Source Code Review of the Hart InterCivic Voting System[1]

Srinivas Inguva
Stanford University[2]

Eric Rescorla
Network Resonance

Hovav Shacham
University of California, San Diego

Dan S. Wallach
Rice University

July 20, 2007

[1]This report was prepared at the University of California, Berkeley under contract to the California Secretary of State as part of a "Top-to-Bottom" review of electronic voting systems certified for use in the State of California.

[2]All author affiliations are for identification only.

Executive Summary

This report considers security issues in Hart InterCivic's voting suite, version 6.2.1. This report was prepared at the request of the California Secretary of State, as part of a "top-to-bottom" review of the state's electronic voting systems. This document is the final report of the team that examined the Hart voting system source code.

Hart's system consists of back-office election management components (SERVO, Rally, Tally, eCM Manager, BOSS, Ballot Now) which are used to configure and collect data from precinct devices (eScan, eSlate, Judge's Booth Controller). The election management software runs on ordinary Windows machines whereas the precinct devices are embedded programs running on specialized hardware.

Component-to-component networks are pervasive in Hart's architecture. A JBC and one or more eSlates are networked together at polling place for voting. JBCs, eSlates, and eScans are networked with SERVO for pre- and post-election setup, auditing, and reset. Rally and Tally communicate over a modem (or leased line) to transmit remote voting records. In addition, the other components communicate indirectly through PCMCIA memory cards called "Mobile Ballot Boxes" (MBBs). Building a secure networked system of this type requires adopting an attitude of *defense in depth*: it must be designed and implemented in such a way that a compromised component cannot induce misbehavior in other components that communicate with it.

Our examination indicates that Hart's system is not designed along these lines. Instead, the design of the components mostly assumes that any other component of the Hart system (or anything that appears to be one) is trustable:

Unsecured network interfaces Network interfaces in the Hart system are not secured against direct attack. Voters can connect to unsecured network links in a polling place to subvert eSlates, as well as to eavesdrop on cast votes and to inject new votes. Poll workers can connect to JBCs or eScans over the management interfaces and perform back-office functions such as modifying the device software. The impact of this is that a malicious voter could potentially take over one or more eSlates in a precinct and a malicious poll worker could potentially take over all the devices in a precinct. The subverted machines could then be used to produce any results of the attacker's choice, regardless of voter input. We emphasize that these are not bugs in the Hart software, but rather features intentionally designed into the system which can be used in a fashion for which they were never intended.

Vulnerability to malicious inputs Because networked devices may be connected to other, potentially malicious devices, they must be prepared to accept robustly any input provided by such devices. The Hart software routinely fails to check the correctness of inputs from other components, and then proceeds to use those inputs in unsafe ways. The most damaging example of this is that SERVO, which is used to back up and verify the correctness of polling place devices can itself be compromised from those same devices. This implies that an attacker could subvert a single polling place device, through it subvert SERVO, and then use SERVO to reprogram every polling place device in the county. Although we have tested some individual components of this attack, we did not have time to confirm it in an end-to-end test.

No or insecure use of cryptography The standard method for securing network communication of the type in use in the Hart system is to use a cryptographic security protocol. However, we

found a notable lack of such techniques in Hart's system. Instead, communications between devices generally happen in the clear, making attack far easier.

Cryptography is used for MBBs, but the key management involves a single county-wide symmetric key that, if revealed, would allow an attacker to forge ballot information and election results. This key is stored insecurely in vulnerable polling-place devices, with the result that compromise of a single polling place device enables an attacker to forge election MBBs carrying election results for any device in the county.

Failure to protect ballot secrecy Hart's system fails to adequately protect ballot secrecy. A poll worker or election official with access to the raw ballot records can reconstruct the order in which those votes were cast. Combined with information about the order in which voters cast their votes, this can be used to reconstruct how each voter voted. In the case of the DRE, it is also possible to reconstruct, for each vote, the order in which the votes were authorized. Combined with information about the order in which voters were authorized, this can likewise be used to reconstruct how each voter voted. Furthermore, a voter who has temporary access to an eSlate device can extract and reconstruct all the votes cast on that device up to that point in time. He may be able to similarly reconstruct all votes cast on any other eSlate connected to the same JBC.

