sometimes this data is stored in encrypted format, and sometimes it is not (19.1.3). Further, the passwords for the EMS applications and the databases they use are available (19.1.2) and easily bypassed (20.1.3). Once database access is gained, the attacker can modify any data they want, potentially manipulating election results or audit data. From an information security standpoint, these issues render the data held in server databases as functionally unprotected.

18.1.2 Communication

Hart is a networked system. The back-end and polling place devices speak to each other over serial, parallel, Ethernet, and other computer-to-computer interfaces. Such communication is inherently dangerous, as an attacker can masquerade as a remote party (e.g., pretend to be SERVO to a JBC) or "listen and modify" to the communications passing between the devices (e.g., modify commands passing between a JBC and an eSlate). It is an obvious and *essential* requirement that device-to-device communication be protected.

With limited exception, Hart provides no device-to-device communication security. This exposes critical data and functionality to the attacker. For example, and attacker may generate voter codes (20.4.1), upload

firmware (19.4.1), and erase voting or audit data (20.4.2). Where Hart does provide secure communications–Tally to Rally communications (over an internal network as mandated in Ohio) and for communication between Ballot Now and a security database–it uses a flawed protocol implementation (19.9.5).

18.1.3 Software and Firmware

The Hart software and firmware internal validity checks, where present, are ineffective at detecting compromise, either because they are designed such that they can never fail (19.7.1), or they can be trivially spoofed (19.7.2, 19.5.4). In the case of the eScan, the entire firmware can be replaced by an attacker with unobserved access to the eScan for 60 seconds (20.3.1). This latter attack would allow an attacker to completely alter the election results recorded on the MBB.

18.2 Ineffective Authentication

Authentication verifies a user, computer, or program's identity. Such identification is needed to ensure that only authorized parties perform security sensitive operations, e.g., create a MBB. We have found that every authentication mechanism in the Hart system is circumventable. The authentication methods include:

- **Hardware tokens** - Spyrus Rosetta hardware tokens store the county-wide keys, and are used in part to authenticate users of the Tally, BOSS, Ballot Now, eCM Manager, and SERVO applications. Tokens must be inserted into the Hart server before certain critical functions are enabled, e.g., the creation of new MBBs. These tokens can be created by anyone with access to the county-wide key (19.2.1). The Spyrus tokens are commercially available, and thus attackers can purchase them and create forged tokens by installing the key to them. The keys can be obtained either via download from eCM Manager (19.2.2), read directly from an eScan (19.1.7), or extracted from another token (20.2.1).

- **Passwords** - All Hart back-end EMS applications request a password when they are started. The cryptographic hashes of these passwords are stored in security databases, which are themselves easy to read and modify (19.1.3, 19.1.4). These databases are protected by passwords which are obfuscated using a trivially reversible technique (19.1.2). So, an adversary with access to the computer can read or create passwords to gain access to the server applications. Further, the applications all implement a flawed password policy that permits the user access if no password data is found in a security database (20.1.3) (after prompting a user for a new user/password).

- **JBC/eScan Access Codes** - MBBs contain the access codes used to enable administrator operations such as opening and closing the polls on the eScan and JBC. The MBB is not encrypted, so the codes can easily be read by an attacker (19.1.7).

- **Voter Codes** - Voter codes are used to authenticate voters before they can vote on an eSlate. The JBC generates the first voter code at random, then repeatedly uses a simple mathematical function on that code to generate each subsequent voter code. This process is perfectly predictable; an attacker who knows any voter code can determine all voter codes that follow it (and preceded it) in the order in which they will be assigned (19.7.6).

18.3 Undocumented, Unprotected, and Unsafe Features

The Hart system consists of many thousands of lines of code distributed over a large number of applications. The code was developed over nearly a decade by an apparently revolving staff of software engineers, based on the number of different authors and styles of coding encountered. A byproduct of this process is that there is a large number of old, unused, and otherwise "orphaned" features built into the software. Such features are often a source of security issues–they can be enabled in subtle ways. We review several of the more potentially dangerous of these functions below.

Note that there are likely many features in the Hart system that the review team was not able to identify–given time constraints, we were only able to understand a tiny fraction of the source code. Thus, we recommend that future development efforts exhaustively audit the code for such features.

18.3.1 Autovote

The Hart system provides an "Autovote" feature as part of the Ballot Now back-end server application (20.7.2). This feature allows election officials to print in bulk eScan ballots whose votes are pre-cast (or alternately that are not pre-cast). A report describing an expected tally of the printed ballots is also provided. The ballots are only accepted when the eScan to which these Autovote ballots is fed are in "test" mode. Each ballot has the words "AUTOVOTE" printed on the side making it visually distinct from others. Autovote is not enabled by default, but is only available when a combination of registry entries is set to specific values.

Autovote is not mentioned in the current product manuals or other provided technical documentation. It was briefly mentioned in a manual from 2001 (this reference was subsequently removed) and discussed in several hearings on voting systems in 2001 and 2003.[2]

Other security issues arise when a specific registry entry is set. When set, the ballots that are created will be accepted in the "election" mode—thereby allowing the ballots to be counted in a live election. Furthermore, if the registry entry is set when using Ballot Now to process ballots from a high speed scanner, Autovote ballots created without the override being set will be accepted as legitimate. Obviously, this is a very dangerous feature–an attacker (whether insider or outsider) who has access to the Ballot Now server and a precinct scanner could fraudulently generate vote tallies that would be accepted in a real election. It also may be possible to change the "AUTOVOTE" banner to some other text, including nothing, so that Autovote ballots would not visibly appear distinct from legitimate ballots.

Autovote is obviously test apparatus. Performing large-scale election testing requires the generation of ballots to scan. However, its inclusion in a production product is extremely dangerous.

18.3.2 Windows Registry Misuse

The Windows registry is a operating system service that maintains configuration parameters for applications installed on the computer. The Hart system makes extensive use of the registry to enable/disable features of the system. While in general the use of the registry is not a problem, Hart uses it to enable critical functions and security sensitive operations. Issues arise because anyone with the appropriate privileges on the computer can read and change the registry. Thus an attacker without any Hart system passwords or

[2] State of California, *Meeting: Secretary of State Voting Systems Panel*. December 16, 2003 ⟨URL: `http://www.sos.ca.gov/elections/vsp_min_121603.pdf`⟩.

hardware tokens can affect the security and behavior of the system. Examples of registry misuse include enabling Autovote for live elections (20.7.2), defining completely the available menus in Tally (20.6.2), and storage of the TLS password key used by Ballot Now (20.1.2).

An interesting characteristic of the registry use in the Hart software is that it (generally) periodically checks registry entries, rather that just checking them at start-up. This has the consequence that triggered features can be turned on and off without restarting the software.

We found references to many dozens or more of registry entries used by the Hart EMS applications. We were only able to investigate a small number of these. The vast majority of registry entries are undocumented, and their purpose is often unclear. An inventory of these items and their function should be constructed and documented for future certification and evaluation activities.

18.3.3 Remote eScan access

There is a server that allows users to remotely login to the eScan device (20.3.2) via the network connector (an unsealed RJ45 Ethernet port on the back of the eScan). This login capability is available before, during, and after an election. The eScan allows remote access to a local Windows `telnet` server. The Windows `telnet` has a long and checkered history of buffer overflows and other vulnerabilities. This represents a attractive target for an attacker to gain access to and compromise the eScan device. It is unclear what the remote access function is intended to be used for.

18.3.4 Adjust Vote Totals

The Tally application allows an attacker to directly change vote tallies through an edit screen available from the application menu (19.9.1). An entry is noted in the audit log. Subsequent election result reports provide no notification that the tallies were modified. This obviously represents a conduit for misuse by an attacker–particularly for a malicious insider. This appears not to be an orphaned function, but was intentionally included to enable local officials to correct election errors.

An attacker who misuses the adjusting vote totals interfaces can cover his tracks by forwarding the election reports, then restoring a previously un-adjusted database. Note that the frequent backing up and restoring of database is a common and vendor recommended operation. The restoration of the database effectively erases the adjustment from the local audit logs. The attacker can also modify the audit log directly (20.1.4).

18.4 System Design, Development, and Maintenance Issues

The lack of sound software and security engineering practices leads to pervasive quality and security problems. We briefly highlight some of the engineering quality problems encountered during the review.

- **Inadequate Low level Design** - The Hart system did not consistently apply an observable design methodology. Standard top-down abstraction and modularization practices were not followed, leading to convoluted, excessively complex, and sometimes circular flows of control within a program. The mixed use of programming languages, while common in commodity software, was improperly managed. This further complicated the design where the use of objects in C++ and C functions was

uneven, functionality was often duplicated, and relationships between data and user interfaces were not often clear or consistent.

- **Inadequate Coding** - The coding errors were universal. Buffer overflows, `printf` errors, and other violations pervade the code (19.1.12, 19.3.1). Note that these problems are often subtle and very difficult to identify after the fact. Excising these errors from a large system is an enormously time-consuming and difficult task. Standards for source code safety should be established and documented, and the source code should be audited for compliance.
- **Maintenance** - The uncontrolled evolution of the system is evident from the code. Features were created and abandoned, interfaces were changed in some parts of the system but not others, and many "hacks" (code fixes that quickly solve a problem, but do so in a suboptimal way) were added to the code. Such modifications accelerate the aging of the system and can incrementally make the system less stable and secure.

The lack of sound practices will seriously undermine any attempt to definitively verify the security of current and future versions of the software. The issues introduced by poor practices are not likely to be encountered in simple testing procedures, as they are typically found only by carefully crafted inputs and environments. Such inputs and environments are only identified through careful and lengthy analysis. However, the verification process is heavily stacked against the reviewers; they need to find all possible vulnerabilities (of which there maybe hundreds or more), whereas the attacker only needs to find one that the validation engineers didn't.

18.5 Audit Procedures

A fundamental and critical requirement for a complex system such as election management is the ability to audit every element of the system. Audit logs serve a vital purpose, as they can alert an auditor of suspicious or uncommon events that occurred, which could indicate the presence of malicious intent against the system. Because of this, it is critically important that an auditor may be completely confident that information retrieved from audit logs is complete and accurate. If an attacker can modify or destroy an audit log, it has no value. Without auditing, it is impossible to be confident that the outcome of an election is legitimate.

All of the voting equipment used in the Hart InterCivic System keeps an audit log. These are stored in different ways depending on the equipment in question. With the exception of the eCM Manager application, all of the back-end EMS software applications (i.e., BOSS, Ballot Now, Rally, Tally, and SERVO) have their own audit logs, stored in a database file associated with each application. During an election, the MBBs inserted into eScans and JBCs, and those written by Ballot Now as it processes absentee ballots, will store copies of audit logs. As a measure of redundancy, the eScan optical scanner stores an audit log inside its internal memory. In addition, the JBC and any eSlates connected to it will all store a copy of the audit log in their internal memory. Because of the VVPAT requirement in Ohio, a paper record is kept of all eSlate votes to be used in case of a recount, while the paper ballots processed by the eScan provide a physical record of cast votes. Note that the VVPAT log can be disabled via software.

All of these logs are subject to manipulation and deletion. The logs of all software applications can be modified by accessing the databases in a simple fashion (20.1.4). The JBC and all associated eSlates can have their audit logs erased from their internal memory by a malicious voter with a handheld computer (20.4.2) and eScan machines are similarly vulnerable (20.3.7). The audit logs on the MBBs in these machines may

be cleared at the same time the internal memory logs are cleared. The VBO printer attached to eSlate machines can be easily opened and the paper record removed (20.5.4). Finally, the paper ballots processed by the eScan can be damaged (20.3.5) and the integrity of the count may be compromised by stuffing the unit with duplicate ballots (20.3.6).

Taken in isolation, each of these attacks may seriously affect the auditability and ultimately, confidence in the election process. With just a small number of well-placed insiders, however, or a combination of insiders and malicious outsiders, it is possible to compromise logs at the polling places and at election headquarters, resulting in a catastrophic loss of verification and accountability for the county. As every piece of the system is vulnerable to attacks against audit logs, there are insufficient protections within the Hart voting equipment and software to prevent a motivated adversary from compromising an entire election.

CHAPTER 19

HART ISSUE CONFIRMATION

The sections of this chapter detail our confirmation of the issues raised in the Hart Source Code report developed as part of the California TTBR study. The sections are organized by component, and provide a brief summary of each issue raised, detail of the substance and potential exploitation of the issue, the means by which the issue was verified, and a strategy for the mitigation of the issue (if any level of mitigation is possible). The public version of this report contains references to sections of the private report. These redacted sections contain confidential information identifying where in the source code or design the issue is manifest, or other vendor-confidential information. Note that all source code, file names, and other apparent vendor specific information in the public version have appeared in previous election study analysis reports.

We were able to confirm 35 of the 36 issues raised in the original California TTBR report. The only exception was issue 14. In that analysis, we were able to identify a dangerous software function (`crc16`), but could not find a means of exploiting its design to attack the system in the short period we allotted to its investigation. However, because we could not view the private version of the California report, we did not have a means of evaluating the specific issue uncovered by the California reviewers. The Hart code is very complex, so it seems likely that the issue exists in the code, but the lack of usable detail prevented us from finding it.

19.1 Hart General Vulnerabilities

19.1.1 Corrupt MBBs can cause Tally to crash

The original California TTBR report suggests that Tally could be crashed by manipulating the data in the MBB in a specific way. The idea was to indicate via internal data that the card was larger than it was in reality. Tally would then process the MBB as if it were larger than it was, and crash as a result.

Our evaluation *cannot conclusively confirm or deny the issue*. We did review an exceptionally unsafe module in the source code that could be abused if the wrong information was sent to it. However, we could not immediately find a way to exploit this code. Because we could not view the private version of the California report, we did not have a means of evaluating the specific issue uncovered by the California reviewers. The Hart code is very complex, so it seems highly plausible that the issue exists in some form in the code.

Description: We reviewed the source code that reads the internal MBB data structures. The original report suggests that the CRC checks of the internal data blocks containing the cast vote record (CVR) and audit logs could be manipulated into reading memory regions outside the (MBB) addressable space. This would

cause invalid reads that would cause Tally to crash.

The checks for the CRC values do appear to prevent reading outside the ranges. Not only did we check these regions, but we also checked internal CVR and audit log entry CRCs, but found no case where the CRCs could be manipulated, i.e., the proper length checks were directly or indirectly made. Thus, we could not confirm that the issue indeed exists as stipulated in the original report.

We were able to confirm that the CRC function itself is unsafe; it provides only minimal checking. Thus, because it is used extensively throughout the Hart system, it is likely that such a design could be exploited to crash or otherwise subvert the Hart components.

This confirms Issue 14 reported in the California TTBR Hart source code report.

Prerequisites: Access to an MBB and control over the Tally process.

Impact: Allows the attacker to insert fraudulent votes into the voter tallies.

Procedural Mitigations: None.

Verification: This vulnerability was assessed through source code analysis and via demonstration.

19.1.2 Database passwords are stored insecurely

Boss, SERVO, Ballot Now and Tally store database passwords in configuration files on the file system. These files may be read by any user on the system.

Description: The Hart EMS applications (including Boss, SERVO, Ballot Now and Tally) store election data in databases. A username and password are required to connect to these databases. These usernames and passwords are stored in files in the same directories as the databases. An attacker able to extract these passwords can connect to the EMS databases and read and modify election, audit log and security data.

This confirms Issue 15 reported in the California TTBR Hart source code report.

Prerequisites In order to exploit this vulnerability, an attacker will need physical access to a Hart EMS system.

Impact: Possession of the username and password stored in the configuration files allows an attacker to connect to a database and read, modify and delete election data as well as EMS usernames and password hashes. Such abilities can be used to modify ballot definitions without knowing the usernames or passwords needed to login to BOSS or Ballot Now.

Procedural Mitigations: Restricting physical access to the Hart EMS systems reduces the opportunities for unauthorized users to connect to the EMS databases. Ensuring that the EMS systems are never connected to a network will also reduce such opportunities.

Verification: This issue was confirmed through source code analysis and via demonstration with the EMS databases. We located the configuration files on the EMS machines and extracted the passwords. We then used these passwords to connect to the EMS databases.

19.1.3 The EMS databases are not encrypted

The databases for BOSS, Tally, Ballot Now and SERVO are not encrypted, allowing election data to be read by any user who can open the database files associated with these applications.

Description: BOSS, SERVO, Ballot Now and Tally all store election data in databases. These databases are stored as files on the file system. Because they are not encrypted, they may be read and modified by any user on the machine.

This confirms Issue 18 reported in the California TTBR Hart source code report.

Prerequisites In order to exploit this vulnerability, an attacker will need physical access to a Hart EMS system, and to have installed a binary editor on the machine for inspecting or modifying the databases.

Impact: Because the EMS databases are not encrypted, an attacker that can open the database files can read, modify or delete election data or ballot definitions, without knowing the usernames or passwords needed to login to BOSS, Ballot Now, Tally or SERVO. This can be accomplished by opening the database in a binary editor.

Procedural Mitigations: Restricting physical access to the Hart EMS systems as much as possible reduces the opportunities for unauthorized users to open the EMS databases. Ensuring that the EMS systems are never connected to a network will also reduce such opportunities.

Verification: This issue was confirmed through source code analysis and via demonstration with the EMS databases. We connected to the databases as described in Issue 19.1.2, and read cleartext election, audit and security data.

This vulnerability was verified by connecting to the databases using the passwords extracted as described in 19.1.2, and reading cleartext election data.

19.1.4 New Users can be inserted into the database

New users with elevated privileges can be inserted into the Hart EMS databases. This allows existing users to escalate their privileges.

Description: It is important to differentiate between the usernames and passwords needed to connect to the EMS databases, and the passwords used to login to the Hart EMS applications (Ballot Now, BOSS, Tally and SERVO). A user needs the application username and password to log into the applications, while the applications use the database username and password to connect to the databases where information specific to the application (e.g., election parameters and ballot definitions in BOSS) is retrieved from. The usernames and passwords used to connect to the databases are stored in configuration files, as discussed in Issue 19.1.2. The usernames and passwords used to login to EMS applications are stored *in the EMS databases themselves*.

The Hart EMS systems use databases to store user credentials. These credentials are used to verify user identities at login. For each user, the following are stored in the database.

- Username - the name the user enters at login time
- Salt - a number used in the hashing of a password
- Password Hash - The result of a hash function performed on a password and salt

- Privileges - A number representing the privileges a user has within an application based on the username and password supplied

The method by which the password hash is generated is publicly known and well-specified,[1] and can be easily implemented in software.

Because the database passwords are not securely stored, as described in Issue 19.1.2, it is possible to connect to the databases and add usernames, salts, and hashes, allowing existing users to login to the EMS programs Ballot Now, BOSS, Tally, and SERVO with elevated privileges.

This confirms Issue 19 reported in the California TTBR Hart source code report.

Prerequisites: In order to exploit this vulnerability, an attacker will need physical access to a Hart EMS system, and to have connected to an EMS database.

Impact: An attacker able to add users to the database can create accounts using existing password hashes with escalated privileges.

Procedural Mitigations: Restricting physical access to the Hart EMS systems as much as possible, reduces the opportunities for unauthorized users to connect to the EMS databases. Ensuring that the EMS systems are never connected to a network will also reduce such opportunities.

Verification: This vulnerability was verified via demonstration. We started the database, and added new users with existing passwords, we then successfully logged in as those users.

19.1.5 Back-end Windows systems may be insecure

The Hart EMS systems, Ballot Now, BOSS, Tally and SERVO, run on the Windows operating system. An attacker exploiting a Windows vulnerability may be able to subvert these EMS systems.

Description: It is not clearly spelled out in the Hart documentation how Windows is to be configured for security on the EMS systems. The system provided by Hart had several unnecessary services turned on including remote registry editing and NetBIOS. We received machines configured with the EMS software from Hart, and verified that they ran Windows 2000 Service Pack 4 with the Update Rollup. We ran tools that showed that these machines contain a vulnerability in Microsoft Outlook Express and Windows Mail deemed "Critical" in severity by Microsoft (issue 941202).

In addition, the generation of signing keys for eCM tokens relies extensively on the `CryptGenRandom` function called by the random number generator in Windows 2000. Recent security research shows that this generator contains vulnerabilities; namely, it is possible to find previous states of the generator in about 19 seconds on a Pentium IV computer, thus allowing an attacker to determine previous signing keys. Future keys can also be determined because of the lack of both *forward security* and *backward security* in the Windows 2000 random number generator.[2]

The California source code review of the Hart system documents that Hart has procedures in place mandating that only Hart personnel may provide maintenance updates to the Windows operating system.[3] It is unclear

[1] D. Eastlake and P. Jones, *US Secure Hash Algorithm 1 (SHA1)*. Internet Engineering Task Force RFC 3174, September 2001.

[2] Leo Dorrendorf, Zvi Gutterman and Benny Pinkas, 'Cryptanalysis of the Random Number Generator of the Windows 2000 Operating System'. In Proceedings of the ACM Conference on Computer and Communications Security. Alexandria, VA, November 2007.

[3] Srinivas Inguva et al., *Source Code Review of the Hart InterCivic Voting System*. University of California, Berkeley un-

to us whether similar mandates are in place in Ohio. Regardless of the answer, there is a potential for significant difficulties. If election administrators install updates themselves, then the servers are vulnerable to inadvertent (or malicious) installation of third-party applications. In addition, it may be the case that operating system patches have unintended consequences on applications, causing them to stop functioning; an example of this was a patch for Microsoft's Internet Explorer application that caused client software from the Siebel enterprise CRM software to become unusable.[4] Election administrators may be nervous about installing patches in case a similarly unintended consequence of patch application causes Hart software to stop functioning. However, in the time needed for a patched operating system to undergo certification, the operating system, and hence the applications that run on top of it, are vulnerable to exploits.

This confirms Issue 20 reported in the California TTBR Hart source code report.

Prerequisites: In order to exploit Windows vulnerabilities, an attacker would need physical access to the EMS machines. The particular vulnerability listed requires the use of Outlook Express or Microsoft Mail, [5] which would require network access that should not be possible on these machines for procedural reasons; however, there are no technical reasons why this access could not be obtained (e.g., a USB wireless dongle could be inserted easily on the machine).

Impact: The integrity of Hart EMS applications and the election data they operate on can be compromised by compromising the underlying Windows operating system. While the current documented Windows vulnerability requires access that should not be available, the greater concern is that EMS machines need to be patched when exploits are announced, as it may be possible that some future exploit is able to cause a compromise on the machine without any additional network access.

Procedural Mitigations: Physical access to the machines running the Hart EMS applications should be restricted to only election administrators. No network interfaces should be active on the machine, and the usage of these machines must be closely supervised. In addition, either election administrators or Hart technicians must be vigilant and respond quickly to patch machines that are susceptible to newly-discovered exploits.

Verification: We found the security vulnerabilities by running the Microsoft Enterprise Scan Tool on the Hart machines. The results from this showed that the systems were susceptible to vulnerability 941202, addressed by Microsoft Security Bulletin MS07-056.

19.1.6 Election management computers may be connected to the Internet

All of the Hart back-end systems can potentially be connected to the Internet, either directly or through some other computer within a election headquarters-local network. This opens up the computer to any number of well known attacks against the operating system. The attacks could corrupt the election software or data, or introduce other malicious software that may corrupt later elections.

Description: The Ohio Revised Code 3506.23 mandates that "A voting machine shall not be connected to the Internet." This apparently does not apply to local area networks, and what constitutes a voting machine is unclear. Several counties may keep the back-end computer equipment on local area networks. However, a

der contract to the California Secretary of State, July 20, 2007 ⟨URL: `http://www.sos.ca.gov/elections/voting_systems/ttbr/Hart-source-public.pdf`⟩.

[4]Microsoft Corporation, *Microsoft Security Bulletin (MS06-013)*. October 2007 ⟨URL: `http://support.microsoft.com/kb/912812`⟩.

[5]We discovered a copy of Outlook Express runnning on the vendor-configured machine where the EMS applications were installed.

2005 Secretary of State directive[6] provides further guidance on the use of networking. This directive states that no "Election Information System" is permitted to connect to any computer network or to the Internet, including for (1) setting up elections; (2) defining ballots; (3) receiving voted data from polling places; or (4) tabulating data related to ballots or votes.

Two exceptions are (1) using local networks for creating or uploading memory cards, ballot definitions, precinct results, etc. and (2) to download software upgrades, repairs or updates (board of elections has to apply for a waiver). Under no circumstances may a polling place device be connected to an outside polling place. Modems can be used to network devices if the proper security plan is developed and permissions allowed.

While these policies are a positive step toward preventing network-based attacks, they fall short in several ways. First, modems are simply a way of networking devices. Thus, the differentiation in policy between "network" and "modem" use seems arbitrary. Any attack that could occur as a result of a network attack could also occur via modem, with the only difference being that the modem-connection is often considerably slower than wired or wireless network communications.

Second, *any* exposure of a computer to the Internet, even through other computers on the same network, is dangerous. There are many attackers on the Internet with tools that constantly scan for vulnerable hosts. It is likely that an unprotected host could be compromised within minutes of being connected to the Internet—often before it has time to download and apply the critical software "patches" that would have protected it from the vulnerability. Thus, exposing computers to the Internet even for a short period to perform maintenance is imprudent policy.

This confirms Issue 21 reported in the California TTBR Hart source code report.

Prerequisites: An attacker needs access to the Internet through which they may exploit vulnerabilities.

Impact: An attacker with network access to an EMS machine can significantly or completely determine election results by modifying the EMS applications themselves.

Procedural Mitigations: Election computers and any network to which they are connected should never be connected to the Internet. Special dispensations allowing any such connections should be eliminated. Distinctions between modems and networks in policies should be removed.

Critical system updates to the software should be certified, placed on CDs, logged, and physically installed on the computers. Note that the use of CDs *does not* prevent an attacker from corrupting the system, but only serves to prevent an attacker from remotely attacking or observing the results of compromise, i.e., this "air-gap security" is not a panacea, but prevents a wide array of common attacks. Thus, enormous care must be taken when updating election computers, and only where there is a clear and critical reason for doing so.

Verification: This issue was confirmed via discussions with Ohio election officials and through review of the relevant Ohio Revised Codes.

Note: This issue does not have any unredacted content. There is no corresponding section in the private appendices.

[6]Ohio Secretary of State. (2005). Directive 2005-23: Telecommunications: General Internet Access and Networking, Downloading Software and Modem Access.

19.1.7 eCM keys are extracted and stored insecurely

Keys used for validating the integrity of received MBBs are stored in the memory of the JBC and eScan without encryption (i.e., in cleartext). The memory of these devices can be retrieved by an attacker, and the keys can then be extracted from the memory.

Description: The eCM signing keys, used for computing HMACs for data on MBBs, are uploaded to eScan and JBC devices by SERVO, using the `NET_CMD_SET_SIGNING_KEY` command. Once uploaded, these keys are stored in the non-volatile memory of these devices in the clear. This memory can be read from a PVS device using the `MEM_READ` command, normally issued by SERVO. As described in Issue 19.4.1, the management interface used by SERVO is insecure, and thus can be used by an attacker to read the signing key from the eScan's and JBC's non-volatile memory.

This confirms Issue 24 reported in the California TTBR Hart source code report.

Prerequisites: In order to exploit this vulnerability, an attacker will need to have physical access to the eScan or JBC and have understanding of the Hart protocol.

Impact: An attacker that can read the memory can extract the signing keys used to create HMACs over data on the MBBs. This can be used to create forged MBBs which will be tallied by Tally. This attack may be performed in a matter of seconds at a polling place by anyone who can access the Ethernet interface on the eScan or the parallel interface on the JBC, including a voter.

Procedural Mitigations: Blocking or sealing the Ethernet port on the eScan and the parallel port on the JBC during an election may help to prevent an attacker from reading the device memory at the polling place. Physically securing the chassis of these devices may prevent an attacker from opening them in order to read the memory directly from the flash device.

Verification: We connected a laptop running a `memread` tool we created to extract the eScan's memory, and extracted the key from the memory image we read.

19.1.8 Vote order can be determined from an MBB

An attacker with access to an MBB can determine the order in which votes were cast. Hence, it would be simple for someone to uncover how a particular user cast his vote.

Description: The audit log and CVR (cast vote record) are two databases written to the local flash memory and MBB in the eSlate and eScan devices. The CVR contains one record for each record cast. The audit log contains entries for each important event, including the submission of a new ballot.

The CVR records are written from a large bank "slots". The first record is written to approximately the middle of the bank, and subsequent records are written to the next available slot above or below the current records based on a random flip of a coin. The audit log contains information related to the CVR. The CVR records a CRC over the unique voter ID and precinct ID. The audit log is written in sequential order and contains a time stamp.

Because the audit log contains uniquely identifying information, and because of the limited randomness of the write-from-the-middle scheme in the CVR, the order in which the votes were cast is easy to recover. This was confirmed by the reviewers in the California TTBR of the Hart system.

One additional fact pertaining to issue was uncovered during confirmation. The virtual coin in the CVR

scheme uses a deterministic random algorithm seeded by the local clock at the time the first vote is case. Because the adversary knows approximately when the first vote is cast, the sequence of coin flips can be identified.

This confirms Issue 25 reported in the California TTBR Hart source code report. The use of a poorly seeded randomness function was not previously reported.

Prerequisites: An attacker needs access to the MBB and knowledge of the pseudorandom function. This could be carried out by a poll worker or election official.

Impact: This attack allows compromise of a voter's privacy, since if the order in which they voted is known, the choices made on their ballot can be retrieved.

Procedural Mitigations: MBBs must be physically secured and access to them restricted through tight controls.

Verification: This vulnerability was confirmed through source code analysis.

19.1.9 Voting and audit data not secured while voting

The data held on the MBB (which is later used to tally election outcomes) is not secured while the polls are open. An adversary may be able to remove and alter the votes and audit logs in any way they wish.

Description: The CVR and audit log stored on the removable MBB has an associated HMAC computed over them to ensure integrity. This is referred to as a "digital signature" over the data. [7]

The data is "signed" only when the election is suspended (in the case of early voting) or when the election closes. Hence, anyone with access to an eScan or JBC could remove the card, and alter its contents. Moreover, such a modification could easily be made undetectable.

This confirms Issue 26 reported in the California TTBR Hart source code report.

Prerequisites: An attacker needs to know the format of the MBB, which could be reverse engineered with a minor effort, and have access to the MBB. Anyone who can gain access to an MBB, such as a poll worker or even a voter who is able to retrieve the card from an eScan or JBC if they are not closely supervised, can carry out this attack.

Impact: This vulnerability would allow an attacker to modify the votes occurring on an eScan or JBC.

Procedural Mitigations: Access to the MBBs must be physically controlled and access closely regulated.

Verification: This vulnerability was confirmed through source code analysis.

19.1.10 The internal vote count on the eScan, eSlate, and JBC can be modified

The Hart voting and JBC devices all maintain a counter in local memory that contains the number of votes cast on the device. The countermeasures used to protect this counter are ineffective. Thus, an attacker who is able to compromise the device can forge or reset the vote count.

Description: On the eScan, the store manager software stores the vote count value in two locations in flash memory under the identifier `STORE_VOTE_COUNTERS`. A CRC is calculated over the value and stored in

[7] This terminology is incorrect.

each of the two locations. When read, the CRC is checked and both values are compared against each other to validate that count was not corrupted. The count is only updated when the audit log is reset. The eSlate and JBC both implement essentially the same mechanisms through another vote count interface–in all other respects, such as the use of CRC and flash memory are identical.

There are two distinct counter measures intended to detect manipulated counters: redundant counters and CRCs. Neither of these solutions are in any effective against an attacker who can run executable code on these devices, e.g., by exploiting a buffer overflow. The attacker would simply replace both values in memory with those of the desired vote count, calculate the CRC of the value and place it in the CRC memory locations co-located with the vote count itself. Thereafter, there does not appear to be any way to recover the correct value.

A second attack would use the management interface to rewrite the memory regions. The `MEM_WRITE` or `MEM_WRITEFILE` commands instruct the target device to overwrite a region of memory with contents identified in the command. One could use these interfaces to simply overwrite the vote count and CRC data blocks. Moreover, the management command `NET_CMD_CLR_PRIVATE_CNT` instructs the target device to reset its internal "private" counter. Note that because of the lack of security of the management interfaces (for example, see issue 19.1.7), these commands may be easily forged by an attacker.

This confirms Issue 29 reported in the California TTBR Hart source code report.

Prerequisites: Knowledge of the memory layout of the victim device and compromise of the victim device code or access to the management interfaces is necessary. An outsider with a handheld device and access to the JBC or eScan (e.g. a voter on election day) can reset the counters within seconds.

Impact: The attacker is able to manipulate the vote counter.

Procedural Mitigations: None.

Verification: This vulnerability was verified by inspecting the source code.

19.1.11 The EMS software uses a vulnerable version of OpenSSL

Rally, Tally, and Ballot Now use OpenSSL version 0.97d, circa March 2004. Any service that uses OpenSSL is vulnerable to attacks against documented bugs in this version of the software.

Description: Rally and Tally can be set up to communicate remotely over a modem or network. Rally may be used to aggregate vote totals and communicate the vote counts to Tally, which performs the final vote tally and subsequently reports the election outcome. In addition, the Ballot Now program connects to a security database where passwords are kept over an SSL connection, which provides confidentiality by encrypting the transmissions, authentication of the other host, and integrity, ensuring that the transmission received contains the same information as what was transmitted.

All of these routines use a version of OpenSSL that is known to have vulnerabilities. The maintainers of the OpenSSL website are aware of these and have publicized them on their website.[8] We verified that an old version of OpenSSL, version 0.97d, circa March 2004, is in use in the Hart system, that and Rally, Tally, and Ballot Now use routines from this library.

This confirms Issue 31 reported in the California TTBR Hart source code report. The reliance of Ballot Now on OpenSSL and the vulnerable version it uses was not previously reported.

[8] ⟨URL: `http://www.openssl.org/news/vulnerabilities.html`⟩.

Prerequisites: An adversary who has knowledge of the attacks possible against OpenSSL 0.97d, including the ability to craft a buffer overflow attack or causing the machine connecting to a server to crash, could make use of these vulnerabilities. Since this is publicly available knowledge, anyone with access to the affected Hart applications (Rally, Tally, Ballot Now) may implement this attack.

Impact: An attack can cause servers to crash, hampering collection of election results or possibly preventing users from logging into applications necessary for running the election.

Procedural Mitigations: Rally and Tally are not allowed to connect remotely in Ohio, but are allowed to operate over a "local" network. Access to this network must be very closely guarded and monitored. Optimally, the machines running these services should have their network and modem interfaces removed.

Verification: The vulnerabilities in OpenSSL v0.97d are publicly known and listed on the website of the software maintainers. The vulnerability was verified through multiple methods of determining the library version.

19.1.12 Pervasive failure to follow safe programming practices

The Hart system implementation does not follow commonly accepted practices for safe source code development. This has caused, and will continue to lead to, frequently occurring security and reliability issues such as those reported throughout this and prior Hart analyses.

Description: There are a multitude of exploitable errors caused by poor coding practices; buffer overflows, `printf` attacks, integer overruns, unchecked or inconsistently checked and propagated error conditions, poor compartmentalization of data and functionality, inconsistent and over-modularization of libraries, improper trust, poor memory and resource management, and dangerous uses of operating system services (e.g., the Windows registry) are persistent throughout the code. Such errors are the bread and butter "hooks" that an attacker uses to subvert a system. We consider coding practices and make recommendations for future activities in 18.4, and defer further discussion to that section.

The Hart Source Code Report in the California TTBR study outlines the ways that the code failed to meet coding practices. We point the reader to that report for deeper exposition on the issues raised by these practices. In all cases, we found the failures pointed out in the Hart were present. Note that issue 36 of the TTBR report is encompassed by the larger design, development, and maintenance issues identified in Chapter 18.

This confirms Issue 36 reported in the California TTBR Hart source code report.

Prerequisites: The means of exploiting the above vulnerabilities are well known and there exist many tools to assist the attacker.

Impact: These errors lead to an exploitable and unstable implementation.

Procedural Mitigations: None

Verification: This vulnerability was confirmed through source code analysis and observationally through the study of this and numerous other issues.

19.2 Hart eCM Vulnerabilities

19.2.1 The same symmetric eCM key is used county-wide

Only one symmetric key is used by the eCM throughout the entire jurisdiction. This key is protected by a PIN that may be the same for each eCM token, and is propagated to every device. If an attacker can compromise this key, they can perform actions such as forging MBBs.

Description: All message integrity in the Hart system is provided by a message authentication code (MAC) key. This is the same throughout the entire county, though Hart indicates that multiple keys may be used. Before an election is called, a single PIN is generated by the appropriate election official, and this is used as the basis for generating the eCM tokens in the eCM Manager program. Only one key may be associated with a key. In eCM Manager, all eCM tokens used in the county are written, commonly using the same PIN, [9] and have the same symmetric key and key ID installed on them. The key on an eCM is then used by BOSS to generate MBBs, and by SERVO, which propagates the key to an eScan or JBC over a wired connection. In this fashion, all devices will be able to validate data that has an HMAC associated with it.

This confirms Issue 22 reported in the California TTBR Hart source code report.

Prerequisites: A poll worker who has access to the PIN can extract the key from the eCM, retrieve the information from an obfuscated file written out by eCM Manager (see Issue 19.2.2), or attempt to extract the key from a JBC or eScan as described in Issue 19.1.7. If any of these attacks are successful, the poll worker will have the key for the entire county.

Impact: An attacker may be able to forge MBBs and cause an arbitrary ballot to appear. This could affect the way a voter casts their vote by having them choose an unintended candidate, and may affect the final vote totals when the MBB is read into Tally.

Procedural Mitigations: It may be possible for different PINs to be associated with different keys and to partition the hardware in such a way that only if a particular PIN is entered will an intended key be used. Thus, MBBs could be written that will only work in eScans or JBCs respectively, or different PINs could be used for different precincts or regions within a county. This would require using the correct PIN each time the eCM is used for each different precinct/region.

Verification: These vulnerabilities were verified by source code inspection.

Note: This issue does not have any unredacted content. There is no corresponding section in the private appendices.

19.2.2 eCM keys are stored insecurely on the eCM manager

The eCM Manager software allows users to store the contents of an eCM cryptographic token to file. The file is "obfuscated" in an insecure fashion on the machine or device where the information is being stored. The obfuscation consists of performing the XOR operation on each character with the letter 'x', a mechanism that can be easily defeated.

[9] Hart suggests in the *Hart Voting System Management & Tasks Training Manual 6300-001 62E* that using multiple PINs will increase security at the costs of needing to specially mark and track PINs for each copy of the eCM. Anecdotal evidence suggests that a single PIN is commonly used.

Description: eCM Manager files are stored by default with the file extension '.eCM'. These files can be easily searched for on a machine. A simple program can be written to perform the XOR operation on each character with the letter 'x' that will recover the original details of what is stored on the eCM token.

This confirms Issue 23 reported in the California TTBR Hart source code report.

Prerequisites: The attacker needs to have access to the machine or device that the .eCM file is stored on. The attacker must know that an XOR encoding has occurred, although this is not difficult to determine for a skilled attacker. A simple program can be written to recover the original data in the file that corresponds to information on the eCM token.

Impact: This vulnerability allows an attacker to know the value of the secret key stored on the eCM token, which allows for further attacks such as forging incorrect ballot details on an MBB.

Procedural Mitigations: Access to the machines running the eCM Manager must be closely monitored. There are few cases where the file should be allowed to be saved and these should be closely monitored.

Verification: This issue was confirmed through via demonstration with the '.eCM' files produced by eCM manager. We reversed the obfuscation technique to extract the eCM keys.

19.3 Hart SERVO Vulnerabilities

19.3.1 Buffer overflows in SERVO

SERVO contains buffer overflows that can allow an attacker to execute arbitrary code.

Description: SERVO receives data from the eScan, eSlate and JBC via the "Net Message" structure. This structure contains a block of data, and a header containing the length of the block. SERVO copies the full data block into its own fixed size buffer without checking the length supplied in the header, allowing a block of data larger than the allocated buffer to be copied. This can result in arbitrary code being injected into SERVO and executed.

These buffer overflows appear in code for reading, writing and verifying the firmware and audit logs on eScan, JBC, and eSlate devices. Some of these buffer overflows exist in functions not reachable by any execution path, but are indicators of insecure programming practices.

This confirms Issue 13 reported in the California TTBR Hart source code report.

Prerequisites: In order to exploit this vulnerability an attacker will need to have compromised an eScan, eSlate, or JBC device. We show in Issue 20.3.1 that the firmware may be completely replaced in under two minutes on an eScan.

Impact: A buffer overflow allows an attacker to execute arbitrary code, and thus, completely control SERVO. This could lead to compromised firmware images or configuration files being uploaded to PVS devices while SERVO is used to perform administrative operations on those devices.

Procedural Mitigations: None

Verification: This vulnerability was verified by inspecting the source code.

19.4 Hart eScan Vulnerabilities

19.4.1 The eScan is managed via an accessible Ethernet port

The eScan optical scan device can be remotely administered via an Ethernet port, located at the back of the unit. An attacker able to access this port can perform management operations on the eScan such as replacing its configuration file or reading its memory.

Description: eScan listens on port 4600 for TCP connections. eScan normally uses a default IP address which is located in its configuration file on the eScan. Normally, this port is used for sending and receiving messages to SERVO. The protocol shared between eScan and SERVO includes commands to perform the following functions:

- Sending and receiving files from the eScan
- Reading the image of the eScan's non-volatile memory
- Uploading the eCM signing key
- Uploading and installing a new configuration file or eScan executable

Neither machine authenticates to the other, making it easy to impersonate SERVO to the eScan, or vice versa.

Note that the communications protocol between SERVO and the eScan unit is the same as that used between SERVO and the JBC, and between the JBC and eSlate units. Thus, knowledge of the protocol between any of these devices yields protocol details for all of them. Reverse-engineering the protocol between the eScan and JBC is facilitated by the fact that data is communicated over an Ethernet interface. Placing a laptop between the two devices to "sniff" the interface would allow that machine to garner details of the communications, and many tools are available to decode protocol details over Ethernet, such as the open source network protocol analyzer *Wireshark*.[10]

This confirms Issue 3 reported in the California TTBR Hart source code report.

Prerequisites: In order to leverage this vulnerability, an attacker will need physical access to the eScan and have an understanding of the Hart protocol. An attacker with access to the Ethernet interface and a handheld device running a tool which can issue commands over this interface can compromise an eScan in seconds.

Impact: Because the protocol allows for the sending and receiving of arbitrary files from the eScan, an attacker that can leverage this vulnerability may completely control the operation of the eScan. It is also possible to read the eScan's memory and extract the eCM signing key as described in Issue 19.1.7.

Procedural Mitigations: Blocking the Ethernet port on the eScan during an election may help to prevent an attacker from being able to command it in the polling place.