Many of these attacks can be mounted in a manner that makes them extremely hard to detect and correct. We expect that many of them could be carried out in the field by a single individual, without extensive effort, and without long-term access to the equipment.

A manual examination of the paper trail would act as a defense against some of our attacks; others may be mitigated by new or existing procedural controls by election officials, or by changes to the Hart system. Where reasonable, we attempted to identify such mitigations and to assess their effectiveness. In some cases, there may be no simple, effective fixes.

We have deliberately avoided addressing the broader issue of whether or how this system should be used for voting in California. Making that judgement requires assessing not only the technical issues described in this report but also the procedures and policies with which the system is used.

Table of Contents

List of Issues

CHAPTER 1

Introduction

This report was prepared by the University of California, Berkeley at the request of the California Secretary of State, as part of a "top-to-bottom" review of the state's electronic voting systems. This document is the final report of the team that examined the Hart voting system source code.

The Hart system source code review team was located at SRI International in Menlo Park and consisted of the four authors of this report: Srinivas Inguva, Eric Rescorla, Hovav Shacham, and Dan Wallach. We frequently consulted with the UC Berkeley-based documentation team[1] and the Sacramento-based "Red Team"[2]. All opinions expressed in this report, however, are solely those of its authors.

We started work on June 15, 2007 and received the Hart system source code on June 18, 2007. Work ended on July 20, 2007 with the delivery of this report.

1.1 System Overview

The Hart software we reviewed is part of a comprehensive system that includes Direct Recording Electronic (DRE) voting machines and optical scan ballot collection equipment for use at polling places, as well as election definition, management and counting software and hardware for use at a county elections headquarters. The specific system components certified for use in California for which we reviewed source code were from Hart system version 6.2.1, comprising the following components:

- Ballot Now, version 3.3.11
- BOSS, version 4.3.13
  - BossUtil, version 2.5.8
  - Translate DLL, version 1.8.2
- eCM Manager, version 1.1.7
- eScan, version 1.3.14
- eSlate, version 4.2.13
- HartLib, version 4.0
- JBC, version 4.3.1
- Rally, version 2.3.7
- SERVO, version 4.2.10

[1] Joseph Lorenzo Hall and Laura Quilter

[2] Robert Abbott, Mark Davis, Joseph Edmonds, Luke Florer, Brian Porter, Elliot Proebstel, Sujeet Shenoi, and Jacob Stauffer

- Tally, version 4.3.10
- VBO Printer Firmware, version 1.8.3

The centralized back-end processing functions (ballot preparation, voting machine configuration, and post-election vote counting) are performed by BOSS, BallotNow, Tally, and SERVO, which are all software running on Windows-based PCs. Cryptographic keys are distributed on eSlate Cryptographic Modules (eCMs) which are managed using the eCM Manager, which also runs on Windows.

Precinct polling stations can be equipped with DRE terminals, optical scan ballot readers or both. Normally, precinct-based optical ballot scanning is performed with Hart's eScan systems while central optical ballot scanning is performed with Hart's Ballot Now.[3] DRE voting uses a network of eSlates controlled by a single JBC. Results are returned to Election Central using Mobile Ballot Boxes (MBBs), which are standard PCMCIA memory cards. Election results can also be returned using the Rally software, which transmits them to Tally via modem connections from regional processing facilities.

The software comprises approximately 360K lines of source code, written primarily in C++, C, and PowerBuilder. The back-end software runs on Microsoft Windows. The eScan runs on Windows CE on custom embedded hardware. The eSlates and JBCs run on an embedded operating system on custom embedded hardware. (We were not provided with source code to any of these operating systems.)