Verification: This issue was confirmed through source code analysis and via demonstration with the eScan. We created a set of tools allowing us to manipulate the eScan via its Ethernet port. Namely, the `getfile` and `upload` tools allowed us to retrieve, upload and install the eScan firmware and configuration file, and our `readmem` utility was used to read the eScan's non-volatile memory.

[10] *Wireshark*. ⟨URL: `http://www.wireshark.org`⟩.

19.5 Hart eSlate Vulnerabilities

19.5.1 The eSlate is managed via a serial port connection to the JBC

The JBC manages the eSlate over a serial connection whose interface is located at the top of the eSlate. The eSlate does not attempt to verify that the commands it receives over this interface are from the JBC. Thus, an attacker may connect a device to the eSlate over this serial connection and control it as the JBC would.

Description: The JBC sends control messages to the eSlate over the serial port on the eSlate. The eSlate runs a "Net Task" that waits for messages from the serial port and processes them. At no point does this task try to verify that the messages it receives are from the JBC. The commands this task will execute include the following:

- Writing to the internal audit log
- Clearing the internal CVR and audit logs
- Getting CVR records from the eSlate
- Getting the status of generated voter codes
- Reading and writing the eSlate's memory

An attacker that can connect to the eSlate's serial port can impersonate the JBC and perform any of the above listed commands.

This confirms Issue 2 reported in the California TTBR Hart source code report.

Prerequisites: In order to exploit this vulnerability, an attacker will need physical access to the eSlate, and have an understanding of the Hart protocol. An attacker using a handheld device and a tool that emulates the JBC, can impersonate the JBC to the eSlate.

Impact: An attacker able to communicate with the eSlate may control the eSlate to the same extent that the JBC may. This includes the ability to read, modify and delete the CVR and Audit logs, and change the eSlate mode, e.g. polls open, polls closed etc. These abilities can be used to modify or destroy the election results from an eSlate and infringe on voter privacy.

Procedural Mitigations: Placing a tamper proof seal between the eSlate and the attached serial cable could make it evident if the cable were detached.

Verification: This issue was confirmed through source code analysis.

19.5.2 Communication between the eSlate and JBC is insecure

The JBC and eSlate communicate via the serial port interface mentioned in issue 19.5.1. The uses of this channel include eSlate management, the transfer of ballots from the JBC to the eSlate and of CVRs from the eSlate to the JBC, and the validation of voter access codes. Because no authentication or encryption of data is used, an attacker may interpose himself between the JBC and eSlate and either eavesdrop on traffic between the JBC and eSlates or impersonate any of these devices.

Description: No cryptographic measures are taken to ensure either authentication, message integrity or message confidentiality. This means that an attacker can impersonate either the JBC or the eSlate, or eavesdrop on the communication over this channel. Impersonating the JBC allows an attacker to do the following:

- Acknowledge the reception of CVRs, thus preventing the JBC from recording them
- Control the eSlate, as described in Issue 19.5.1
- Claim that invalid voter codes are valid, thus preventing a CVR from being recorded

Impersonating an eSlate allows an attacker to do the following.

- Send forged CVRs to the JBC
- Mislead the JBC into believing that the management operations described in Issue 19.5.1 were carried out, when they were actually ignored

This confirms Issue 5 reported in the California TTBR Hart source code report.

Prerequisites: In order to exploit this vulnerability, an attacker will need to have physical access to the serial cable connected to the top of the eSlate or at the back of the JBC, and have an understanding of the Hart protocol.

Impact: An attacker able to eavesdrop on the JBC to eSlate connection or impersonate an eSlate or JBC may, read, modify or destroy election records, prevent votes from being counted, and infringe on voter privacy by monitoring votes as they are cast.

Procedural Mitigations: A tamper evident seal could be placed between the serial port connector and the eSlate, as well as the serial port connector and the JBC. Note that this would only make it evident that an attack had occurred, and would not reveal any information about the nature of the attack or to what extent it impacted the election results.

Verification: This issue was confirmed through source code analysis.

19.5.3 Compromised eSlates can provide votes without voter records

The JBC does not verify that the eSlate provides only one vote per voter code. Essentially, the eSlate is charged with performing checks to see that the voter code is correctly processed. Hence, a compromised eSlate can ignore such checks and simply "stuff" the ballot box with votes.

Description: The JBC polls the eSlates for status at a regular interval. Included with the response is a status flag indicating, among other things, that a) the local voter code needs to be validated, and b) whether there is a CVR record ready to be processed. A compromised eSlate can simply signal (b) every poll and thereafter insert votes into the record.

Moreover, the compromised eSlate can listen in on the communications between the JBC and other eSlates to extract voter codes, or predict voter codes before they are issued. If properly timed, the additional inserted votes will be difficult to identify after the fact.

This confirms Issue 8 reported in the California TTBR Hart source code report.

Prerequisites: An attacker needs to have compromised one of the eSlates used in the election.

Impact: A compromised eSlate will allow the attacker to hijack voter codes and use them to vote in an arbitrary manner.

Procedural Mitigations: Prevent the eSlate program from being compromised.

Verification: This vulnerability was confirmed through source code analysis.

19.5.4 The periodic memory integrity checks used by eSlate and JBC are ineffective

Both eSlate and JBC devices check the integrity of blocks of internal memory used every 10 seconds. However, these checks are trivially bypassable. Thus, these checks are very unlikely to detect a system compromise.

Description: A mechanism often used to check data integrity is a *cyclic redundancy check* (CRC). This is commonly used to detect errors in communication or within blocks of data on persistent storage or memory devices. A CRC function works by calculating a mathematical function over the digital values stored in each byte of the checked region. The resulting value (2 bytes in the case of the Hart implementation) is called the *CRC value*. Later, the integrity of the target device is checked by recomputing the CRC value over the same data. If one or more bits of data have changed, it is highly likely that the CRC value will change.

Both the eSlate and JBC execute a background thread that performs a number of service functions. One of these functions detects when memory regions have been changed. Each of the devices maintains a table of "important"[11] data regions and associated CRC values. Every 10 seconds, the processor recalculates the CRC of the tested regions and compares it against the values stored in the table. If the CRC values do not match, a error is generated and the system fails.

One problem with CRCs is that they are not secure. An adversary who has control over the value in the region can easily craft a new (and different) set of data values that calculate a new set of data values that compute to the same CRC.

This approach is unlikely to be effective in detecting or preventing malicious activity. Here are but a few of the reasons for this conclusion:

- An attacker can penetrate the system and replace the existing data/code with its own such that it calculates to the same CRC.
- An attacker can replace the code that executes the check with some others that perform some other function, or none at all. Simply replacing the call with `noop` (null operation) codes would be a quick and easy way to patch the existing code.
- An attacker can replace the data held in the table to some known region. For example, there is nothing in the source code to prevent the table from simply checking itself over and over.
- An attacker could patch the firmware, but not the table in memory. This would cause a crash, but the device would run the new firmware when it boots.
- An adversary could use this list in conjunction with buffer overflows to change data values in one of the checked areas. If carefully placed, the failure would only be detected at the next CRC check sequence. Because of the potentially long period between the malicious input and check, the adversary could mask his behavior.

[11] It is unclear what policy is used to determine what regions of memory are deemed important.

Note that 10 seconds is very long time in computer terms. Literally billions of instructions are executed even on modest hardware in a single 10 second period. Thus, the adversary could launch a very large number of activities in this period.

This confirms Issue 9 reported in the California TTBR Hart source code report.

Prerequisites: In order to exploit this issue, an attacker will need knowledge of the memory layout of the victim device, and access to the device to upload firmware.

Impact: If an attacker has compromised an eSlate or JBC device, it will likely be undetected.

Procedural Mitigations: None

Verification: This vulnerability was verified by inspecting the source code.

19.6 Hart Servo Vulnerabilities

19.6.1 SERVO-based device firmware checking can be spoofed

SERVO can verify the integrity of the firmware on a JBC, eSlate or eScan by comparing the SHA-1 hash of that device's firmware image to a Hart supplied hash. Non-matching hashes result in an error. SERVO depends on the device in question to provide its firmware image, and thus, a compromised device may simply store an uncompromised copy of the firmware image, and provide it to SERVO as prompted.

Description: For each device to be inspected, SERVO first obtains that device's ID and checks to see if it is already in a database of devices for a given county. If it is not, SERVO will obtain the device's firmware version number, consisting of major, minor and release fields, and compare it against the version number stored in its database. Next, SERVO requests the device's firmware image, which it computes a SHA-1 hash over, and compares against the hash in its database.

This method for device firmware verification assumes that the device will provide the correct firmware image. This is not necessarily the case. A compromised device could store an uncompromised copy of the firmware, and provide it to SERVO when prompted.

This confirms Issue 11 reported in the California TTBR Hart source code report.

Prerequisites: In order to exploit this vulnerability, an attacker will need to have compromised a JBC, eSlate or eScan.

Impact: If an attacker has compromised a JBC, eSlate or eScan, this vulnerability allows the compromise to go unnoticed by SERVO. This creates the possibility that a compromised JBC, eSlate or eScan could be re-used in future elections without the compromised firmware being noticed.

Procedural Mitigations: None.

Verification: This issue was confirmed through source code analysis.

19.7 Hart JBC Vulnerabilities

19.7.1 The JBC internal version checks can never fail.

The JBC checks its own version number on startup. This check can never fail, and thus provides no meaningful security function.

Description: A statically defined version number is compiled into the code, but it is never checked. A function apparently exists to compare this static version against some legitimate numbers, but this is not implemented. Thus, the functionality of the call is suspect. While this is not a vulnerability on its own, it is indicative of incomplete functionality that is pervasive throughout the system, as discussed further in chapter 18.

This confirms Issue 10 reported in the California TTBR Hart source code report.

Prerequisites: None

Impact: None

Procedural Mitigations: None.

Verification: This vulnerability was confirmed through source code analysis.

19.7.2 JBC-based eSlate firmware checking can be spoofed

The JBC attempts to verify the integrity of attached eSlate devices by performing a CRC check against the eSlate and checking its version number. This approach requires the eSlate to be answering truthfully, which may not be the case if the device has been maliciously altered.

Description: When the JBC is started, it finds eSlates attached to it and retrieves information associated with them. It in turn assigns a node ID value to address the machines. A value identified as a CRC value in the code is retrieved, as is the version number of the eSlate. As a test of the eSlate's integrity, the major version number of the device is compared to a hard-coded value that the JBC possesses; the other parts of the version number (minor number) and CRC are not checked at this time. If the verification of the major number succeeds, the eSlate passes the test. This value is given by the eSlate, meaning that if a malicious device has provided incorrect numbers to the JBC to disguise itself as a legitimate eSlate, the JBC will not detect this. In addition, the check only occurs at startup, so an eSlate that is compromised during an election will not be rechecked.

This confirms Issue 12 reported in the California TTBR Hart source code report.

Prerequisites: An attacker will need to have compromised an eSlate device to leverage this vulnerability. This could potentially be performed by a poll worker.

Impact: If an attacker has compromised an eSlate device, this vulnerability allows the device to mask itself by passing the integrity check, and then act without concern of being detected.

Procedural Mitigations: Strong physical security and controlled access to the eSlate is necessary to attempt to prevent attacks that may compromise the machine.

Verification: This vulnerability was verified by inspecting the source code.

19.7.3 The JBC is managed via an accessible parallel port

The JBC is managed by the SERVO application over a parallel port in the back of the unit labeled "printer". An attacker that can access this port can control the JBC and any connected eSlates.

Description: SERVO connects to the JBC over a parallel port to perform administrative tasks. These tasks include the following.

- Uploading the signing key
- Controlling the JBC's LCD display
- Clearing the CVR logs and audit logs

No authentication of the device acting as SERVO is performed by the JBC. Thus, an attacker with any device with a parallel port interface can issue any and all commands to the JBC and eSlates that SERVO can. We describe our utility and a separate attack that may be used to fully delete the results of an election in Section 20.4.2.

This confirms Issue 1 reported in the California TTBR Hart source code report, which identified the vulnerability in source code, but did not perform any verification.

Prerequisites: In order to exploit this vulnerability, an attacker will need to have physical access to the JBC and an understanding of the Hart communication protocol.

Impact: An attacker controlling the JBC over this interface can perform the same administrative functions as SERVO, including those listed above.

Procedural Mitigations: Securely blocking access to the JBC's parallel port, such as with a lock or a seal, at all times except when it is being administered by SERVO, will be a step towards preventing unauthorized devices from being connected to the JBC, though the utility of this mitigation may be limited in the face of a determined attacker with the correct tools to circumvent the physical countermeasures.

Verification: This issue was confirmed by source code analysis and via demonstration with the JBC. We wrote a `jbccmd` program which we ran on a laptop connected to the JBC's parallel port. Using this program we successfully issued commands to the JBC and eSlate with no authentication.

19.7.4 The JBC Voter Registration Interface can be used to generate voter access codes

A modem interface, accessible with a DB-9 cable, is located on the back of the JBC. This interface is serial and is known as a VRI, or Voter Registration Interface. A VRI device attached to the JBC can be used to send instructions to generate access codes for voters. An attacker can use this port to generate their own codes to be used for voting on the eSlate machines.

Description: In early voting mode, the VRI may be used to generate access codes. These may be printed from the JBC's printer or printing may be suppressed. In either event, as long as a properly formatted request is issued over the VRI to the JBC, it will return a valid access code. Furthermore, when the JBC is in general election mode and the polls are marked as open, the VRI assumes the existence of a barcode scanner that can be interfaced with the JBC. Commands can be issued through this interface that allow access codes to be printed out on the JBC's printer; an arbitrary number of these codes may be generated, and they are valid for a length of time determined when the election is defined in BOSS. While the copy of BOSS sent to us

defaults to 30 minutes as a timeout period for the access codes, this value can be set as high as 960 minutes (16 hours). [12]

In a further result to what was noted about this vulnerability in the California TTBR Source Code report, we found that despite the documented limit of 150 outstanding voter access code requests, we were able to generate many more valid codes; we were, in fact, able to generate over 10,000 access codes within a single expiration period, meaning that duplication in codes must have occurred. At this point, any code entered into the eSlate, as long as it had not already been used, would be valid for voting with. One note is that the number of active codes is displayed on the JBC, and as the number passes 500, the value displayed turns to '____'.

This confirms Issue 4 reported in the California TTBR Hart source code report.

Prerequisites: The attacker must be able to access the serial port on the JBC and possess a device, such as a PDA, to run a terminal emulator program. They must also know the precinct ID.

Impact: A voter would be able to vote an arbitrary number of times before the access codes expire.

Procedural Mitigations:

We were not supplied with any devices to leverage the VRI, which leads us to suspect that devices such as voter registration equipment or barcode scanners are not meant to be plugged into the JBC for elections in Ohio. Accordingly, the serial interface on the JBC should be sealed off or removed altogether, as it has no purpose. There are two models of JBC that are manufactured by Hart, the JBC 1000 B, which has a modem, and the JBC 1000, which does not.[13] We received a JBC 1000 B for testing purposes with a serial port blocker from the Hamilton County Board of Elections. This indicates that the JBC 1000 B is used in the field in Ohio. The port blocker is attached to the modem port on the JBC with two screws that may be easily removed by an attacker, and the security it provides is minimal, if not cosmetic. Only JBC 1000 models, which do not contain a modem, should be used in Ohio.

Verification: This vulnerability was confirmed through source code analysis and demonstration with the JBC. We wrote a program to retrieve voter codes over the VRI in early voting mode, and to issue code requests on election day through the same interface as our program pretended to act as a barcode scanner. We verified that over 10,000 codes could be active at a given time by printing an access code report from the JBC.

19.7.5 Format String vulnerabilities in the JBC report mode

The implementation of several reports available to poll workers at the close of the polls has errors that can be exploited to extract arbitrary data from the JBC. This attack may be used to extract key data from the JBC or to execute other code.

Description: Poll workers have the option of generating three kinds of reports after closing the poll: a voter code summary report, a vote tally report, and a write-in report. The implementation does not perform *input filtering* on the write-in data, meaning it is possible to enter code disguised as regular input text. Specifically, code on the JBC allows user-supplied data to be interpreted as `printf` format strings. These allow the user to extract data from the stack or heap and print them on the reports, and possibly to execute other code.

[12] Hart Intercivic, *BOSS Operations Manual 6100-019 Rev. 43-62A*, page 489.

[13] Hart Intercivic, *Hart Voting System, System for Election Records and Verification of Operations: Operations Manual, Software V. 4.2, 6100-102 Rev. 42-62B (SERVO Operations Manual)*..

This confirms Issue 6 reported in the California TTBR Hart source code report.

Prerequisites: Access to a vote operation on the JBC and a method of inputting special characters necessary to mimic `printf` command statements. While these are not generally accessible through the eSlate's on-screen keyboard, it would be possible to send these commands if the eSlate was compromised (Issue 19.5.1) or if the broadcast network between a JBC and eSlate was tapped such that commands could be sent over it (Issue 19.5.2).

Impact: All secret data may be extracted from the JBC, and arbitrary code could potentially be run on it. This could be an attack vector for loading malicious code onto the JBC, for example, as a means of infecting SERVO. [14]

Procedural Mitigations: Restrict access to the JBC and all eSlates to minimize the chance of a compromised host.

Verification: This vulnerability was confirmed through source code analysis.

19.7.6 Voter codes are not selected randomly

All past and future voter codes for a precinct can be determined from a single voter code. Anyone with access to a single voter code (a legitimate voter, for example) can easily determine the voter code of everyone who has voted or will vote at that precinct.

Description: Four-digit numerical voter codes are used to authenticate voters using an eSlate at a polling place. After receiving the number from a poll worker, the voter enters it at an eSlate to begin voting. The eSlate checks with the JBC to ensure that the voter code was issued and that it has not been used before.

Voter codes are selected by selecting a random number from 0-9999 and performing a permutation on the "index" value. Subsequent voter codes are determined by adding one (wrapping from 0-9999) to the index and permuting as before. Because the permutation is invertible and constant, anyone with access to one voter code can compute the sequence.

This confirms Issue 7 reported in the California TTBR Hart source code report.

Prerequisites: Knowledge of the algorithm used, which has been reverse-engineered without source code.[15] It should be assumed that this algorithm is known, as it can be easily recovered from anyone who has access to a working JBC or can see a sequence of values, e.g., any poll worker handing out voter codes.

Impact: This vulnerability allows an attacker to cast multiple votes, potentially without detection, and to prevent others from using that same vote.

Procedural Mitigations: Only one eSlate should be used in a polling place, and only one voter code should be allocated at any one time. This is the only effective way to prevent multiple voting and stealing access codes from other voters.

Verification: This vulnerability was confirmed through source code analysis. In addition, we developed a utility that will print the next series of access codes that will be generated given a code, as well as previous codes in the sequence. We tested this by receiving an access code from the JBC and running the utility with the given code as input. We verified that the sequence of access codes generated were the same as those

[14] See Section 21.1 for more information on this attack.

[15] Elliot Proebstel et al., 'An Analysis of the Hart Intercivic DAU eSlate'. In Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop. August 2007.

issued by the JBC.

19.8 Hart VBO Vulnerabilities

19.8.1 The VBO record is easily manipulatable by an eSlate program

The VBO record is printed using a rudimentary printer. This printer is completely under control of the program to which it is attached. This allows a compromised eSlate to prevent any use of the VVPAT, to invalidate legitimate votes, and to insert illegitimate votes.

Description: The printer used by the eSlate is under control of the eSlate software. The printer software simply follows commands provided by the eSlate and executes them. Two functions, `Print` and `FormFeed` allow the eSlate to arbitrarily modify the VVPAT record, often without the knowledge of the voters.

An attacker can use this function to invalidate legitimate ballots by printing a `BALLOT REJECTED`, then printing `BALLOT ACCEPTED` with votes of the attackers choice.

Finally, the printer can be forced to forward the paper to the end of the reel, thus disabling printing.

This confirms Issue 33 reported in the California TTBR Hart source code report.

Prerequisites: An attacker must compromise the eSlate and upload new firmware either to the eSlate or VBO.

Procedural Mitigations: Because limiting physical access to the eSlate and VBO is not possible, the usage of tamper evident seals would at least reveal if the VBO or eSlate's serial port interface has been tampered with.

Verification: This vulnerability was confirmed through source code analysis.

19.8.2 VBO printer allows the paper to be rewound

The VBO record is printed using a rudimentary printer. The interface allows the controlling software to instruct the printer to rewind the paper. It may be possible for an adversary to use this feature in the VBO software to overwrite legitimate ballots.

Description: The printer has an I/O pin that, when raised, appears to instruct the printer to rewind the paper. If the printer software is compromised, it could trigger the reverse motor after the ballot is completed. This may cast doubt on the ballot and possibly make its content unreadable.

Note that this code is compiled via MPLAB embedded processor tools. One would need to exploit for example, a buffer overflow, to take control of the processor and execute commands to perform this attack. There do not appear to be any commands that allow the user to exploit this feature without taking over the firmware.

This confirms Issue 34 reported in the California TTBR Hart source code report.

Prerequisites: Access to the printer driver software.

Impact: Allows the attacker to invalidate or corrupt the VVPAT record generated by an eSlate system.

Procedural Mitigations: Prevent application printer driver program from being compromised.

Verification: This vulnerability was confirmed through source code analysis.

19.8.3 VBO ballots are printed sequentially

The VBO record is printed in order on a single roll of printer paper. An attacker with access to the voter record book can therefore correlate the vote on the roll to the individual voter.