A detailed description of the system architecture and operation can be found in Section 4.

1.2 Methodology

Discovery of programming errors is a notoriously difficult problem in computer science, and no general methodology exists that is guaranteed to find all problems in even very small programs. The large size and complexity of the Hart InterCivic system makes a complete review an especially daunting task under even the best conditions, but particularly so here given the limited time available and the other constraints imposed on us by the terms of the review.

Our focus was on whether the software contains effective safeguards against error and abuse aimed at altering election results, changing votes, denying service, altering audit logs, and compromising voters' ballot secrecy. More broadly, we explored issues related to our confidence in the security and reliability of the architecture and implementation as a whole. In general, in our review we attempted to explore questions of architectural soundness:

- Does the design and implementation follow sound, generally accepted engineering practices? Is code defensively written against bad data, errors in other modules, changes in environment, and so on?
- Is the cryptography and key management sound? Is cryptography correctly used to protect sensitive data on untrusted media? Does the cryptography employ standard algorithms and protocols? Are keys managed according to good practices?
- What are the trusted components of the system, when are they trusted and for what purposes? What parties are trusted and for what purposes? What are the implications of a compromise of trusted components?
- Are security failures likely to be detected? Are audit mechanisms reliable and tamper-resistant? Are data that might be subject to tampering properly validated and authenticated?

Our review of the source code was focused on answering these questions, rather than examining the entire code base. We were particularly interested in issues which allow an untrusted or

[3]"Ballot Now is to be used only as central processing application [sic] and is not to be deployed to remote locations outside of central jurisdiction election headquarters." [21], page 33.

minimally trusted user to escalate his capabilities beyond those for which he was authorized. We also attempted to focus on issues that would require substantial rearchitecting to fix rather than on individual vulnerabilities unless they were reflective of systematic design or architectural issues.

We used a variety of tools to support our analysis. We used an open source wiki on our secure network to summarize and track the various software issues under investigation and to provide a common knowledge base among the team members[4]. We used the Fortify SCA static analysis tool to identify potential problem areas in parts of the Hart InterCivic software. (We are grateful to Fortify Software for making the tool available to us.) Various debuggers, program editors, decompilers and other tools were used to experiment with and confirm software structure and behavior.

We have deliberately avoided addressing the broader issue of whether or how this system should be used for voting in California. Making that judgement requires assessing not only the technical issues described in this report but also the procedures and policies with which the system is used.

[4]The contents of our wiki were submitted as the private annex to this report.

CHAPTER 2

Limitations

Our analysis focused on system design and architecture. Although source code was available, the large size of the code base and the limited time available precluded a comprehensive review of the source code. No security analysis can guarantee discovery of all system vulnerabilities; due to the short timeframe, this analysis is even more limited. Therefore, no assertions can be made about nonexistence of particular vulnerabilities in the source. We made a good-faith effort to identify and prioritize security vulnerabilities based on generally known and accepted security principles, but we caution that additional vulnerabilities may exist that are not documented in this report.

We did not attempt to verify that the voting software is completely free of defects or to exhaustively enumerate all defects, as that exceeds what is feasible with the state of art:

- The scale of the Hart InterCivic system makes a comprehensive review infeasible. The Hart InterCivic system contains over 300,000 lines of code (See Appendix A for details), which would have been impractical to review in the time allowed.
- Even thorough manual code review misses many problems. People are fallible; a code reviewer might overlook a defect in the code the same way that the developer who wrote the code did. A manual review that found half of all problems in the code would be doing pretty well, by industry standards, but that would still leave many undetected defects. Moreover, manual code review often misses architectural flaws and other issues that do not clearly manifest themselves at the implementation level.
- Manual source code inspection is laborious, time-intensive, and costly. A rough estimate is that a trained software engineer can inspect something like 100 lines of code per hour, under optimal conditions. If team members did nothing other than read source code for hours on end—something that few developers can sustain for any length of time—then it would have taken us a person-year or more of effort to read all of the source code. That would have significantly exceeded our budget and the time available to us.
- We made no attempt to find all bugs or vulnerabilities in the code. Once we found several related vulnerabilities in the same portion of the code, we stopped looking for other vulnerabilities of the same type. We made no attempt to catalog all bugs that might enable any particular kind of attack. Instead, we structured our analysis as an attempt to find evidence to confirm or refute the hypothesis that the voting system is secure from tampering. Once we found strong evidence that some aspect of the system was vulnerable to a certain kind of tampering, we moved on to examine some other aspect of the system. This methodology was selected due to the limited time available for this review and because it seems to make little difference whether an attacker has 3 or 30 different attack variations available to him if all the variations have the same impact.