Description: The eSlate only allows printing of a ballot to occur on the roll of paper. Thus, the order of voting is recorded implicitly from the order in which it is recorded. Anyone who has access to the voter signin or can observe the order in which known voters voted can determine the way in which they voted.

This confirms Issue 35 reported in the California TTBR Hart source code report.

Prerequisites: Access to the printer voter rolls or observation and VVPAT.

Impact: An attacker able to correlate a vote on the VVPAT to an individual voter can compromise that voter's privacy.

Procedural Mitigations: Restrict access to VVPATs and voter records.

Verification: This vulnerability was confirmed through source code analysis and by observation.

19.9 Hart Tally Vulnerabilities

19.9.1 The Tally interface allows a Tally administrator to "adjust vote totals"

Tally allows an election administrator to manually adjust the vote totals for a jurisdiction, to allow for voting sites where no Hart equipment is being used. While an audit log record is created, there are no other records made that indicate an adjustment has been made, and there is no history of adjustments displayed in case an error in entering the totals is made.

Description: Adjusting the vote total will add to the total number of votes captured in the database; there is no distinguishing these votes from those that were counted through the Hart equipment. Rather than showing the adjustments to the vote total, the total is overwritten outright in the database. This can cause errors during a recount or, if a recount is not called, the changes may go unnoticed, allowing for the election results to be tampered with.

This confirms Issue 17 reported in the California TTBR Hart source code report.

Prerequisites: Accessing the "adjust vote" feature is only possible by an administrator on the system. If an administrator is malicious, or if someone has access to the system and can guess the administrator username and password, they can alter the vote counts. Note that any user with physical access to the EMS machines can login to the EMS applications with administrator privileges as described in Issue 20.1.3.

Impact: The number of votes can be altered for an election. If a number has been incorrectly entered, it is unclear how many votes should be reversed as no history of adjustments has been shown; this can be a source of confusion to election officials who may end up making incorrect adjustments. In addition, while the adjustments are logged in the audit log, other vulnerabilities have shown that this log can be modified

(particularly Issue 19.1.3) by modifying the databases; thus, it is possible for a malicious user to both adjust totals and cover their tracks.

Procedural Mitigations: A potential procedure to mitigate this threat is for a to observe an administrator adjusting the vote totals, and logging this value on paper. In addition, access to the machine running Tally must be secured against any user without a corresponding witness observing all activity.

Verification: This vulnerability was confirmed through source code analysis. We also ran Tally and verified that when the manual adjustment is made, there is no on-screen indication of the number of adjusted votes.

19.9.2 Precinct IDs are improperly handled by the system

MBBs can be manipulated to force the Tally software to accept votes for precincts that that MBB is not authorized for. Manipulation of the Tally database can prevent votes from being being counted.

Description: Every MBB is authorized to report votes for a precinct identified in its internal data. This information is recorded in an election database created by BOSS and subsequently used by Tally for reconciling the MBBs. Cast vote record (CVR) data tagged with precinct identifiers not associated with the card in the database are ignored. MBBs encoded as being SERVO recount MBBs (designated by an equipment identifier in the MBB header) are an exception to this. These cards are implicitly authorized to report data for all precincts.

Because the equipment identifier is read from the header, an attacker can create a card listing votes from any number of legitimate precincts simply by changing the identifier to SERVO. Tally will see the SERVO identifier and unquestioningly add the votes to the election results.

Further, an adversary who can manipulate the Tally database can change the precincts associated with a legitimate MBB. When the legitimate MBB is processed by Tally, the votes counted are ignored.

This confirms Issue 27 reported in the California TTBR Hart source code report.

Prerequisites: An attacker must know the format of the MBB, which could be reverse engineered with a minor effort, and have knowledge of the MBB key.

Impact: This vulnerability may allow an adversary to add votes to an arbitrary precinct, or to prevent legitimate votes from being counted.

Procedural Mitigations: MBBs must be well-protected.

Verification: This vulnerability was confirmed through source code analysis.

19.9.3 Users can tally unclosed or corrupted MBBs

Failure of the integrity check software can be ignored by the user when processing MBBs. In short, the user can ignore error messages and introduce potentially corrupt, modified, or forged voting data into the Tally software.

Description: The Tally software checks the cryptographic integrity when an MBB is inserted into the local drive by calculating an HMAC. If this check fails, the user is presented with a message “The MBB data fails cryptographic validation. Accept the MBB?” and buttons for “yes” and “no”. If the user clicks the “yes” button, then the MBB is processed as if the integrity check did not fail.

This feature could be used by an attacker to insert fraudulent votes into the database. The attacker who gains access to the MBB during or after an election, but before it is tallied, can create cast vote record (CVR) entries that appear to be legitimate. A malicious election official, or simply one who ignores the warning and accepts the MBB, will introduce invalid votes.

Tally creates two events when the HMAC failure ("MBB digital signature failure") is detected and MBB is subsequently accepted ("MBB User Accepted"). This is the only indication that the corrupt MBB was used.

This confirms Issue 28 reported in the California TTBR Hart source code report.

Prerequisites: Access to an MBB and control over the Tally process. The attacker would also need a program capable of reading the MBB and writing the CVR record onto it.

Impact: This attack allows the attacker to insert fraudulent votes into the voter tallies.

Procedural Mitigations: Access to the MBBs must be closely controlled. The tallying process must be carefully monitored, and audit logs must be carefully reviewed after the election is over.

Verification: This vulnerability was confirmed through source code analysis and via demonstration. The review team created a program to duplicate previously cast votes in the MBB CVR log. The HMAC was not changed, and an error was correctly detected. The fraudulent votes were accepted and reflected in the vote tallies.

19.9.4 MBBs are only processed if the Tally database allows it

Tally records in its internal database the unique identifier of all MBBs that it has processed. Tally will not record votes from an MBB that is already in this list of "tallied" cards. An attacker who can either modify the internal Tally database or provide forged MBBs to Tally can prevent legitimate MBBs (and thus votes) from being tallied.

Description: Tally keeps a table of processed MBBs in its internal database, which is keyed by the unique ID of each MBB. A MBB's votes are tallied only if a record is not in the processed MBBs table. Specifically, when an MBB is inserted and tallying is attempted, Tally checks to see if a record in the above table associated with the unique ID exists. If no record exists, the MBB votes are added to the election tallies. If a record does exist, the MBB will be rejected.

Legitimate MBBs can be prevented from being tallied in at least two ways. First, an attacker with access to the database can simply insert records containing the IDs of MBBs that they do not wish to be counted. Second, an attacker with access to the county-wide key (see Issue 19.2.1) can create MBBs with the same IDs as the legitimate ones it would like to prevent from being processed.

This confirms Issue 30 reported in the California TTBR Hart source code report. The misuse of the Tally database was not previously reported.

Prerequisites: An attacker needs the MBB key (for forging MBBs), which can be obtained from the eCM tokens or via other means (see Issues 19.2.2, 19.1.7), or access to the Tally database.

Impact: This vulnerability allows an attacker to prevent legitimate votes from being counted.

Procedural Mitigations: The integrity of the Tally database must be ensured. Access should be strictly controlled to the county-wide key, and strict physical control and inventorying of the MBBs is required such that only MBBs from legitimate voting machines are provided to Tally.

Verification: This vulnerability was confirmed through source code analysis.

19.9.5 Rally and Tally allow a user to accept previously unrecognized certificates

Rally and Tally protect their connection with OpenSSL, which provides encryption and authentication over a network connection. This connection is set up through the use of certificates, which are digitally signed by a certificate authority trusted by both Rally and Tally. In the implementation of Rally and Tally, there is no reliance on a certificate authority; rather, each application will obtain certificates from the first connection made to them. Thereafter, if the connecting software presents a certificate other than the originally supplied, the user is prompted as to whether the new certificate should be should be accepted. This certificate is used to authenticate Rally to Tally and vice-versa. Thus, anyone who connects to Tally or Rally can pretend to be the other application, and that application will simply prompt an unsuspecting user to accept the replaced certificate.

Description: The Rally and Tally software perform no real certificate management on behalf of the user, but assume the user is competent to judge the received certificate based only on some asserted source. An improved setup would have the key of the certificate authority installed in both applications, such that they could verify that a received certificate was digitally signed by the trusted certificate authority. As part of the initial mutual authentication process, the user accepts the certificate by hitting an "OK" button in a presented interface. Thereafter, if a new certificate is presented, the same accept dialog is repeated.

The interface presented to the user includes information about the organization, city, state, and country of the certificate owner. Note that an adversary can simply reuse the same information in an original legitimate certificate, as one can create a certificate asserting any identity desired. Thus, the user has no meaningful information to use when deciding if a received certificate is authentic. This introduces opportunities for *man-in-the-middle* attacks, where the attacker is able to intercept traffic between two parties and alter the contents so that the messages the attacker can choose what messages are received by each party, or outright *masquerading* attacks, where the attacker pretends to be the other party.

This confirms Issue 32 reported in the California TTBR Hart source code report.

Prerequisites: Access to the Ethernet interface on Rally or Tally.

Impact: Allows the adversary to impersonate a Rally or Tally host, or to act as a man-in-the-middle (and observe and modify all communication between the two).

Procedural Mitigations: Develop careful controls over how and when certificates should be accepted.

Verification: This vulnerability was confirmed through source code analysis.

19.10 Hart Ballot Now Vulnerabilities

19.10.1 Ballot Now ballot counters are stored in a database

Ballot Now keeps a private counter of the number of ballots that have been processed, and a public counter of the number of ballots processed for a given election. An attacker, such as a malicious election official who has access to the database, can modify these values directly, thereby rendering the ballot allocation and process checks unusable.

Description: The state of the system is largely captured in Ballot Now in an internal database. Among the values stored are counters that record the number of ballots that have been processed by the system for a particular election. These counters may be changed by anyone able to connect to the Ballot Now database. Passwords to access the database are easily obtainable, as they are insecurely stored (as described by Issue 19.1.2).

This confirms Issue 16 reported in the California TTBR Hart source code report.

Prerequisites: Access to the database.

Impact: This attack allows the attacker to falsely cast doubt on the correctness of an election. In addition, it may be used to cover another attack, such as ballot forgery.

Procedural Mitigations: One needs to ensure the attackers cannot access the database.

Verification: This vulnerability was confirmed through source code analysis.

CHAPTER 20

HART NEW ISSUES

This chapter describes vulnerabilities we found in the Hart InterCivic election management system, beyond those discovered in the California TTB Source Code Report for Hart. Note that as discussed in Section 2.4, we were not given access to a central count optical scanner, used for processing absentee ballots with Ballot Now. This limited our ability to test the operation of Ballot Now as an input processing device during elections. In addition, due to time restrictions, we only performed a cursory overview of the Disabled Access Unit (DAU)-specific features of the eSlate. However, many of these issues have been covered in a recent academic report focused on study of the DAU.[1]

The public version of this report contains references to sections of the private report. These redacted sections contain confidential information identifying where in the source code or design the issue is manifest, or other vendor-confidential information. Note that all source code, file names, and other apparent vendor specific information in the public version have appeared in previous election study analysis reports.

20.1 Hart General Vulnerabilities

20.1.1 An MBB image can be copied and restored without any credentials.

The data on an MBB (called the MBB "image") can be removed from the card and restored using simple tools. This allows an attacker to effectively edit out certain parts of an election by periodically removing the card and saving it to its previous state. Thus, voters will have their votes erased from the MBB during an election. Unless the internal CVR logs of the devices in question are examined after the election, the modified votes may never be noticed.

Description: The MBB is a readable and writable electronic storage device similar to a hard or floppy disk. At a low level, it is simply a persistent device that can be read and written to freely by an operating system. An attacker can exploit this design by simply copying the data off the MBB (and onto, for example, a local hard drive) and restoring using commonly available tools such as the UNIX `cpio` utility. This feature was enormously helpful in our analysis; it allowed us to save the state of an election at any time and replay and experiment with it later.

This feature can be exploited by an attacker to manipulate an election. For example, assume an attacker knows that a set of voters who vote over lunch are predisposed to vote in a certain way that is undesirable

[1] Elliot Proebstel et al., 'An Analysis of the Hart Intercivic DAU eSlate'. In Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop. August 2007.

to him. He could then copy the MBB image onto a hard at 11:30am and reinsert it back into the election device. At and reinsert back into the device. When the device boots again, the preceding 2 hours would be effectively erased from the MBB with no visible notification. This would prevent the votes over that period from being counted; in fact, there would be no indication that they had voted at all apart from the internal logs in the devices.

Further, if that same attacker has access to the VVPAT, he can simply cut/tear out the removed entries. Thus, there will be no evidence of the votes occurring other than a count of the number of voter sign-ins.

Prerequisites: The attacker needs access to the MBBs, and possibly the VVPATs.

Impact: This attack may allow an attacker to remove targeted votes from an election.

Procedural Mitigations: Any discrepancy between the voters signed into a precinct and the number of votes counted should be flagged as a potential compromise. Note that this would not correct the problem (there is no correction), but simply detect the problem potentially exists.

Verification: We verified this issue via demonstration. We were able to remove a voter's vote from being tallied by copying and restoring an MBB.

20.1.2 EMS systems make improper use of the Windows registry

The Windows registry is used to store configuration data for applications running on a windows system. Each configuration parameter is stored in a single registry entry. EMS applications use the Windows registry to store configuration parameters that can enable debugging functionality, reveal sensitive information and disable or modify the behavior of EMS applications.

Description: The Windows registry is used for storing configuration parameters used by applications. The Hart EMS applications misuse the registry in several ways.

- Debugging features can be activated through the registry.
- Confidential information is stored in the registry.
- The registry is used to control the functionality of user interfaces.

The practice of leaving debugging or "back door" features in production level software is considered dangerous. Debugging functionality left in applications can expose less tested parts of the application which may lead to the compromise or misuse. The use of registry entries to control debugging features of Ballot Now is discussed in Issue 20.7.2.

Many of the registry entries on the EMS machines can be read by any user on the machine. For this reason, any confidential information such as passwords stored in the registry will be viewable by any user on the system. The password for the Ballot Now security database private key is stored in the registry, and thus is readable by any user on the system. An attacker able to read this key can monitor and potentially modify the traffic between Ballot Now and the Ballot Now Security database.

The registry is used to store parameters affecting the availability and behavior of user interface components in Tally. Misconfiguration or tampering of these parameters can make Tally unusable or cause subtle errors in its operation, which could be overlooked by the user. The use of registry entries to configure the Tally user interface is discussed in Issue 20.6.2.

Prerequisites: In order to leverage registry related issues, an attacker will need physical access to a Hart EMS machine.

Impact: An attacker able to modify registry entries on EMS machines can print pre-voted ballots, view confidential information and modify the Tally user interface.

Procedural Mitigations: Limiting physical access to the EMS machines at all time may reduce the opportunities for an attacker to tamper with the registry. Ensuring that the EMS systems are never connected to a network may also reduce such opportunities.

Verification: We verified these vulnerabilities by enabling the Autovote functionality in Ballot Now, [2] and removing and modifying the behavior of user interface components in Tally. [3] We did not attempt to monitor or modify traffic between Ballot Now and the Ballot Now security database by decrypting the Ballot Now private key using the password stored in the registry.

20.1.3 Hart EMS passwords can be bypassed

The EMS applications, BOSS, Ballot Now, SERVO and Tally, all require a username and password to login. An attacker can bypass having to enter these credentials by modifying the security database associated with each application.

Description: Each EMS application requires users to enter a username and password before using the application. Once the user's credentials are entered, they are checked against values stored in the security database for each application. If no usernames are present in the security database, the application will prompt the user for a new administrator username and password, and use them to create a new administrator account which the user can then login with.

As described in Issue 19.1.2, an attacker can connect to the EMS databases, including the security databases, and modify their contents. Once an attacker has deleted the usernames in the security database for an EMS application, he may open the application, create a new administrator account, and login using that account.

Prerequisites: In order to exploit this vulnerability, an attacker must have physical access to any of the EMS machines, and to have extracted the database passwords from the configuration files as described in Issue 19.1.2.

Impact: An attacker that can login to the EMS applications can completely undermine any or all of the following.

- Ballot definition
- Ballot creation
- Vote tallying
- PVS maintenance

The specific actions that can be performed by an attacker with access to the EMS applications include the following.

- Adjusting the vote totals in Tally

[2] See Issue 20.7.2 for more information on this vulnerability.
[3] See Issue 20.6.2 for more information on this vulnerability.

- Adding new user logins to EMS applications
- Clearing the audit logs stored in the JBC, eSlate, eScan and MBBs with SERVO
- Printing Ballots with Ballot Now

Procedural Mitigations: Limiting physical access to the EMS machines would reduce the opportunities an attacker would have to connect to the EMS security databases. Ensuring that the EMS systems are never connected to a network would also reduce such opportunities.

Verification: We connected to the security database for each EMS application, BOSS, Ballot Now, Tally and SERVO, as described in Issue 19.1.2, and deleted the usernames. This allowed us to create new administrator accounts upon starting each application.

20.1.4 Hart EMS audit logs can be modified or erased

The EMS applications, BOSS, Ballot Now, SERVO and Tally, all maintain audit logs of the functions they have performed. The EMS audit logs are stored in the databases for each application. An unprivileged attacker may open these databases and modify the audit logs, covering any evidence of tampering with EMS applications and election data.

Description: The EMS audit logs are used to maintain an audit trail of the actions taken by EMS applications. An attacker that has tampered with the EMS applications, perhaps by circumventing the passwords as described in Issue 20.1.3, could completely remove and trace of this tampering in the EMS audit logs.

Each audit log entry contains at least the following.

- **date and time**: The date and time at which the action was logged
- **user**: The name user that performed the logged action
- **code**: A unique identifier used to identify the action performed, the code/description pairs are located in another table in the database
- **data**: Any special information recorded about the log entry, for example, if the vote totals were adjusted, the **data** field will contain the amount of the adjustment.

Once an attacker has connected to an EMS database, suspicious audit log entries can be removed in several ways. The simplest way would be to delete all audit log entries. This method will also arouse the most suspicion, as some EMS applications place a sequential **audit ID** number with each audit log entry. A more discreet method would be to delete entries by **audit ID** and replace them with entries with less suspicious **codes** for actions such as "successful login," with the same **audit ID**.

Prerequisites: In order to exploit this issue, an attacker will need access to the Hart EMS systems, and to have extracted the database passwords from the configuration files as described in Issue 19.1.2. With this access, modifications to the audit log for a single entry could be performed within seconds, while large-scale modifications could be performed in minutes.

Impact: An attacker able to modify the audit log can make it difficult or impossible to detect that an EMS application has been tampered with. This would allow an attacker to cover up having performed any and all functions of the EMS applications discussed in Issue 20.1.3.

Mitigation Strategies

Procedural Mitigations: Limiting physical access to the EMS machines may reduce the opportunities an attacker would have to connect to the EMS databases and modify the audit logs. Ensuring that the EMS systems are never connected to a network may also reduce such opportunities.

Verification: We connected to the databases for each EMS application, BOSS, Ballot Now, Tally and SERVO, as described in Issue 19.1.2, and modified the audit logs to remove specific entries. We then replaced the removed entries with less suspicious **code** fields, a method which would arouse little or no suspicion on the part of one inspecting the logs.

20.2 Hart eCM Vulnerabilities

20.2.1 eCM keys may be quietly recorded to a debug file

The EMS applications (BOSS, Ballot Now, Tally, SERVO and eCM manager) all use a Spyrus library for accessing the keys stored on the eCM tokens. This library checks a Windows registry key for a path to a debugging output file. If found, the library will write debugging information, including the eCM key or part of the eCM key to the debug file. An attacker able to set this registry entry can obtain the eCM key used in a precinct, and thus create forged MBBs.

Description: BOSS, Ballot Now, Tally and SERVO all use eCM tokens (Spyrus Rosetta USB PKCS#11 cryptographic tokens) to compute cryptographic hashes. The EMS applications use a Spyrus library to access these tokens. This library checks a "Debug / Filename" registry entry. If the registry entry is found, the library will print debugging information to this file. This file contains the eCM key as well as the key GUID. From our observations, at no point in time during the execution of the EMS applications was the user informed that the eCM key or any other information was being written to a debug file.

For each EMS application, any operation that reads the eCM key causes 16 out of 40 bytes of the key to be written to the debug file. The eCM manager "Write Module" operation, used to write the key to an eCM token causes the entire key to be written to the debug file. Because this operation is used in the creation of eCM keys, an attacker able to set the debug registry entry any time before key creation can obtain the key that will be used in the precinct before ballot definition or creation has begun.

Prerequisites: In order to exploit this issue, an attacker will need physical access to the EMS machines, and know the proper registry entry to specify the path to the debug file. This is not difficult to find, as the registry key used for the debug parameter is located in the Spyrus DLL used by the EMS applications.

Impact: An attacker that can set this registry key can obtain the eCM signing key necessary to forge MBBs.

Procedural Mitigations: Strictly limiting physical access to the machines running EMS applications may reduce the opportunities for an attacker to create the Spyrus debug registry entry or subsequently be able to retrieve the key from the machine. Ensuring that the EMS systems are never connected to a network may also reduce such opportunities.

Verification: This issue was confirmed via demonstration with the EMS applications. We set the debug registry entry, and ran each application. After running the EMS applications, we checked the contents of the debug file, and found 16 bits of the key present in the debug file. After performing the eCM Manager "Write Module" operation, we found the entire key present in the debug file. We also verified that the path to the debug registry entry was in the Spyrus DLL on the EMS machines.

20.3 Hart eScan Vulnerabilities

20.3.1 The flash memory containing the eScan executable and file system can be replaced

The eScan maintains its firmware image and file system on a compact flash card. An attacker can change this card in under two minutes, completely subverting the operation of the eScan.

Description: The eScan's firmware and file system are located on a standard Compact Flash card. To access this card, an attacker must remove the screws around the base of the perimeter of the eScan. Two of these screws are covered by tamper evident seals. Upon removing these screws, the top of the eScan may be removed. The Compact Flash card, which is located beneath an IDE cable on a daughter board, may then be removed. There is one more tamper evident seal on the Compact Flash card.

Prerequisites: In order to replace the eScan internal flash memory, an attacker will need physical access to the eScan for two minutes.

Impact: An attacker able to replace the eScan's internal flash memory can completely control the eScan, thereby controlling election results and compromising voter privacy in a precinct.

Procedural Mitigations: Limiting physical access to the eScan as much as possible can reduce the opportunities for an attacker to replace the flash memory card. Tamper-evident seals are a potential *partial* mitigation to prevent unauthorized access. For practical limitations, see Section 3.5.

Verification: We loaded the Linux operating system onto a Compact Flash card, opened the eScan, installed the card and closed the eScan in under 2 minutes. We then successfully booted the machine with the new memory card.

20.3.2 The eScan runs a telnet server

The eScan runs a telnet server, a program that readily accepts network connections. Telnet is an old and insecure protocol, and there are known vulnerabilities in widely used telnet servers. An attacker able to communicate with the eScan's telnet server may be able to subvert the eScan.

Description: The telnet server running on the eScan is the Microsoft Windows CE Telnet service. Several vulnerabilities for other Microsoft telnet servers are known[4], suggesting the telnet server on the eScan may also be vulnerable. If an attacker knew of a vulnerability in the Windows CE Telnet server, it could be possible to compromise the eScan. We cannot speculate as to the purpose of the telnet server as we could not find any reference to it in the Hart documentation we received.

Prerequisites: In order to communicate with the telnet server, an attacker must have physical access to the eScan's Ethernet port.

Impact: An attacker able to leverage a vulnerability in the telnet server running on the eScan could possibly execute arbitrary code, completely compromising the machine. This would allow for the modification of election results and the compromise of voter privacy for a precinct.

[4]Microsoft Corporation, *Microsoft Security Bulletin (MS00-0500)*. July 2000 ⟨URL: `http://www.microsoft.com/technet/security/Bulletin/MS00-050.mspx`⟩; Microsoft Corporation, *Microsoft Security Bulletin (MS02-004)*. February 2004 ⟨URL: `http://www.microsoft.com/technet/security/Bulletin/MS02-004.mspx`⟩.

Procedural Mitigations: Access to the telnet server may be limited by blocking the Ethernet port on the eScan during elections.

Verification: We connected to the telnet server over port 23 and reached a login prompt. We could not successfully login to the telnet server.

20.3.3 The eScan scanner surface can be occluded to affect ballot processing

The eScan scanner surface can be accessed by lifting the unsealed device doors. An attacker can place inks or tape over certain parts of a surface to prevent ballots from being processed or votes for targeted races from being counted. In some cases, the voter will not be able to know that his votes were ignored.

Description: The eScan scanner feeds the voter's completed ballot between two small glass scanning windows above and below the paper. Optical scanners "read" both sides of the ballot as it is fed by paper rollers. In this way, the scanner scans the ballot either face-down or face-up.

An attacker who can paint opaque inks or place tape over the glass plates can affect the way the eScan processes ballots. The tape or ink is aligned vertically with the vote "boxes" in the roller feed direction. In this way, the optical sensor is blinded from seeing the boxes passing over the occluded area. The inks/tape can be placed such that the same races can be blocked when fed in either face-up or face-down.

The eScan glass plates are readily accessible. A door on the right side of the scanner can easily be opened and inks or tape aligned and applied in a few seconds. This attack may be difficult for an voter to mount as the eScan is likely to be attended by a poll worker. In a more subtle attack, a voter may be able to place the inks on a ballot, and then feed it through the scanner. The inks would then smear across the surface resulting in the same "blinding" of the optical sensors. This attack would likely be undetectable until the next voter votes.

When the ballot is scanned using black or reflective tape (or ink), the scanner indicates an over-vote of the first vote in the vertical column–the over-vote can be accepted by the poll official pressing the "over-vote" red-button on the back of the computer. However, instead of the one contest being over-voted *all* contests vertically aligned on the ballot are ignored as over-votes. The discarding of votes for the latter races occurs without voter notification. When the ballot is scanned with white or red tape/ink, is simply rejected as not being scanned properly.

Note that there is an option programmed into the MBB that allows election officials to disable any notification of over- or under-vote. Thus, in those precincts where this features was enabled, the votes would be uncounted silently; the voters would receive no indications that that their vote was not counted.

If permanent inks were used, the eScan would be unusable until its glass surface was replaced. This could seriously inhibit the processing of votes within a precinct.

Prerequisites: Access to the eScan, or in the voter attack, access to a ballot.

Impact: This would prevent votes for a given race from being counted, or disable the eScan entirely.

Procedural Mitigations: The door to scanning surface should be adequately sealed.

Verification: This issue was confirmed via red-teaming as described above.

20.3.4 The eScan ballot box collecting votes allows vote order to be reconstructed

The eScan optical scanner is placed in an "eScan Ballot Box", a large plastic tub that collects the paper ballot fed through the scanner. The ballots drop in one by one and there are no mechanisms in place to change the order in which they fall. Thus, specific voters may be targeted by noting the order in which they vote and examining the pile of ballots at the close of the election.

Description: During election day, an eScan optical scan unit is placed in a special housing called an eScan Ballot Box. This is a wheeled plastic tub that has a lock to keep the eScan in place and a side door that can be used to retrieve the cast paper ballots after an election. The door is locked on election day and when the eScan is in place, when a ballot is fed through it, the ballot subsequently drops through a small opening into the tub. The path of the paper when it is dropped into the tub generally causes it to drop into a pile that may be subsequently retrieved at the close of the election.

We examined the tub and discovered that there are no mechanisms within it to inhibit or otherwise change the way that ballots were dropped into the pile, or to (mechanically or otherwise) re-order or shuffle the ballots in any way. As a result, at the close of an election, many of the ballots will be in the pile in the order in which they were entered into the eScan, i.e., the first ballot cast will be at the bottom of the pile. While it is possible that some ballots may end up counted randomly, it appears that the majority of ballots will maintain their cast order.

We cast 10 ballots through the eScan machine that we numbered in order; upon checking the tub, we examined the pile of ballots and found that the order had been maintained. A watchful adversary who had access to the ballots after they had been cast would be able to report, with high probability, the way that a given voter had cast their ballot if they knew the order in which that vote had been cast.

Prerequisites: An attacker would need to watch the order in which votes were cast at the eScan machine, then be able to access the pile of ballots at the end of the election. A malicious poll worker, for example, could determine the order of a cast ballot and make note of the way in which that voter had voted.

Impact: This allows a compromise in voter privacy, as an attacker could determine the way a vote was cast. This attack could be used for the purposes of vote-buying or coercion.

Procedural Mitigations: The election administrator should shuffle the pile of ballots in the eScan ballot box at the close of the election. This will have limited effect if the administrator or any workers observing the pre-shuffled state of the ballots is corrupt.

Verification: This issue was confirmed via red-teaming as described above.

20.3.5 The eScan ballot box is vulnerable to attacks that may destroy ballots

As described in issue 20.3.4 the "eScan Ballot Box" is a large plastic tub that collects the paper ballot fed through the eScan optical scan unit. The ballot box has an "emergency ballot slot" for use if power to the eScan goes out; voters can cast their ballots through this slot into a small, removable tub. This tub has a hole in it, allowing an attacker to pour corrosive liquid into the unit through the emergency slot and having it leak onto the main pile of ballots, which can be dissolved or severely damaged.

Description: The emergency ballot slot on the eScan Ballot Box is meant to be used if the eScan becomes inaccessible, such as in the event of a power outage to the unit. A tub is installed inside the ballot box in order to collect ballots cast this way, which are meant to be counted and tallied at election headquarters, since the

eScan is incapable of tallying them. The door to access both the pile of ballots that fall into the box from the eScan and the tub containing ballots cast through the emergency slot is locked. A hole approximately one inch in diameter is in the tub on the opposite side from where the emergency ballot slot on the ballot box is located. This hole is presumably used to easily handle the tub so that it may be easily removed from the ballot box.

The tub is located above where the ballots normally cast through an eScan would collect. By partially filling the tub with a quantity of corrosive liquid such as acid-based drain cleaner, with a high percentage of sulfuric acid, the liquid would fall through the hole and onto the paper ballots below, dissolving them. Additionally, the unit is top-heavy because of the weight of the eScan on top of the ballot box. If someone was to fall into the unit can cause it to tip a few degrees, this would be sufficient for even a small quantity of a corrosive agent to slide down into the hole and subsequently onto the ballots beneath. The emergency ballot slot is approximately 8.5 inches wide by 1 inch high, which is more than sufficient for an attacker to surreptitiously pour a noxious substance into the tub. The gate to this ballot slot can be closed when emergency balloting is not in used, but there is enough of a gap that flexible tubing may be inserted through for liquid to be poured in.

Prerequisites: An attacker needs to have access to a corrosive liquid such as acid-based drain cleaner. They would need to be able to pour the liquid into the tub, which could be surreptitiously performed while they are voting at the eScan machine.

Impact: This attack could cause the paper ballots to be severely damaged or destroyed outright. Because the paper ballot is the official ballot of the election, destroyed ballots inside the voting machine could cast the results in significant doubt.

Procedural Mitigations: The emergency ballot slot should be sealed shut. Ballots should be collected in another fashion if there are power or technical issues with the eSlate unit.

Verification: We poured liquid into the collection tub and simulated falling into the machine to knock it askew and the liquid into the hole. We did not test the attack with sulfuric acid.

20.3.6 The eScan may be modified to allow casting of duplicate ballots

The configuration file for the eScan may be modified to allow the eScan to cast duplicate ballots. We confirmed in Section 19.4.1 that the eScan file is accessible over an Ethernet port and may be both downloaded and uploaded. A configuration parameter in this file allows duplicate ballots to be scanned, overriding controls in the eScan that should otherwise prevent this from occurring.

Description: The configuration file is common to all eScan devices. It is a text file that is found on the eScan, accessible by connecting to the eScan's Ethernet port with the correct IP address. [5] An option in the configuration file to allow duplicate ballots to be accepted is commented out by default. Setting this option in the configuration file and uploading it to the eScan via the Ethernet interface will enable the counting of duplicate ballots. These can be photocopies of a single ballot, or the same ballot used multiple times, as described in Issue 20.3.9. The only indication of a duplicate ballot having been scanned is the fact that the serial number of the ballot is logged in the eScan's audit log. Unless the log was closely scrutinized, however, it would be difficult to notice that multiple ballots with the same serial number were cast, particularly if they were interspersed amongst many legitimate votes.

It is not clear why an option to allow duplicate ballots is available on production eScan units. This may be

[5] More information about connecting to the eScan may be found in Issue 19.4.1.

a testing feature that is not meant to be enabled; however, the functionality is easily accessible.

Prerequisites: An attacker needs to have access to the Ethernet port of the eScan and a computing device, such as a Palm handheld, to connect to the port and upload a new eScan file. The attacker would then need access to a single ballot which may be arbitrarily duplicated.

Impact: This attack allows for "ballot stuffing" with an arbitrary number of duplicate ballots. If combined with attacks to erase or modify the audit logs of the eScan after the election, as detailed in Chapter 21, these ballots will not be detected except by manual recount, and only if the examiners closely note the serial number of the ballots.

Procedural Mitigations: The configuration files of all machines should be be downloaded and examined before an election. The Ethernet port on eScan devices should be sealed to prevent access during an election.

Verification: We retrieved the configuration file from the eScan unit, modified the file to allow duplicates, and uploaded it back to the eScan. We then scanned a ballot into the eScan repeatedly and it worked every time. We also made photocopies of the original ballot and scanned those in. The eScan accepted every ballot.

20.3.7 The eScan has an open interface allowing erasure of vote and audit log records

The eScan is accessible through its Ethernet port, as described in Issue 19.4.1. Using a device such as a smartphone or a handheld computing device (e.g., a Palm device), an attacker can issue commands to the eScan that include resetting it and clearing its internal records and logs.

Description: Before an election occurs, all voting equipment is initialized using the SERVO application. Part of the initialization process includes clearing the cast vote records (CVRs) and audit logs from the voting equipment, notably the eSlate, JBC, and eScan devices. The eScan is managed by SERVO through its Ethernet interface. Performing the erase action does not require the use of an eCM cryptographic token.

With a computing device such as a Palm handheld computer and a cable that will interface to the Ethernet port on the eScan, an attacker can mimic the actions of SERVO to the eScan during an election, causing the CVR and audit log records to be erased from the machine. There is no authentication performed to determine the machine interfacing with the eScan. So, these commands will be executed by the eScan regardless of the device that issued them. Any voting that has occurred on the machine to this point will be erased from the eScan's internal logs as well as from the MBB.

Prerequisites: An attacker needs to have access to the Ethernet port of the eScan and a computing device, such as a Palm handheld, to connect to the port. If an attacker is left unsupervised for only 30 seconds or less at the polling place, this attack would be possible. A poll worker could easily mount the attack if they have access to the eScan for approximately 30 seconds.

Impact: Any electronic evidence of voting having occurred on the eScan will be removed from both the eScan's internal logs and the MBB with this attack.

Procedural Mitigations: The Ethernet port on the eScan should be sealed on election day, as there is no need for it to be accessible by any device or application that is not SERVO. Poll workers should never leave the eScan unguarded. However, if the attack is carried out by an insider, this mitigation will not work.

Verification: This vulnerability was confirmed through source code analysis. We also wrote a program to allow us to reset the eScan from a laptop computer connected through the eScan's Ethernet interface. With

this tool, we were able to issue commands to the eScan and cause a full reset of audit and CVR logs on the eScan.

20.3.8 The Premier ballot box key works in the Hart ballot box

As described in Section 14.4.3, while the Premier and Hart ballot boxes that we received are significantly different, the same key works in both. If it is the case that all Hart and Premier ballot boxes use the same key, then if an attacker gains access to the ballot box key in one county, other ballot boxes within the state, regardless of the vendor, could potentially be compromised.

Description: Both Premier and Hart InterCivic have precinct optical scanners on top of which a small scanner device is secured. The ballot box appears to be custom made for each vendor and, in the case of Premier, performs special functionality (sorting write-in ballots from non-write-in ballots). There is no special functionality in the Hart ballot box; it is merely a collection area for ballots. We received keys for the ballot boxes from the Secretary of State and discovered that they are interchangeable between the two boxes. A photograph of these ballot boxes can be found in Section 14.4.3, which also shows that the same key works in both ballot boxes. Both of these ballot boxes that we received have multiple locks, and they are all accessible with the same key.

Prerequisites: Access to the ballot box key in one county.

Impact: If the ballot boxes do take the same keys throughout the state, then Ballot boxes in other counties within the state, regardless of the vendor, can be compromised.

Procedural Mitigations: Each ballot box should have unique keys.

Verification: This issue was confirmed via demonstration with the ballot box.

20.3.9 A voter can cast a ballot and then retrieve it from the ballot box

After casting a ballot on the eScan, a voter should not be able to gain access to their ballot. However, it is possible for a voter to do precisely this without opening the ballot box. If a machine is set to accept the same ballot more than once, an attacker could cast multiple ballots. If in its normal operating mode, the attacker could cause a discrepancy between the electronic and paper vote counts, thereby casting doubt on the results of an election.

Description: After an eScan unit reads the choices encoded on a voter's ballot, it attempts to drop that ballot into the locked ballot box. This allows elections officials to perform a recount if the results of an election are in doubt. However, it is possible for an attacker to have the eScan count their ballot and then retrieve it from the ballot box. By attaching a long strip of Post-It notes to the ballot, the attacker can simply retrieve their ballot from the eScan after it is recorded.

Prerequisites: An attacker would need a few seconds of access to the eScan at the polling place.

Impact: If the eScan allows a ballot to be cast multiple times (see Issue 20.3.6), an attacker could use a single ballot to cast multiple votes. If the machine were instead running in its standard configuration, in which a single ballot could only be cast once, the attacker could remove their ballot after it was counted. If a recount were required, the number of votes cast on the eScan would be greater than the number of ballots in the ballot box. Such an attack could cast further doubt on the accuracy of the election.

Procedural Mitigations: Because of privacy regulations, elections officials should not be able to see a voter cast their ballot. We can therefore not suggest any procedural mitigations.

Verification: We created a daisy-chain of Post-it notes approximately 1 foot in length. The attack is made easier if tape is added to the Post-It notes as it helps them hold together better.

20.4 Hart JBC Vulnerabilities

20.4.1 The JBC can rapidly create an unlimited number of access codes on election day

The Voter Registration Interface, or VRI, is a serial interface on the back of the JBC. We found that on election day, it is possible to use this interface to rapidly enter an arbitrary number of requests for access codes to the JBC, which can be guessed and used to vote on the eSlate machines.

Description: The California TTBR Source Code report showed that it was possible to generate access codes when the JBC is in early voting mode; we confirmed this vulnerability as Issue 19.7.4 in our report. The California team found that performing the same operation on election day was different because the JBC expects a barcode scanner to be attached to the VRI. While in early voting mode, it is possible to not have codes printed out but only returned over the serial interface, for Election Day voting, an access code would be printed out from the JBC printer. This limits the number of codes that can be generated.

We found, however, that it is possible to disable the printer on the JBC. If a report such as "Access Code Report" is requested from the JBC while there is no paper in the printer (e.g., when the paper on a roll runs out, or the roll is removed), the JBC informs the user that no paper is in the printer, and an option is provided to disable the printer. We wrote a program to generate access code requests through the VRI, and when the printer has been disabled, these can be submitted at a rapid rate to the machine; at least 300 codes a minute may be submitted this way. While the JBC documentation claims a maximum of 150 outstanding voter access codes, we found that we can issue as many codes as we desire.

After the codes have been requested from the JBC, the attacker can cause the JBC to resume normal operation. The roll of paper can be reinserted at any time after the "Disable Print" function is selected from the JBC, as nothing will be printed. When the attacker wants to allow the printer on the JBC to resume functioning, the "Enable Printer" option on the JBC menu can be selected.

While the codes are not printed out or returned over the interface, if a sufficiently large number of codes are printed, an attacker can guess an active code. For example, if 1000 codes are generated then on average, 1 out of every 10 guesses will result in a valid access code. Furthermore, as we confirmed from the California TTBR Source Code report in Issue 19.7.6, the way that access codes are generated is not random, so once one code is guessed correctly, the attacker can enter a correct sequence of access codes from that point. We also found that if access codes expired, they could be easily regenerated, so this attack could be performed repeatedly if the expiration time for the codes was set to be shorter than the period in which the polls are open.

Prerequisites: The attacker must be able to access the serial port on the JBC and possess a device, such as a PDA, to run a terminal emulator program. They must also know the precinct ID. They must also be able to remove the paper from the JBC printer, perhaps under the auspices of changing the roll, and be in proximity to the JBC in order to press the buttons corresponding to the "Disable Printer" option, and subsequently, the "Enable Printer" option.

Impact: On election day, a voter would be able to vote an arbitrary number of times before the access codes expire.

Procedural Mitigations: The serial interface on the JBC should be sealed as it is not used in Ohio.

Verification: This vulnerability was confirmed through source code analysis. We also wrote a program to request access codes from the JBC in election day mode and tested the attack by disabling the print function.

20.4.2 The JBC has an open interface allowing erasure of vote and audit log records

The back of the JBC unit has a connector for a parallel port interface. Using a device such as a smartphone or a handheld computing device (e.g., a Palm device), an attacker can issue commands to the JBC that include resetting it and clearing its records and logs, in addition to those of any connected eSlates.

Description: Before an election occurs, all voting equipment is initialized using the SERVO application. Part of the initialization process includes clearing the cast vote records (CVRs) and audit logs from the voting equipment, notably the eSlate, JBC, and eScan devices. SERVO often runs on a laptop that has a PC card slot. A Quatech SPP-100 card with a connecting cable is used to interface the laptop with the parallel interfaces on the JBC. Performing the erase action does not require the use of an eCM cryptographic token.

With a computing device such as a Palm handheld computer and a cable that will interface to the parallel port on the JBC, an attacker can mimic the actions of SERVO to the JBC during an election, causing the CVR and audit log records to be erased from the machine. Because the JBC controls the eSlates as well, the attack can clear the eSlate machines of all of their records as well. There is no authentication performed to determine the machine interfacing with the JBC, so any device that issues these commands will have them accepted by the JBC. Any voting that has occurred on the machine to this point will have its record erased. In addition, the MBB inside the JBC will also have its records and logs cleared.

In combination with vulnerability 20.5.4, where we showed it is possible to remove the paper record from the VBO unit, this attack will remove all evidence of any voting haven taken place on the machines at all, as the internal memory on the eSlates and the JBC will have been initialized, the MBB will have been cleared, and the paper record will have been removed.

The eSlates will not be functional until they are reset after they have been initialized; however, by unplugging the JBC and plugging it again, the eSlates will become functional again. If the attacker knows the startup code, the polling place ID, and the access code to open the polls, the election will be able to continue without arousing suspicion. In addition, the JBC itself will operate as normal when its totals have been reset; there is no evidence on of any reset operation having taken place when the polls are open.

Prerequisites: The attacker must be able to access the parallel interface on the JBC and possess a device, such as a PDA, to run a terminal emulator program so they can send commands to the JBC. This attack could be performed in under 30 seconds.