As a consequence, the list of issues and defects in this report should not be taken as a comprehensive list. In our review, we singled out a limited subset of the code as especially critical, based on our architectural analysis, and then subjected that portion of the source code to intensive code inspection. Since we did find security vulnerabilities in the source code that was subject to inspection, one might anticipate that the rest of the code that we did not have time to inspect might also contain vulnerabilities. For this reason, it is likely that this report underestimates the true number of vulnerabilities in the code, and it is possible that the issues identified in this report might represent only the "tip of the iceberg." However, because we were only able to inspect a small fraction of the code, we simply do not know whether the rest of the code contains security vulnerabilities.

Because the time available for this report was so short, we were unable to perform as complete testing for each issue as we would have liked. Instead, we prioritized issues based on our estimates of potential severity and our confidence in our analysis of the code. This allowed us to directly confirm some issues and to partially confirm others. However, this still leaves a number of issues which we did not have time to confirm and others for which we would have wished to perform more tests. For each issue we discuss, we also describe the experiments (if any) we performed to confirm it, allowing the reader to draw their own conclusions. In many cases, issues were discovered in the source code without confirmation on actual Hart systems. For such issues, we describe our relative confidence in each one being a viable attack.

The analysis contained in this document is based on data gathered from documentation and source code provided by Hart InterCivic, the State of California, and the ITAs (Independent Testing Authorities). We made no attempt to validate the materials provided to us. Our conclusions depend on interpretation of those documents and source code. To the extent to which those materials are incomplete, inaccurate, or do not reflect the systems and practices currently in use in California, this may lead to material inaccuracies in this report.

There was some software that we were not provided and thus did not review. The software whose source code we believe was not made available to us is listed in Appendix A. However, we do not have a complete list of the software that is used in the voting system but whose source code was not provided to us. We understand that some of this software, such as the standard C libraries, may be classified as unmodified COTS (commercial off-the-shelf) software under the federal voting standards and thus may be exempt from scrutiny by the ITAs (Independent Testing Authorities). Other software was apparently written by Hart InterCivic but was not made available to us. In the absence of source code to the COTS software, we were not able to verify whether it qualifies as unmodified COTS under the provisions of the federal voting standards. It was beyond the scope of this review to evaluate whether any of the missing source code falls within the definition of unmodified COTS software, as defined in the federal voting standards.

We made no attempt to verify that the source code provided to us matches the binary code that is executed on election day in the Hart InterCivic voting equipment. That was beyond the scope of this review. Also, while we were provided with binary executables for some of this software, we were not provided with a full build environment that would enable us to compile the source code for ourselves and verify that the results were identical to the binary executables provided to us.

We made no attempt to search for security problems in the hardware. That work was done by a separate "Red Team." Instead, our analysis was based on analyzing the source code provided to us.

We did not attempt to search for configuration problems or to analyze whether the voting system, as installed in California counties, is configured correctly. This was out of the scope of this work and we did not have access to any county's Hart installations in any case.

We did not attempt to analyze the procedures, processes, or practices used by local election officials for potential security problems. We were given operator's manuals for the Hart InterCivic equipment, and we frequently used them to gain insight how the system might be used in practice. However, we were not provided with information about typical county-level practices, limiting our ability to examine how the voting system is used in the field and how those practices may enable or hinder security attacks.