Impact: Any electronic evidence of voting on any eSlate connected to the JBC having occurred will be removed with this attack. In addition, if the paper records are removed from the VBOs connected to any eSlate attached to the JBC, all record of voting to that point will have been removed.

Procedural Mitigations: The parallel port on the JBC should be sealed on election day, as there is no need for it to be accessible by any device or application that is not SERVO. Poll workers should never leave the JBC unguarded, though if the attack is carried out by an insider, this mitigation will not work.

Verification: This vulnerability was confirmed through source code analysis. We also wrote a program to allow us to reset the JBC and eSlate machines from a laptop computer connected through a parallel port interface; with this tool, we were able to issue commands to the JBC and cause a full reset of audit and CVR logs on the JBC and MBB. We entered a number of votes through the eSlate before running the tool. Once we ran the tool, the JBC continued to function but closing the polls showed the final tally of counted votes to be 0 for each candidate. In addition, the MBB read zero votes when we inserted it in Tally and the eSlate and JBC internal memories did not have evidence of an audit or CVR log when we connected them to SERVO.

20.4.3 JBC/eSlate voting can be completely automated

The JBC and eSlate posses the capability to accept simulated button presses via their parallel and serial interfaces respectively. These simulated button presses are interpreted by the JBC and eSlate the same as real button presses, allowing voting to be completely automated. The results of this automated voting are recorded to the VVPAT and an election mode MBB that can be tallied by Tally. An attacker can write a computer program which will automatically cast votes using this interface, thus significantly or completely controlling the election results at a polling place.

Description: The JBC and eSlate accept simulated button presses over their parallel and serial port interfaces respectively. All buttons on the JBC, and the buttons and wheel on the eSlate may be simulated. These simulated button presses are sent to the JBC and eSlate using the same protocol used to perform administrative functions such as resetting the JBC. As described in Issue 19.7.3, there is no authentication between the JBC, and the machine commanding it over this interface. Thus, any device connected to the JBC over its parallel interface may send it button presses which will be interpreted the same as physical button presses. Simulated button presses and wheel turns may also be sent to the eSlate through the JBC. An attack program using this interface would work as follows.

- A voter code is requested from the JBC over the parallel port interface.
- This voter code is then entered on the eSlate by sending simulated wheel turns.
- Simulated wheel turns are used to select options for each contest.
- A simulated cast ballot button press is used to finalize the choices.
- Repeat the above four steps.

Each time the above program votes, the results are recorded on the VVPAT and the MBB. These results may then be tallied from an election mode MBB in Tally.

Prerequisites: In order to exploit this issue, an attacker will need physical access to the JBC. The attacker will also need to know the ballot layout so that the correct number of wheel turns can be programmed.

Impact: An attacker able to send simulated button presses and wheel turns to the JBC and eSlate can largely influence election results at a polling place. Because each automated vote requires a voter code to be requested over the JBC's parallel port interface, this attack could most feasibly be executed by a poll worker. Any time an eSlate is not being used, a poll worker may use it to cast automated votes with this method. The poll worker may take a more active approach and mark an eSlate as out of order, allowing it to be used exclusively for automated voting.

Procedural Mitigations: Locking the parallel port interface on the JBC and eSlate will make it harder for an attacker to use it to send simulated button presses to the JBC and eSlates. Removing the parallel port

from the JBC altogether during an election will prevent it from being used to send simulated button presses to the JBC and eSlates. In addition, randomizing the order of candidates on the ballot may significantly mitigate this issue. [6]

Verification: We verified this issue by creating a `jbckeys` program to automatically perform votes via simulated button presses on the eSlate. These votes were evident on the eSlate and JBC's public counters, the VVPAT, the JBC's unofficial tally printout, and the CVRs from the votes were tallied by Tally and reported indistinguishably from regular votes cast in the same election. `jbckeys` contains approximately 200 lines of new code and took slightly over an hour to write. An additional hour was required to port the Hart parallel interface code to the Linux operating system.

20.5 Hart VBO Vulnerabilities

20.5.1 The VBO Printer is controlled via an accessible 1/8" port

The VBO Printer connects to the eSlate via a 1/8" jack. An attacker able to connect to the printer via this jack can control the VBO printer.

Description: The eSlate controls the VBO printer via a 1/8" port in the back of the VBO. This port can be accessed to pressing the release button at the top of the VBO and lifting it out of the chassis. While the removal of this connection will cause the LED on the JBC corresponding to the disconnected VBO to blink, this can go unnoticed if the polling place worker is distracted by another voter.

Some of the actions that can be performed over this port are as follows.

- Sending messages to be printed to the VVPAT
- Checking if the printer is low on paper
- Setting the VBO printer serial number
- Printing debug information
- Checking for error conditions in the printer.

The eSlate side of the connection is a standard 6-pin data input. Because the VBO is a "dumb" device, and it does not attempt to authenticate the machine sending it commands.

Prerequisites: In order to exploit this vulnerability, an attacker must have physical access to a VBO, and have understanding of the VBO protocol.

Impact: An attacker able to control the VBO printer can print arbitrary data to the VVPAT. This would allow forged votes to be added to the VVPAT, defeating the purpose of the VVPAT. Note that an attacker does not need to control the eSlate to exploit this vulnerability.

Procedural Mitigations: A tamper evident seal could be placed between the VBO and the chassis containing the eSlate and VBO, to make it evident if a VBO has been removed from the chassis. Tamper-evident seals are a *partial* mitigation to protect the VBO. For practical limitations, see Section 3.5.

Verification: This vulnerability was verified through source code inspection. We have not attempted to exploit this vulnerability.

[6] Random ordering is mandated in certain states such as California.

20.5.2 The VBO printer can be disabled by cutting the wires to it

The VBO (Verified Ballot Option) printer is the name Hart gives to the printer attached to an eSlate device. This provides a verified voter paper audit trail, or VVPAT, allowing the voter to view their candidate selections on paper as well as on the electronic screen of the eSlate. The communication between the eSlate and the VBO printer can be quickly and easily severed by an attacker with a sharp object, such as a knife or scissors.

Description: There are two cables that connect into the VBO: a power cable and a data cable. The stand for an eSlate includes cables tied down inside it, one which connects to a port that a connection from a power supply is plugged into, and one that connect behind where the eSlate is mounted, where an interface point for its contacts are found. If either of these cables is severed, the eSlate will be rendered inoperable, as it requires a connection to the VBO printer to function.

If an attacker successfully severs either the data or power connection, after 15 seconds the eSlate will report the message "Contact Poll Worker" followed by the line "Printer Unavailable" and below this, "Printer Status: EVBO-105". According to vendor documentation [7], the troubleshooting guide shows this to be a communication error and outlines a series of steps for the poll worker to undertake. These involve actions such as re-seating the eSlate inside its booth and cleaning the contacts the eSlate connects with. None of these steps will diagnose the problem of a severed cable, particularly if the attacker is skilled and resplices the cable such that it is not apparent that it has been tampered with. Because of the privacy guard available to the voter, an attacker with good manual dexterity can quickly remove the VBO from its housing on the booth by pressing a large black button that pops out and turning it counter-clockwise, and quickly cut the cables attached to the bottom of the printer as it comes out, all without being seen. This attack would take no more than a few seconds.

Prerequisites: The attacker must have access to the eSlate with an attached VBO printer, which every eSlate in Ohio should have. They must have a sharp object such as a pocket knife with them and 30 seconds or less to cut the cables.

Impact: The eSlate machine will be unavailable to voters until technical support personnel from outside the precinct are available to diagnose the problem and replace the cables. The recommended solution to problems with the VBO is often to change the unit[8]. If the attacker skillfully respliced the cable, it may be difficult and time-consuming to find the root cause of the error, knocking the eSlate out of service until the problem is found, if it is. With 15 seconds before the unit sends an error code, the adversary could quickly exit the polling place before any malfeasance is suspected.

Procedural Mitigations: Extra cables should be kept on hand and technical support personnel should be aware of checking and replacing the cables in case of malicious activity against them. Poll workers should be vigilant in examining the LEDs on the JBC in case of errors, as the LED of the affected eSlate will blink red and green if an error occurs.

Verification: We simulated this attack by removing the power and data cables in turn and together. We did not physically sever the cables.

Note: This issue does not have any unredacted content. There is no corresponding section in the private appendices.

[7] *eSlate(tm) Polling Place System Election Day Desk Reference (System Version 6.2)*

[8] Hart Intercivic, *Hart Voting System Support Procedures Training Manual, System Version 6.2*. page 169.

20.5.3 Damaging contacts on the eSlate booth can cause the eSlate-JBC connection to fail

The eSlate is connected to the VBO printer through a set of six contacts in the booth that the eSlate is housed in. If the contacts are destroyed or pulled out, communication with the printer will not be possible.

Description: The cable connecting the eSlate with the VBO printer is housed inside the eSlate's booth. This cable has six pins that allow for transfer of data and commands between the two devices. An attacker can gain access to this contact surface when the privacy shield of the eSlate's booth is raised. The eSlate can be lifted out of the housing, and is designed to do so in order to enable "curbside voting", a procedure that is not followed in the state of Ohio.

If an attacker successfully snaps the pins or severely damages the contacts given access to the surface, after 15 seconds the eSlate will report the message "Contact Poll Worker" followed by the line "Printer Unavailable" and below this, "Printer Status: EVBO-105". As described in Issue 20.5.1, the troubleshooting procedure will involve examining the contact surface. When it has been found to be damaged, the eSlate and the printer will be inoperable until a new booth is retrieved to mount the eSlate in.

Prerequisites: The attacker must have access to the eSlate with an attached VBO printer, which every eSlate in Ohio should have. They must be able to lift the eSlate out of its housing in a surreptitious manner and have a tool in hand, such as pliers, to snap the contact pins or otherwise render them inoperable. This could be carried out within a matter of seconds.

Impact: The eSlate machine will be unavailable to voters until a technical support personnel from outside the precinct are available to diagnose the problem and replace the booth, as the VBO will not be operable until the connection is restored and the eSlate will not operate without the connection to the printer. During this time, no voting on this machine will be possible.

Procedural Mitigations: Extra booths should be kept on hand and technical support personnel should be aware of checking and replacing the data cables and booths in case of malicious activity against them. Poll workers should be vigilant in examining the LEDs on the JBC in case of errors, as the LED of the affected eSlate will blink red and green if an error occurs. The eSlate should be more fully secured as there is no reason in Ohio why it should be able to be easily lifted out of its housing, given the lack of curbside voting.

Verification: We removed the eSlate from its booth and noted the subsequent communication error between the eSlate and the VBO printer. We did not snap or otherwise destroy the contacts.

Note: This issue does not have any unredacted content. There is no corresponding section in the private appendices.

20.5.4 The paper roll in the VBO can be tampered with or removed

The VBO printer contains a roll of paper, protected under a plastic screen, that a voter can use to verify the choices they made with the eSlate machine. It is possible for an attacker to open the VBO unit and remove the roll of paper, thus removing any paper trail of the votes made to that point.

Description: The VBO printer is attached to an eSlate machine by a data cable, such that votes for a candidate made on the eSlate are printed on the corresponding VBO unit. This allows the voter to verify their voting preferences on paper. The VBO unit may be removed from its housing in the eSlate booth by pressing a large black button on the top of the unit, which pops the button out, and turning it either left or right. The VBO unit can then be handled by the voter. On the back of the unit is a pair of screws that have a

plastic cladding around them, allowing them to be turned by hand. If these are removed, the interior of the VBO unit can be accessed.

Within the VBO unit is the print head assembly and two spools of paper, one fed through the print unit and the other a take-up spool that contains the record of all printouts on the roll. The spools come apart easily by twisting, allowing the roll of paper on the take-up spool to be easily removed by tearing the roll. An attacker can then reattach the roll of paper to the take-up spool such that the VBO unit is functional again.

This attack can be performed in approximately one minute by a skilled attacker with some manual dexterity. It is feasible by an average voter because of the privacy shield for the eSlate booth. Because the paper ballot is the legal ballot in the state of Ohio, removing the roll of paper removes the official ballots of everyone who has voted on the eSlate machine that day.

Prerequisites: The voter needs access to the VBO printer and some degree of manual dexterity to prevent them from dropping the unit and causing suspicion. They also need to be able to place the roll of paper in their pocket or elsewhere on their person without being caught.

Impact: Without the roll of paper providing the verification, the official voter ballot will not exist.

Procedural Mitigations: The VBO unit is not meant to be accessible to anyone in a polling place during an election; Hart procedures call for it to be removed altogether and replaced with a new unit in case of any problems. There are no current mitigations from a procedural standpoint. The LED associate with the eSlate being attacked may flash on the JBC but this may not happen. In either event, the best hope for mitigation is a general vigilance amongst poll workers to strange sounds or appearances of activities, and constant monitoring of the JBC LEDs.

Validation: We were able to open the VBO unit with no tools by removing it within the eSlate housing, and remove the tape with no special equipment. This was possible on the first attempt to do within approximately two minutes. It would be possible to perform this faster with some training.

Note: This issue does not have any unredacted content. There is no corresponding section in the private appendices.

20.5.5 The serial number of the VBO can be modified

The VBO maintains a serial number which is printed to the VVPAT at different times throughout an election. This serial number can be modified through several different means. An attacker able to modify this serial number during an election may cast doubt on the integrity of the VVPAT.

Description: The VBO Serial number is of the form V#####, where # is in 0-9, A-Z or a-z, and should match the serial number printed on the sticker on the VBO. The serial number is printed on the VVPAT at the following times.

- At eSlate or VBO power on time
- When the Polls open
- With every cast vote
- When the Polls close

There are three interfaces through which an attacker may issue the command to change the serial number.

- The parallel port on the JBC
- The serial port on the eSlate
- The 1/8" port on the VBO as described in Issue 20.5.1

Prerequisites: In order to leverage this issue, an attacker will need access to either the parallel port on the JBC, the serial port on the eSlate or the 1/8" port on the VBO.

Impact: When the serial number is changed, all votes occurring after the change will appear on the VVPAT with the modified serial number. This will call the validity of these votes into question. Changing the serial number before an election begins could result in a VVPAT being matched with an incorrect VBO at a later date, as the serial on the VVPAT will no longer match that on the exterior of the VBO.

Procedural Mitigations: Limiting physical access to the parallel port on the JBC, the serial port on the eSlate and the 1/8" port on the VBO may reduce the opportunities for an attacker to change the serial number through these interfaces.

Validation: We used the JBC parallel port interface to change the VBO serial number. We have not attempted to use the eSlate serial port or the 1/8" port on the VBO.

20.5.6 The VVPAT paper record may be forged

The paper tape generated by the VBO device can be generated by any generic thermal printer; there is nothing to differentiate this record. Anyone with access to a thermal printer, a roll of the correct width, and knowledge of the barcode generation format can create a paper tape indistinguishable from a legitimate one created by a VBO.

Description: The VBO generates voter choices and when the vote is accepted, prints "BALLOT ACCEPTED" on the tape, followed by a two-dimensional barcode written in the standard PDF-417 format (the barcode is printed with "BALLOT REJECTED" above it if the ballot was not accepted). [9] By examining the barcodes generated by different cast ballots on the paper tape, an attacker will be able to reconstruct the format in which barcodes are written to tape. A full description of the barcode format is provided in the Hart VBO Functional Specification.[10] The barcode is not encrypted in any way. The rest of the paper tape is plain text and can easily be forged.

Prerequisites: An attacker needs access to a thermal printer and a roll of thermal paper 4 inches in width. The attack may be completely undetectable if the attacker knows the serial numbers of the VBO units at the precinct and can have those written out to the paper roll. An attacker will need to either be able to replace the VBO roll at the precinct (see Issue 20.5.4 for details) or at the precinct when the tape is removed from the VBO unit.

Impact: The VVPAT can be forged, and in conjunction with a forged MBB, provides a full set of forged election results. Unless a full recount that involves the internal memory of the JBC or eSlate is performed,

[9] The role of the barcode is unclear for elections in Ohio, as Ohio's recount procedures are vague on the question of permitted uses of barcodes during recounts. An Ohio Secretary of State directive stipulates that hand counting must be done by "physical examination of the VVPAT roll", and permits "electonic tabulation" if the hand count shows no discrepancy. It is unclear if "electronic tabulation includes counting VVPAT ballots with a barcode reader. *See* Ohio Secretary of State, *Directive 2007-30: Recount Procedures*, (2007), Retrieved November 25, 2007, from `http://www.sos.state.oh.us/sos/ElectionsVoter/directives/2007/Dir2007-30.pdf`.

[10] Hart Intercivic, *VBO Functional Specification, Rev. 10-62A*..

there is a high probability the forgery may go undetected.

Procedural Mitigations: The chain of custody of the VBO unit must be rigorously overseen. The VBO should never be left out of sight, particularly when an election is complete. Chain-of-custody logs are a *partial* mitigation to protect the VBO. For practical limitations, see Section 3.5.

Validation: We validated this attack through source code analysis.

20.6 Hart Tally Vulnerabilities

20.6.1 Tallied MBBs can be used in election

Tally keeps track of all the MBBs it has tallied in an election. Tally will not allow a previously tallied MBB from being processed again. If an MBB is marked as counted within the Tally database before the election begins, then the votes residing on it after the polls will not be counted.

Description: Tally and BOSS share a database that contains information on all MBBs used in an election. One of the fields in that database indicates whether the MBB has been counted by Tally. Once the MBB is read and the votes counted, the database sets that information to indicate that the data has been read. Thereafter, Tally will not allow the card to be read again.

The MBB "read" state information can be abused to prevent votes from being counted. A pre-tallied MBB can be used normally within a precinct; there is no notification on the JBC or eScan that an MBB has been tallied before. After the polls have closed, the MBB will not be able to be tallied, and the votes on it will not be counted. An attacker might, for example, use this feature to disenfranchise voters by:

1. An MBB is created for an election using the BOSS application.
2. The election is "Finalized for Tally" via BOSS on the back-end host.
3. The created MBB is tallied by inserting it Tally and "counting" the votes on it. There will be no votes on the MBB because it is freshly created.
4. The MBB is inserted into a JBC or eScan and used at a precinct. There is no notification at the precinct that the card has been previously used.
5. The election is carried out normally and the MBB returned to the election headquarters for tallying.
6. The MBB will be rejected by Tally, and the votes will not be counted.

Access to the Tally database alone may enable an attacker to mount a similar attack. They may simply mark the MBBs of the precincts they do not want counted as already being tallied. Thus, when the polls close, the associated MBBs returned from the field will be judged as counted, and the associated votes discarded. We have not attempted to confirm that this attack by manipulating the system directly.

An attacker who can forge MBBs might also be able to mount a similar attack. If the forged MBB falsely indicates it is the targeted victim precinct/device and it is counted first, then the legitimate MBB will be discarded.

Note that an election official may simply ignore such an attack; it is reasonable to assume that election officials occasionally (and possibly commonly) accidentally insert the same MBB more than once. Officials faced with potentially hundreds of such devices may simply lose track of those they have and have not used.

Prerequisites: The attacker needs to have access to the MBB (or possibly only the Tally database) prior to the election.

Impact: This allows an attacker to prevent an entire precinct's votes from being counted.

Procedural Mitigations: Very careful records should be kept of MBB processing by Tally. Any MBB rejected from being processed by Tally should be set aside, and the issue documented and resolved by viewing the available audit records.

Verification: We confirmed this via source code analysis and via demonstration of the enumerated attack described above.

20.6.2 The Tally user interface is completely configured in the registry

Tally uses registry entries to store configuration information controlling the display and behavior of user interface components including the menu bar and tabs which display all of Tally's functionality. An attacker able to modify these entries could render Tally useless or introduce subtle changes that could lead to user error.

Description: Tally has two main user interface components, the menubar which contains the *File*, *View*, *Options* and *Help* menus, and the tabs which contain all content including *Election Databases*, *Election Information*, *Reporting*, *Write-In Resolution* and *Provisional Ballots*, all of which are also accessible from the *view* menu. All of these components are configured with registry keys which if renamed or deleted result in the component becoming unavailable in Tally. Renaming or removing all such registry entries results in Tally becoming completely unusable, as all functionality is unavailable.

The registry entries for each menu item appear as a list of numbers, each one containing a name to be displayed for the menu item. An attacker can change these numbers to make the menu items unavailable upon Tally startup. In the simplest such attack, an attacker would change menu item number 1 to a larger number such as 100, causing the Tally menu loader to miss all menu items.

These registry keys are also used to define the behavior of each component. For each user interface component, a registry key lists the dynamic link libraries, DLLs used to implement its functionality. This can be used to create subtle errors in Tally's behavior such as the import menu item launching the export dialog box. A mistake made by a human user due to such a misconfiguration could be difficult to diagnose and correct.

In addition, Tally contains a pluggable architecture that presents seemingly-unbounded functionality. Examples are given in the source code for how to build components of Tally that will be activated at runtime. This is part of an entire configurable and modular architecture that can be invoked with this application. This also allows for any externally created module to be bound into Tally and gain its functionality, creating potential vulnerabilities that measures such as static code analysis will not be able to detect.

Prerequisites: In order to exploit this issue, an attacker will need physical access to the EMS machine running Tally.

Impact: An attacker able to modify the registry entries can make Tally completely unusable or make changes in its behavior leading to potential mistakes by human operators.

Procedural Mitigations: Limiting physical access to the EMS machine running Tally may reduce the opportunities for an attacker to tamper with Tally's interface through the registry. Ensuring that the EMS

systems are never connected to a network may also reduce such opportunities.

Verification: We placed the letter 'e' at the end of each UI component's registry key. This resulted in Tally having no usable functionality when it was started. We then changed the DLLs mapped to each menu item, resulting in hard to spot inconsistencies in the UI, such as the import menu item launching the export dialog.

20.7 Hart Ballot Now Vulnerabilities

20.7.1 The Ballot Now username and password can be bypassed

Ballot Now usernames and passwords are stored and checked in a database. An attacker with no special permissions can modify this database to allow arbitrary users to login to Ballot Now.

Description: Ballot now as well as the other Hart EMS applications store usernames and passwords in a *Sybase* database. As shown in Issue 19.1.2, it is possible for anyone with physical access to an EMS system to connect to these databases, and modify the contents. When a user logs in to Ballot Now, the username and password hash are sent to a stored procedure in the database to be validated. Using the database passwords extracted as described in Issue 19.1.2, it is possible to connect to the database and modify the stored procedure to allow any user to connect using any username and password, or none at all.

Prerequisites: In order to exploit this vulnerability, an attacker will need access to the EMS machine running Ballot Now.

Impact: An attacker that can modify the stored procedure for user validation can allow any user with any credentials or none at all to log into Ballot Now. In a more sophisticated attack, the attacker would use the stored procedure to create specific backdoor accounts, which would be much harder to detect than the former attack.

Procedural Mitigations: Restricting physical access to the Hart EMS systems as much as possible reduces the opportunities for unauthorized users to connect to the EMS databases. Ensuring that the EMS systems are never connected to a network will also reduce such opportunities.

Verification: We connected to the Ballot Now security database and replaced the definition of the stored procedure with a single line of code that allowed any user to login to Ballot Now. After doing this, we could login to Ballot Now with any username and password or none at all.

20.7.2 Ballot Now hidden menu options are controlled by registry entries

Registry entries are used to determine which options are available to a Ballot Now user. These entries may be modified to enable special debugging functionality including Autovote, a test mode which can be used to generate pre-filled in paper ballots that can be scanned, counted and tallied from an election mode MBB, just like a regular paper ballot.

Description: Ballot Now uses parameters stored in the registry to enable hidden menu options. One of these options allows for the creation and printing of paper ballots which are pre-filled in. These ballots can then be used in an election.

Autovote can be used to create paper ballots as follows.

- Three registry entries are set to enable the creation of Autovote ballots.
- Ballot definitions are prepared in BOSS, and opened in Ballot Now.
- Once the ballot definitions are opened in Ballot Now, the Autovote menu item is available under the administrator menu.
- When this menu item is selected, the user is presented with a dialog box with the following GUI elements.
  - Setup: A menu item allowing the user to specify which precincts to be chosen for Autovote ballot generation.
  - Reports: A menu item allowing the user to generate reports showing the filled in ballots that will be generated.
  - Number of precincts selected: A field containing the number of precincts selected in the Setup menu item
  - Number of Generated Autovote Ballots: The number of unique ballots to be generated, equal to the number of candidates in the largest contest $\times$ the number of precincts selected
  - Number of Autovote Cycles: The number of copies of the unique set of ballots specified in the previous field
  - Total Number of Ballots to Autovote: The total number of ballots to be printed, equal to the product of the previous two fields.
- Once the number of Autovote cycles for the selected number of precincts is specified, the user may generate and print the pre-filled in ballots.

These ballots are identical to regular ballots in every way, except for the text "Autovote Ballot" in the left and right margins of the ballot. They are not rejected by the eScan, and are counted in the precinct report printed by the eScan. The CVRs for votes cast with Autovote ballots are stored on the election mode MBB placed in the eScan, and tallied by Tally like CVRs from regular ballots.

Prerequisites: In order to exploit this issue, an attacker will need physical access to the Ballot Now machine and know which registry entries to set.

Impact: An attacker able to print Autovote ballots can potentially stuff a ballot box with a large number of pre-filled in ballots. Because each Autovote ballot has a unique serial code the number of times an attacker may vote is limited by the number of generated Autovote ballots, and the number of registered voters in the precinct that have yet to vote.

While the Autovote ballots contain the text, "Autovote Ballot," it may be possible to remove this label. When the attacker prints the ballots in Ballot Now, the option to select a printer driver is presented. An attacker able to install a new printer driver, an action performable by the election administrator account on the EMS machines, can choose to print the ballots to a bitmap file, where the "Autovote Ballot" label could be removed by an image editing program. After this, the images can be printed with the serial number and bar codes intact, but without the Autovote label.

Mitigations:

Procedural Mitigations: Limiting physical access to the EMS machines will reduce the opportunities for an attacker to create Autovote ballots. Ensuring that the EMS systems are never connected to a network will

also reduce such opportunities. Instructing polling place workers to inspect ballots for the Autovote label before scanning would reduce the chances of Autovote ballots being cast in an election. Note that this will not prevent against the case above in which the label is removed using image editing software.

Verification: We used Ballot Now with the three registry entries set to enable the creation of Autovote ballots. We generated and printed both regular ballots and Autovote ballots. We then scanned both types of ballots with an eScan loaded with an election MBB. There was no distinction between the regular and Autovote ballots shown on the eScan paper printout. The votes from the regular and Autovote ballots were counted. The CVRs for both the regular and Autovote ballots were recorded to the election MBB and tallied by Tally. The votes cast with both types of ballots were used in the final tally, and no distinction was made between them.

CHAPTER 21

HART COMBINED IMPLICATIONS AND ATTACK SCENARIOS

21.1 Voting Machine Viruses

A virus is a malicious computer program with the ability to automatically move from machine to machine. Upon infection, such programs can change or delete the contents of files, record a user's operations or make a computer unusable. Because they can spread silently even in the presence of procedural safeguards, viruses represent a significant and realistic threat to the integrity of an election. By exploiting one of many of the issues presented in this report, an attacker could inject a virus into any JBC or eScan unit and gain control of an election in a county.

Infecting all of the voting machines in a county could occur as follows: An attacker gaining access to a JBC or an eScan voting unit could load malicious firmware onto these devices (Issues 19.7.3, 19.4.1, 19.7.5). When these devices are returned to the county headquarters after an election, the malicious firmware could then infect SERVO using any one of a number of vulnerabilities (19.3.1). Because all eSlates, JBCs and eScan voting machines in a county must interact with SERVO after an election for auditing reasons and before a new election to zero their counters, all such devices could be infected before the subsequent election occurred.

Our review of the Hart electronic machines indicates that such attacks are entirely plausible. This report has not only confirmed many of the previously known vulnerabilities enabling such attacks, but it has also uncovered a number of new serious new avenues through which viruses could infect voting machines.

21.2 Automated and Mass Voting in a Precinct

This scenario describes a way of generating an arbitrary number of automated votes on the eSlate and eScan voting machines.

We showed in Issue 20.1.4 that is possible to connect a JBC to a computing device, such as a laptop or Palm handheld, and have eSlate machines vote by themselves without any input from a voter. We also demonstrated in Issue 20.3.6 that by modifying a configuration file of the eScan, duplicate and photocopied may be cast without raising any alarms or warnings.

With these two attacks, it is possible to place an arbitrary number of votes for the candidate of choice into

the voting process. The auto-generated votes and duplicate ballots will be counted in the fashion as any legitimate votes on the voting machines. The only indication of malfeasance will be if the audit log on the eScan is checked and it is noticed that multiple ballots with the same serial number were scanned. In practice, it appears unlikely that the audit logs will be scrutinized to this extent. If such scrutiny of the electronic logs is expected, an attacker who has access to the back-end software running at county election headquarters, notably the PC running the SERVO application may attempt to access this machine if the audit logs have already been transferred after the election to SERVO, As we demonstrated in Issue 20.1.4, it is possible to erase entries from the audit log altogether. The only remaining evidence of ballot-box stuffing will come from the eSlate, and this will only be noticed if a manual recount is called for the state. Even in that case, detection of irregularities relies on a poll worker being able to compare multiple printed ballots and noting that the serial number is the same.

It may be noticed that the number of votes is inconsistent with the number of voters who were signed in. If an adversary gains access to the poll book and can add names on to the list, the numbers may be reconciled. If this is not possible, then the election may be viewed with suspicion. Scrupulous examination of the paper ballots may reveal the existence of duplicate votes. However, there will be no way to determine which of the votes cast on the eSlate are legitimate and which are not.

21.3 Forging Entire DRE Precinct Results

It is possible to forge the entire results from the eSlate DRE machines in a voting precinct with a small amount of effort and limited to no collusion amongst poll workers. At any point during an election, including after the polls have closed, the JBC and eSlates can have their memory completely cleared, including audit logs and cast vote records held within the internal memory of the JBC and eSlate devices, as well as those on the MBB within the eSlate. This attack is described in Issue 20.4.2, and only requires a malicious poll worker with about 30 seconds of seconds of access to the JBC and a computing device such as a Palm handheld. The MBBs for an election are fully forgeable, as described in Issue 19.1.7. In addition, because the rolls of paper can be removed from the VBO printer unit associated with each eSlate, it is not difficult to install a forged roll of paper in the machine with pre-generated election results, as it is merely a printout of results (see Issue 20.5.6). Because the rolls are meant to be sealed within the unit until it arrives at election headquarters, there is no place for election officials at the polling site to sign the roll; thus, if the roll is replaced, there will be little that will arouse suspicion. Thus, the entire electronic and paper record of the election may be forged, and only if the eSlate and JBC internal logs are examined will it be noticed that they do not contain audit or CVR information. However, because the MBBs and paper roll, representing the legal ballot of the election, exist, it is certainly possible that this may be chalked up to a technical glitch within the MBB and eSlate units rather than an attempt at malicious activity.

This full attack would require pre-generated paper rolls and pre-forged MBBs. It would also require a poll worker with the capability of plugging a small computing device into the JBC's parallel port interface to clear the memory of the electronic voting devices, a worker with approximately one minute of time to access the printer unit and replace the roll of paper with a forged copy, and a malicious worker, possibly the same person in all cases, to replace the MBB in the JBC unit with the forged MBB containing the desired results.

A variant of this attack is complete vote disenfranchisement with the electronic units. This would be possible by accessing the JBC to clear all internal information from it, the eSlate, and the MBB, and stealing the roll of paper outright from the VBO printer. This attack is described in Issue 20.4.2.

21.4 Vote Adjustments at Election Headquarters

The Tally application allows for the arbitrary adjustment of vote totals, as described in Issue 19.9.1. Because the audit logs from Tally may be easily and selectively deleted, as described in Issue 20.1.4, an administrator with access to Tally may adjust the votes from any precinct with impunity. The adjustment feature has been shown to work in practice with only one administrator needing access to Tally.[1]

The consequences of this attack are significant. If a worker at county headquarters has unimpeded access to Tally for less than a minute, they can arbitrarily change vote totals that will not arouse suspicion. Because of the copious numbers of entries from an audit log, unless they are closely scrutinized, the adjustment may never be found. Even if the audit logs are checked, a few more seconds with the machine running Tally will be sufficient to clear the adjustment totals from them. Only a full manual recount in the affected precincts will bring this issue to light, which in itself could potentially shake the public's confidence in the election results.

We now describe how this attack could be implemented by either an insider working at election headquarters in an administrative capacity, or by an outsider who has no administrative access to Tally but is able to get into the room where it is located and log into the running PC.

21.4.1 Insider Attack

An administrator implementing the vote adjustment attack requires less than a minute to modify precinct vote totals. If he or she has access to the username and password of Tally, plus an eCM token, using the vote adjustment feature takes seconds. With a few more seconds, the administrator can extract the password to the Tally database (Issue 19.1.2) and run an easily-found freeware application, perhaps from a USB stick, that allows logging into the database through the Windows ODBC interface and quickly eliminating all audit log entries to do with modifying the vote totals (Issue 20.1.4). It takes no more time to adjust the votes for 200 voters than it does for one.

21.4.2 Outsider Attack

This attack may be implemented by any sufficiently-prepared attacker who is able to enter the room where Tally is running. The only piece of information about the system necessary is the username and password of a Windows administrator on the account. Even in this case, there are many possible attacks against the underlying Windows operating system that could yield administrative access (Issue 19.1.5).

In order to access the administrative functions in Tally, access to an eCM token is required. A single key is used on eCM tokens throughout a county (Issue 19.2.1). It is possible to retrieve this key from voting equipment in a precinct without administrative access to the machines (Issue 19.1.7). Once an attacker has access to this secret key, it may be easily transferred to a commercially available Spyrus Rosetta token. The resulting token is completely indistinguishable from any other token used in the county.

As described in Issue 20.1.3, the password to access Tally is easily guessable. Hence, an attacker who can walk up to a machine running Tally may log in, present the token, and adjust the votes. They will have the same ability to adjust the audit logs to evade detection as an insider would, following the process described

[1] Alan Bernstein, 'Election fixes stir worries on ballot security'. Houston Chronicle, Nov. 14 2007 ⟨URL: `http://www.chron.com/disp/story.mpl/front/5299827.html`⟩.

above.

21.5 Attacks That Delay Elections

In many situations, preventing specific or all voters from participating in an election may be valuable to an attacker. For instance, the expense of attempting to run multiple elections may put significant financial strain on a county. Alternatively, by repeatedly "canceling" elections, an attacker may disenfranchise some portion of likely-voters. Vulnerabilities in every component of these systems allow such attacks to be successfully executed. We highlight a few of these in various devices.

The MBBs in the system can be compromised in a variety of ways. If they are placed in Tally before an election begins, they can be used for the entire election in a JBC or eSlate, but when they are brought back to Tally they will not report results (Issue 19.9.4). Other attacks can target the voting equipment within a precinct itself; for example, a marker can be used to cause an eScan to overvote every ballot or silently block a voter from voting for a certain candidate (Issue 20.3.3. Similarly, with low effort it is possible to attack the VBO units attached to eSlates in ways that can put the eSlate out of service for a considerable amount of time (Issues 20.5.2,20.5.3).

21.6 Denial of service or destruction of election results in a polling place

A group of voters and possibly poll workers can cooperate to disable election equipment in a precinct or destroy the results of an election, effectively denying legitimate voters the ability to vote or have their votes counted. This can be accomplished through the physical destruction of paper ballots and voting equipment, as well as the electronic deletion of all cast votes and audit logs in the polling place.

By themselves, a group of voters could disable the VVPAT capabilities provided by DREs (20.5.2), destroy the ballots scanned by the eScan (20.3.5), or prevent ballots from being recorded by the eScan (20.3.3). With the cooperation of a poll worker it is possible to delete all cast vote records and audit log entries from the eSlate, JBC and MBBs (20.4.2), as well as the eScan (20.3.7).

Part V

Appendices

CHAPTER 22

ES&S PRIVATE REPORT

22.7.1 Unity

22.7.1.1 Unity ERM Buffer Overflow when Reading a Results PEB

This section has been redacted from the public report. The missing content relates to the issue discussed in section 7.1.1.

22.7.1.2 Data from M100 can cause a buffer overflow in Unity

This section has been redacted from the public report. The missing content relates to the issue discussed in section 7.1.2.

22.7.1.5 Processing audit data can cause buffer overflow of a global variable

This section has been redacted from the public report. The missing content relates to the issue discussed in section 7.1.5.

22.7.1.6 Unity decrypts a PEB using the EQC from the PEB

This section has been redacted from the public report. The missing content relates to the issue discussed in section 7.1.6.

22.7.1.7 Unity contains large pieces of duplicated code

This section has been redacted from the public report. The missing content relates to the issue discussed in section 7.1.7.

22.7.1.8 Unity contains many small buffer overflows

This section has been redacted from the public report. The missing content relates to the issue discussed in section 7.1.8.

22.7.2 iVotronic

22.7.2.5 Buffer overflow in poll opening process is exploitable

This section has been redacted from the public report. The missing content relates to the issue discussed in section 7.2.5.

22.7.2.6 Buffer overflow in "Hotspot" image processing is exploitable

This section has been redacted from the public report. The missing content relates to the issue discussed in section 7.2.6.

22.7.2.7 Buffer overflow in Supervisor iVotronic initialization process is exploitable

This section has been redacted from the public report. The missing content relates to the issue discussed in section 7.2.7.

22.7.2.10 (Emulated) Factory QA PEBs bypass all password checks

This section has been redacted from the public report. The missing content relates to the issue discussed in section 7.2.10.

22.7.2.15 Audit data is insufficiently randomized

This section has been redacted from the public report. The missing content relates to the issue discussed in section 7.2.15.

22.7.2.16 VVPAT barcodes contain timestamps

This section has been redacted from the public report. The missing content relates to the issue discussed in section 7.2.16.

22.7.2.17 VVPAT barcodes contain vote information

This section has been redacted from the public report. The missing content relates to the issue discussed in section 7.2.17.

22.7.2.18 Accuracy testing mode detectable

This section has been redacted from the public report. The missing content relates to the issue discussed in section 7.2.18.

22.7.4 M650

22.7.4.1 M650 runs executable on Zip Disk on power up

This section has been redacted from the public report. The missing content relates to the issue discussed in section 7.4.1.

22.7.4.3 M650 fails to validate variable-length strings

This section has been redacted from the public report. The missing content relates to the issue discussed in section 7.4.3.

22.7.4.4 M650 fails to protect against integer overflows

This section has been redacted from the public report. The missing content relates to the issue discussed in section 7.4.4.

22.9.1 Tools

22.9.1.1 A framework for delivering a malicious payload to iVotronic systems

This section has been redacted from the public report. The missing content relates to the issue discussed in section 9.1.1.

22.9.1.2 Pebserial: a tool for reading and writing PEBs

This section has been redacted from the public report. The missing content relates to the issue discussed in section 9.1.2.

22.9.1.3 The iVotronic Serial Debugger

This section has been redacted from the public report. The missing content relates to the issue discussed in section 9.1.3.

22.9.1.4 A tool to read and write M100 PCMCIA memory cards

This section has been redacted from the public report. The missing content relates to the issue discussed in section 9.1.4.

22.9.1.5 The JTAG hardware debugger

This section has been redacted from the public report. The missing content relates to the issue discussed in section 9.1.5.

22.9.1.6 A tool for extracting the QNX filesystem from the M100

This section has been redacted from the public report. The missing content relates to the issue discussed in section 9.1.6.

22.9.3 Successful Attack Scenarios

22.9.3.1 Attack Scenario peb.1: Changing an Unattentive Voter's Vote

This section has been redacted from the public report. The missing content relates to the issue discussed in section 9.3.1.

22.9.3.2 Attack Scenario peb.2: Changing a Careful Voter's Vote

This section has been redacted from the public report. The missing content relates to the issue discussed in section 9.3.2.

22.9.3.3 Attack Scenario peb.3: Canceling the Vote of a Fleeing Voter

This section has been redacted from the public report. The missing content relates to the issue discussed in section 9.3.3.

22.9.3.4 Attack Scenario peb.4: Canceling a Vote by Faking a Fleeing Voter

This section has been redacted from the public report. The missing content relates to the issue discussed in section 9.3.4.

22.9.3.5 Attack Scenario flash.1: iVotronic Denial-of-Service

This section has been redacted from the public report. The missing content relates to the issue discussed in section 9.3.5.

22.9.3.6 Attack Scenario flash.2: Voter Confusion

This section has been redacted from the public report. The missing content relates to the issue discussed in section 9.3.6.

22.9.3.7 Attack Scenario unity.1: Unrestricted Access to Unity Ballot Preparation Software

This section has been redacted from the public report. The missing content relates to the issue discussed in section 9.3.7.

22.9.3.8 Attack Scenario unity.2: Compromising the Unity Election Results Manager

This section has been redacted from the public report. The missing content relates to the issue discussed in section 9.3.8.

22.9.3.9 Attack Scenario m100.1: Changing the Firmware on the M100 Scanner

This section has been redacted from the public report. The missing content relates to the issue discussed in section 9.3.9.

22.9.3.10 Attack Scenario m100.2: M100 Denial-of-Service

This section has been redacted from the public report. The missing content relates to the issue discussed in section 9.3.10.

22.9.3.11 Attack Scenario virus.1: Compromising Entire Election Process with a Virus

This section has been redacted from the public report. The missing content relates to the issue discussed in section 9.3.11.

CHAPTER 23

PREMIER EQUIPMENT

Figure 23.1: The Premier AV-OS Central Count

Figure 23.2: The Premier AV-OS Precinct Count

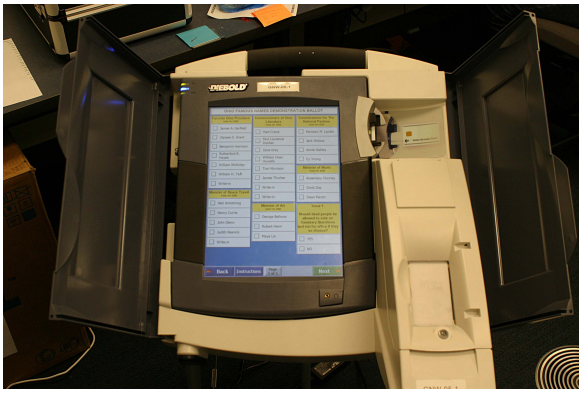

Figure 23.3: The Premier AV-TSX

Figure 23.4: The Digi PortServer II

Figure 23.5: The Premier EMP Server

Figure 23.6: The Premier ExpressPoll

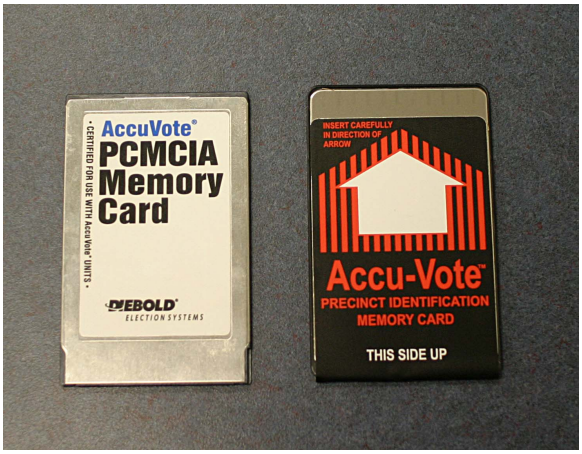

Figure 23.7: The Premier PCMCIA (AV-TSX) and Epson (AV-OS) memory cards

Figure 23.8: The Premier Voter Card Encoder and Smart Cards

CHAPTER 24

PREMIER PRIVATE REPORT - ISSUE CONFIRMATION

24.1 Premier GEMS Vulnerabilities

24.1.1 GEMS uses the Microsoft Jet data layer

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.1.1.