The security of a voting system depends upon both the technology (e.g., the software) as well as upon how it is used (e.g., the processes and procedures in place). We were asked to focus in this review primarily on the software and accompanying documentation, not on how it is used. Therefore, any potential issues we identify in this report might or might be relevant to any particular user of the equipment, depending upon the practices in place in that jurisdiction. It would require a separate, follow-on study to evaluate the impact of these issues on individual California counties and other users of this voting system.

The scope of this work was limited to analysis of the Hart InterCivic voting system, not that of California's entire election system. For instance, voter registration systems, county practices, and election law were outside the scope of this study.

Our analysis was limited to a particular version of the Hart InterCivic voting system. This report is not intended as an endorsement or repudiation of electronic voting in general.

CHAPTER 3

Threat Model

The first step in security analysis of a system is to define the *threat model*. The threat model for a system is intended to describe the goals an attacker might have (e.g., to manipulate the vote count) and the types of attackers that might attempt to attack the system (e.g., voters, poll workers, etc.) as well as the capabilities available to each type of attacker. It is equally important to describe the threats that are out of scope for the analysis. This study was chartered to consider the security of voting systems proper only, not that of California's entire election system, and therefore many possible attacks are out of scope for this report.

3.1 Reference Model

In order to simplify the analysis, we assume a common reference model, which distills what are hopefully the essential features of all of the voting systems involved in this study. The system consists of the following components.
In the polling place:

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

At Election Central:

- 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 are ignoring early voting centers):

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.

Voting: On election day, voters arrive at the polling place, are verified as being permitted to vote, and cast their ballots.

Post-voting: After the polls are closed, the votes are tallied, the official canvass (including the one percent manual recount) is performed, and the results are certified.

The relationship between these components is shown in Figure 3.1.

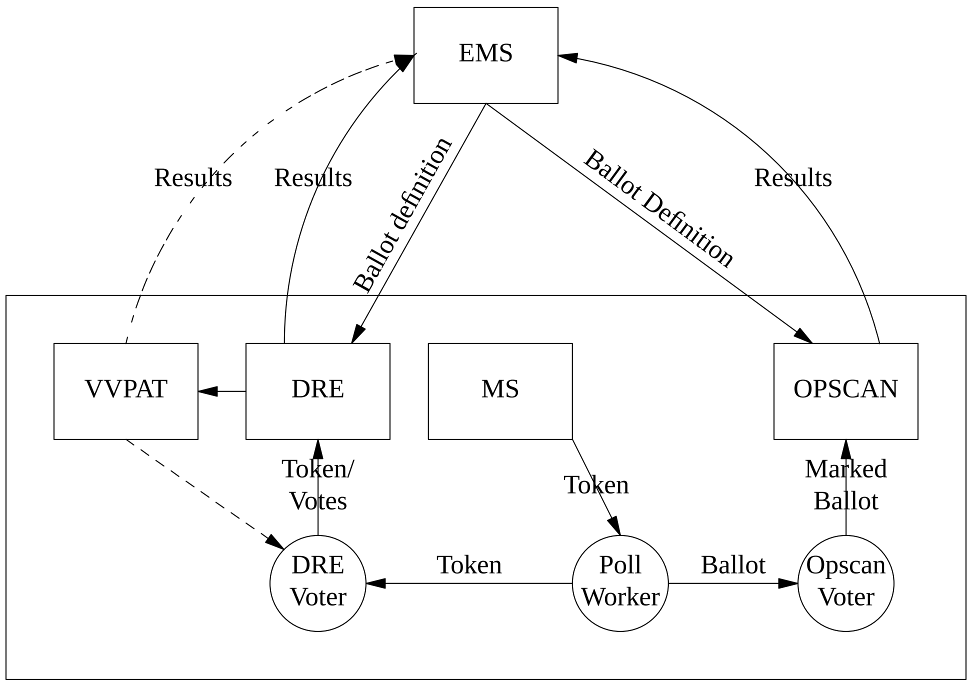

Figure 3.1: Reference architecture

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.
- Distribute the local voting equipment to the polling places.