24.1.2 GEMS databases can be modified with access to the local hard disk

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.1.2.

24.1.3 GEMS relies on the graphical user interface (GUI) to enforce security

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.1.3.

24.1.4 Third parties translation services may introduce a virus into the system

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.1.4.

24.1.5 Race and candidate labels may be changed after GEMS has been "set-for-election"

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.1.5.

24.1.6 GEMS fails to filter login field for special characters

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.1.6.

24.1.7 Unsafe handling of election database content may allow a virus to control GEMS

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.1.7.

24.1.8 Election database passwords are stored with insufficient protection

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.1.8.

24.1.9 Poor handling of integers may cause GEMS to crash

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.1.9.

24.2 Premier AV-OS PC Vulnerabilities

24.2.1 The AV-OS memory card data is not authenticated

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.2.1.

24.2.2 The GEMS server and AV-OS PC communication is not authenticated

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.2.2.

24.2.3 The memory card checksums do not protect against malicious tampering

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.2.3.

24.2.4 The audit log is unprotected and old entries are overwritten

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.2.4.

24.2.5 The memory card "signature" does not prevent malicious tampering

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.2.5.

24.2.6 Improper string handling can result in arbitrary code execution

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.2.6.

24.2.7 Candidate vote counters are not checked for overflow

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.2.7.

24.2.8 The supervisor "password" is stored with inadequate protection

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.2.8.

24.2.9 Candidate ballot coordinates can be modified to manipulate election results

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.2.9.

24.2.10 Flaws in the AccuBasic interpreter may allow a virus to control an AV-OS PC

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.2.10.

24.2.11 Memory card attacks could be masked by modifying the zero report AccuBasic script

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.2.11.

24.2.12 The AV-OS PC software controls the physical paper ballot box deflector

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.2.12.

24.2.13 A voter can cause the AV-OS PC to stop accepting ballots

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.2.13.

24.2.14 Memory card contents can be accessed from the serial port

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.2.14.

24.2.15 The AV-OS PC accepts the same ballot multiple times

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.2.15.

24.2.16 The AV-OS PC printer can be temporarily disabled without the supervisor password

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.2.16.

24.3 Premier AV-TSX Vulnerabilities

24.3.1 The AV-TSX will install an unauthenticated bootloader and operating system

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.1.

24.3.2 The AV-TSX will install unauthenticated application updates

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.2.

24.3.3 There are multiple buffer overflow vulnerabilities in the handling of .ins files

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.3.

24.3.4 An unauthenticated user can read and or tamper with the memory of the AV-TSX

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.4.

24.3.5 The cryptographic keys are not adequately protected

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.5.

24.3.6 Malicious software could alter files critical to correctly reporting an election

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.6.

24.3.7 The smart card authentication protocol can easily be broken

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.7.

24.3.8 Security Cards can be forged and used to change keys

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.8.

24.3.9 The System Setup Menu is accessible without using a smart card

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.9.

24.3.10 The protected counter is not protected

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.10.

24.3.11 SSL certificates are not sufficiently protected

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.11.

24.3.12 OpenSSL is not initialized with adequate entropy

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.12.

24.3.13 Flaws in the AccuBasic interpreter may allow a virus to control an AV-TSX

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.13.

24.3.14 Failure to check data on memory cards may allow a virus to run on the AV-TSX

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.14.

24.3.15 Unsafe handling of election resource files may allow a virus to control an AV-TSX

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.15.

24.3.16 Unsafe handling of election database files may allow a virus to control an AV-TSX

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.16.

24.3.17 A voter may be able to gain control of an AV-TSX

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.17.

24.3.18 A malicious GEMS server can cause an AV-TSX to crash on download

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.18.

24.3.19 Ballots are stored in the order in which they are cast

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.19.

24.3.20 Cast ballots and VVPAT barcodes contain a timestamp

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.20.

24.3.21 An attacker can reconstruct the order in which ballots were cast

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.21.

24.3.22 The AV-TSX does not securely delete files

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.22.

24.3.23 Logic errors in the bootloader create a vulnerability when loading bitmaps

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.23.

24.3.24 There are a number of errors in the AV-TSX startup code

This section has been redacted from the public report. The missing content relates to the issue discussed in section 13.3.24.

CHAPTER 25

PREMIER PRIVATE REPORT - NEW ISSUES

25.1 Premier EMP Vulnerabilities

25.1.1 Tampering with a memory card allows attackers to crash or take control of an EMP server.

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.1.1.

25.1.2 A single Data Key is used to encrypt all ballots in a county

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.1.2.

25.1.3 The warning that a default key is in use is not sufficient

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.1.3.

25.1.4 A malformed IP address can freeze the EMP server

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.1.4.

25.1.5 The contents of the logs on the EMP server are not authenticated

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.1.5.

25.1.6 The EMP server shares the same default SSL Certificate as the AV-TSX

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.1.6.

25.1.7 The System Key for the EMP is insecure

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.1.7.

25.1.8 The integrity of the Data Key is not protected

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.1.8.

25.1.9 GEMS trusts the EMP to deliver correct ballot definitions and results

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.1.9.

25.1.10 A malicious GEMS server can crash an EMP server

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.1.10.

25.1.11 A compromised AV-TSX can crash an EMP server

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.1.11.

25.2 Premier General Vulnerabilities

25.2.1 Premier may follow insecure media format procedures

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.2.1.

25.3 Premier GEMS Vulnerabilities

25.3.1 Vulnerabilities in Microsoft Windows provided DLL files affect GEMS

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.3.1.

25.3.2 Many GEMS servers state-wide use the same BIOS password

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.3.2.

25.4 Premier AV-OS PC Vulnerabilities

25.4.1 Forged ballots can be forced into the ballot box

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.4.1.

25.4.2 The AV-OS PC ballot box collecting votes allows vote order to be reconstructed

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.4.2.

25.4.3 The Hart ballot box key works in the Premier ballot box

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.1.

25.5 Premier VCEncoder Vulnerabilities

25.5.1 Access to the Voter Card Encoder is not protected by a PIN

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.5.1.

25.5.2 Once the VCE is enabled, no mechanism prevents smart cards from being encoded

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.5.2.

25.5.3 Software can be loaded by anyone with access to the VCE

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.2.

25.5.4 The VCE accepts the default smart card key

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.5.4.

25.6 Premier ExpressPoll Vulnerabilities

25.6.1 The ExpressPoll runs a webserver

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.6.1.

25.6.2 The ExpressPoll will install an unauthenticated bootloader and operating system

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.6.2.

25.6.3 The ExpressPoll uses an unprotected database for poll data

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.6.3.

25.6.4 The ExpressPoll uses an unprotected resource file

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.6.4.

25.6.5 The ExpressPoll can be rendered useless in transit

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.6.5.

25.6.6 The ExpressPoll audit logs are not protected

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.6.6.

25.6.7 ExpressPoll audit logs violate voter privacy

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.6.7.

25.7 Premier Digital Guardian Vulnerabilities

25.7.1 Users with administrative access can circumvent boot restrictions

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.7.1.

25.7.2 The *GEMSUser* account is in the *Administrators* group

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.7.2.

25.7.3 Digital Guardian can be disabled by booting from external media

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.7.3.

25.7.4 An administrative user can disable the Digital Guardian device drivers once

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.7.4.

25.7.5 Users with administrative access can circumvent many Digital Guardian controls

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.7.5.

25.7.6 The Digital Guardian policy denies only specific known unwanted applications

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.7.6.

25.7.7 Digital Guardian execution restrictions are circumventable by copying files

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.7.7.

25.7.8 *GEMSUser* can use the CD burning application to modify the election database

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.7.8.

25.7.9 The election database protections only apply to files on the local hard drive

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.7.9.

25.7.10 Digital Guardian logging is disabled

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.7.10.

25.7.11 Digital Guardian does not immediately detect if GEMS is replaced

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.7.11.

25.8 Premier AV-TSX Vulnerabilities

25.8.1 The wires connecting the VVPAT printer can be cut without opening the AV-TSX

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.8.1.

25.8.2 The plastic housing protecting the printer can easily be removed

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.8.2.

25.8.3 An adversary can destroy paper copies of cast ballots without opening the AV-TSX

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.3.

25.8.4 The power button is accessible when all panels on the AV-TSX are closed and locked

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.8.4.

25.8.5 The bootloader can be manipulated to give access to the file system on the AV-TSX

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.8.5.

25.8.6 The memory of an AV-TSX can be erased by a memory card

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.8.6.

25.8.7 A voter can gain administrative access to the AV-TSX

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.8.7.

25.8.8 A voter can cast an unlimited number of votes without any tools or knowledge

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.8.8.

25.8.9 The smart card reader can be jammed by an attacker

This section has been redacted from the public report. The missing content relates to the issue discussed in section 14.8.9.

CHAPTER 26

HART EQUIPMENT

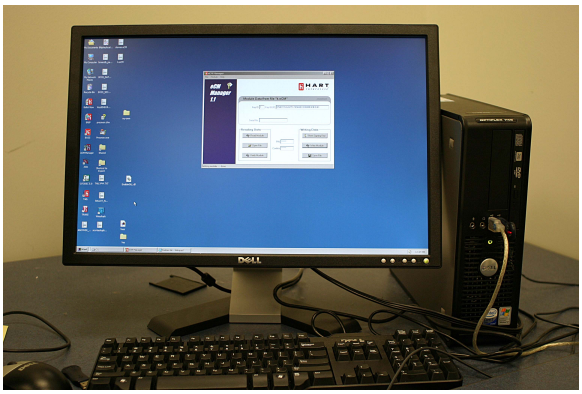

Figure 26.1: The Hart Election Management Server

Figure 26.2: The Hart eScan Voting terminal

Figure 26.3: The Hart eSlate DRE Voting terminal

Figure 26.4: The Hart Judge's Booth Controller (JBC)

Figure 26.5: The Hart SERVO terminal

CHAPTER 27

HART PRIVATE REPORT - ISSUE CONFIRMATION

27.1 Hart General Vulnerabilities

27.1.1 Corrupt MBBs can cause Tally to crash

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.1.1.

27.1.2 Database passwords are stored insecurely

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.1.2.

27.1.3 The EMS databases are not encrypted

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.1.3.

27.1.4 New Users can be inserted into the database

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.1.4.

27.1.5 Back-end Windows systems may be insecure

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.1.5.

27.1.6 eCM keys are extracted and stored insecurely

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.1.7.

27.1.7 Vote order can be determined from an MBB

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.1.8.

27.1.8 Voting and audit data not secured while voting

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.1.9.

27.1.9 The internal vote count on the eScan, eSlate, and JBC can be modified

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.1.10.

27.1.10 The EMS software uses a vulnerable version of OpenSSL

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.1.11.

27.1.11 Pervasive failure to follow safe programming practices

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.1.12.

27.2 Hart eCM Vulnerabilities

27.2.1 eCM keys are stored insecurely on the eCM manager

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.2.2.

27.3 Hart SERVO Vulnerabilities

27.3.1 Buffer overflows in SERVO

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.3.1.

27.4 Hart eScan Vulnerabilities

27.4.1 The eScan is managed via an accessible Ethernet port

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.4.1.

27.5 Hart eSlate Vulnerabilities

27.5.1 The eSlate is managed via a serial port connection to the JBC

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.5.1.

27.5.2 Communication between the eSlate and JBC is insecure

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.5.2.

27.5.3 Compromised eSlates can provide votes without voter records

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.5.3.

27.5.4 The periodic memory integrity checks used by eSlate and JBC are ineffective

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.5.4.

27.6 Hart Servo Vulnerabilities

27.6.1 SERVO-based device firmware checking can be spoofed

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.6.1.

27.7 Hart JBC Vulnerabilities

27.7.1 The JBC internal version checks can never fail.

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.7.1.

27.7.2 JBC-based eSlate firmware checking can be spoofed

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.7.2.

27.7.3 The JBC is managed via an accessible parallel port

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.7.3.

27.7.4 The JBC Voter Registration Interface can be used to generate voter access codes

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.7.4.

27.7.5 Format String vulnerabilities in the JBC report mode

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.7.5.

27.7.6 Voter codes are not selected randomly

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.7.6.

27.8 Hart VBO Vulnerabilities

27.8.1 The VBO record is easily manipulatable by an eSlate program

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.8.1.

27.8.2 VBO printer allows the paper to be rewound

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.8.2.

27.8.3 VBO ballots are printed sequentially

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.8.3.

27.9 Hart Tally Vulnerabilities

27.9.1 The Tally interface allows a Tally administrator to "adjust vote totals"

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.9.1.

27.9.2 Precinct IDs are improperly handled by the system

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.9.2.

27.9.3 Users can tally unclosed or corrupted MBBs

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.9.3.

27.9.4 MBBs are only processed if the Tally database allows it

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.9.4.

27.9.5 Rally and Tally allow a user to accept previously unrecognized certificates

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.9.5.

27.10 Hart Ballot Now Vulnerabilities

27.10.1 Ballot Now ballot counters are stored in a database

This section has been redacted from the public report. The missing content relates to the issue discussed in section 19.10.1.

CHAPTER 28

HART PRIVATE REPORT - NEW ISSUES

28.1 Hart General Vulnerabilities

28.1.1 An MBB image can be copied and restored without any credentials.

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.1.1.

28.1.2 EMS systems make improper use of the Windows registry

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.1.2.

28.1.3 Hart EMS passwords can be bypassed

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.1.3.

28.1.4 Hart EMS audit logs can be modified or erased

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.1.4.

28.2 Hart eCM Vulnerabilities

28.2.1 eCM keys may be quietly recorded to a debug file

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.2.1.

28.3 Hart eScan Vulnerabilities

28.3.1 The flash memory containing the eScan executable and file system can be replaced

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.3.1.

28.3.2 The eScan runs a telnet server

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.3.2.

28.3.3 The eScan scanner surface can be occluded to affect ballot processing

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.3.3.

28.3.4 The eScan ballot box collecting votes allows vote order to be reconstructed

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.3.4.

28.3.5 The eScan ballot box is vulnerable to attacks that may destroy ballots

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.3.5.

28.3.6 The eScan may be modified to allow casting of duplicate ballots

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.3.6.

28.3.7 The eScan has an open interface allowing erasure of vote and audit log records

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.3.7.

28.3.8 The Premier ballot box key works in the Hart ballot box

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.3.8.

28.3.9 New Vulnerability 09

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.3.9.

28.4 Hart JBC Vulnerabilities

28.4.1 The JBC can rapidly create an unlimited number of access codes on election day

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.4.1.

28.4.2 The JBC has an open interface allowing erasure of vote and audit log records

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.4.2.

28.4.3 JBC/eSlate voting can be completely automated

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.4.3.

28.5 Hart VBO Vulnerabilities

28.5.1 The VBO Printer is controlled via an accessible 1/8" port

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.5.1.

28.5.2 The serial number of the VBO can be modified

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.5.5.

28.5.3 The VVPAT paper record may be forged

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.5.6.

28.6 Hart Tally Vulnerabilities

28.6.1 Tallied MBBs can be used in election

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.6.1.

28.6.2 The Tally user interface is completely configured in the registry

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.6.2.

28.7 Hart Ballot Now Vulnerabilities

28.7.1 The Ballot Now username and password can be bypassed

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.7.1.

28.7.2 Ballot Now hidden menu options are controlled by registry entries

This section has been redacted from the public report. The missing content relates to the issue discussed in section 20.7.2.

CHAPTER 29

PROVIDED SOURCE CODE AND EQUIPMENT

All source code lines of code generated using David A. Wheeler's 'SLOCCount'[1] or the Linux *wc* utility.

Table 29.1: Provided Source Code for ES&S

Component	**Languages**	**Lines of Code**
Unity 3.0.1.1	BAT Scripts, C, C++, Cobol, Java, VisualBasic	364,136
AIMS 1.2	C++, SQL, VisualBasic	59,984
iVotronic firmware 9.1.6.4	C, ASM	87,157
iVotronic PIC	C, PIC ASM	1,753
M100 firmware 5.2.1.0	C, ASM, shell script	29,952
M650 firmware 2.1.0.0	C, C++, shell script	22,458
VAT (AutoMARK)	C, C++, C#, ASM, VisualBasic	104,305

[1] David A. Wheeler, *SLOCCount*. ⟨URL: `http://www.dwheeler.com/sloccount/`⟩.

Table 29.2: Provided Equipment for ES&S

Component
Unity Back-End Computer (Dell Optiplex GX745 Core2 Duo) with: • Unity 3.0.1.1 • Hardware Program Manager 5.2.4.0 • Election Deffinition Manager 7.4.4.0 • Election Results Manager 7.1.2.1 • Audit Manager 7.3.0.0 • ESS Image Manager 7.4.2.0 • iVotronic Image Manager 2.0.1.0 • Microsoft Windows XP Pro SP2 (COTS)
iVotronic DRE • ADA enabled iVotronic DRE • iVotronic DRE • Firmware versions: 9.1.6.2, 9.1.6.4 • Supervisor PEBs • Voter PEBs • CF Cards
M100 • M100 optical scan unit • Firmware Version 5.2.1.0 • PCMCIA cards • Ballot Box • Scanner Key • Ballot Box Key • Omni Drive USB PCMCIA reader/writer (COTS)
M650 • M650 batch optical scan unit • Zip Disk (COTS) • Iomega Zip Drive (COTS) • Microline 520 Dot Matrix Printers x2 (COTS)
AutoMARK • VAT AutoMARK • Firmware Version 1.1

Table 29.3: Provided Source Code for Premier

Component	Languages	Lines of Code
GEMS 1.18.24	C++	115,637
KeyCardTool 4.6.1	C++	1,153
VCProgrammer 4.6.1	C++	1,742
BallotStation 4.6.4	C++	64,889
WildCat BootLoader 1.2.1	C, C++, ASM	72,910
EMP 4.6.2	C++	31,273
ExpressPoll CardWriter 1.1.5[2]	C#	139
PCMCard 1.1.5[3]	C++	1,494
VCE 1.3.2	C	879
AV-OS CC 2.0.12	C, ASM	24,441
AV-OS PC 1.96.6	C, ASM	20,195

Table 29.4: Provided Equipment for Premier

Component
GEMS Server (Dell PowerEdge 2900) with: • GEMS 1.18.24 • KeyCardTool 4.6.1 • Microsoft Windows 2000 Server SP4 (COTS) • Verdasys Digital Guardian Agent (COTS) • Sygate Security Agent 4.1 (COTS) • McAfee VirusScan Enterprise 8.0.0 (COTS) • Secure-Tech ST-100 smart card reader (COTS)
AccuVote-TSx running: • BallotStation 4.6.4 • Wildcat Bootloader 1.2.1
AccuVote-OS Precinct Count 1.96.6 (with ballot box)
AccuVote-OS Central Count 2.0.12
Voter Card Encoder (VCE) 1.3.2
Digi PortServer II 16 (COTS)
ExpressPoll 5000 2.0.29
Election Media Processor (EMP) (Dell Dimension 3100) with: • EMP 4.6.2 • External 6-bay PCMCIA chassis • Microsoft Windows XP (COTS)
Digital Guardian Console (IBM Thinkpad T43) • Verdasys Digital Guardian Console (COTS) • Microsoft Windows 2000 Server SP4 (COTS)

Table 29.5: Provided Source Code for Hart

Component	Languages	Lines of Code
Hartlib Sys 6.2-4.0	C++, ASM	56,030
Ballot Now 3.3.11	C++	64,212
bossutil 2.5.8 (Boss 4.3.13)	C++	31,116
Translate DLL 1.8.2 (Boss 4.3.13)	C++	3,900
Tally 4.3.10	C++	51,582
Rally 2.3.7	C++	7,026
SERVO 4.2.10	C++	23,897
eCM Manager 1.1.7	C++	1,945
eScan 1.3.14	C++	76,482
JBC 4.3.1	C++	23,104
eSlate (epc) 4.2.13	C++	19,249
VBO 1.8.3	C	1,686

Table 29.6: Provided Equipment for Hart

Component
EMS Server (Dell Optiplex 745) with: • Ballot Now 3.3.11 • BOSS 4.3.13 • Tally 4.3.10 • Microsoft Windows 2000 Server SP4 (COTS) • Lexar Media Card Reader/Writer Model GS-UFD20SATP (PCMCIA) (COTS)
SERVO Laptop (Dell Latitude D520) with: • SERVO 4.2.10 • Microsoft Windows 2000 Server SP4 (COTS) • Quatech SPP-100 (PCCard to Parallel) (COTS)
Judges Booth Controller (JBC) 4.3.1
eSlate 1.3.14
eScan 4.2.14 (with ballot box)