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 may take the form of metal or plastic loops, individually numbered, which 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 technology and manufacturer in use, 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 centrally. 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 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 (i.e., the county's election headquarters) where they are tabulated with a high-speed scanner under supervision of the 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 records 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 may be transmitted via modems, 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 the 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 Diebold and Sequoia, the token is a smartcard. 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 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 California, a *voter-verified paper audit trail* (VVPAT) is required. On all the machines under study, this 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 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. 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 "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 the legally required 1% audit.

In larger counties, many vendors offer the ability to establish "regional processing centers." Election results are delivered by courier from the precincts to the centers, aggregated, and then communicated back to Election Central via electronic means (modems, Internet connections) or via courier.

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. In California, each county determines which model is most appropriate for their needs.

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 may involve reconciling the number of voters who have signed in against the number of ballots cast, performing the statutory 1% manual recount, and other tasks.

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).

The 1% 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.
- 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 mount a similar attack 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 overrepresented.
- 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.

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, but could potentially be used for extortion as well. These attacks would generally require tampering with the software within the voting machines.

3.2.3 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 Attacker Types

A voting system can be subject to attack by a number of different types of attackers with different capabilities and who will therefore be able to mount different kinds of attacks. 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 malware-based attacks. Outsiders may also be able to break into locations where voting equipment is stored unattended 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.

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.

A particular focus of security analysis is *privilege escalation*. In many cases, one participant in the system is forbidden to perform actions which 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. [27].

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—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.

3.3.2 Voters

It is very easy to become a voter in California 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.

Optical scan machines. In optical scan voting, the voter marks the ballot himself but the ballot is simply a special 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 malicious 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 voter is also provided with some kind of token used to authorize access to the DRE terminal to accept the voter's vote. In the Sequoia and Diebold systems this is a smartcard and in the Hart system it is a four-digit access code. As the voter has access to these tokens, they are also a potential target of attack and the voter might attempt to substitute his or her own token or subvert them (in the case of smartcards).

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. Voters may also be able to take advantage of "curb-side voting," where available, to have private access to a voting machine inside their car (where they may then have tools that would be infeasible to bring into a polling place and the privacy to use them).

Because any United States citizen and California resident is potentially a voter in California, 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, while procedural mechanisms may be sufficient to address such risks.

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).

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.

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, many attacks can be mounted using only the attacker's authorized access and within the context of the technical controls which the systems are expected to enforce. 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 able 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: they cannot be detected, 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 poses a threat to election confidence. 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 (e.g., recovery strategies that involve a 100% manual recount impose a heavy administrative and financial burden).

  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 has the potential to diminish public confidence, depending upon the circumstances. 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, in most contests, retail fraud is not enough to change the outcome of the election. Because wholesale fraud has a more significant impact, we focused especially on analyzing whether the systems are vulnerable to wholesale fraud.

- *Casual vs sophisticated.* Some attacks require little technical knowledge, sophisticated, 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 Hart InterCivic voting system, where possible we identify these distinctions to enable readers to form their own judgments about the severity and impact of those attacks.

3.5 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 flip-side of the coin to 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). For 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 [11] and Nedap/Groenendaal [14] voting systems.

Wire loops Many voting machines have adopted a mechanism commonly used with traditional ballot boxes—the use of holes through which a 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 [31].

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 [28].

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" [22]. 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 [22].

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.

CHAPTER 4

Overview of System Architecture

We now present a brief overview of the various components of the Hart InterCivic architecture. We will consider all the major components, in the rough order in which they are used in an actual election. Note that not all counties use all of the components described here.

4.1 Pre-Election

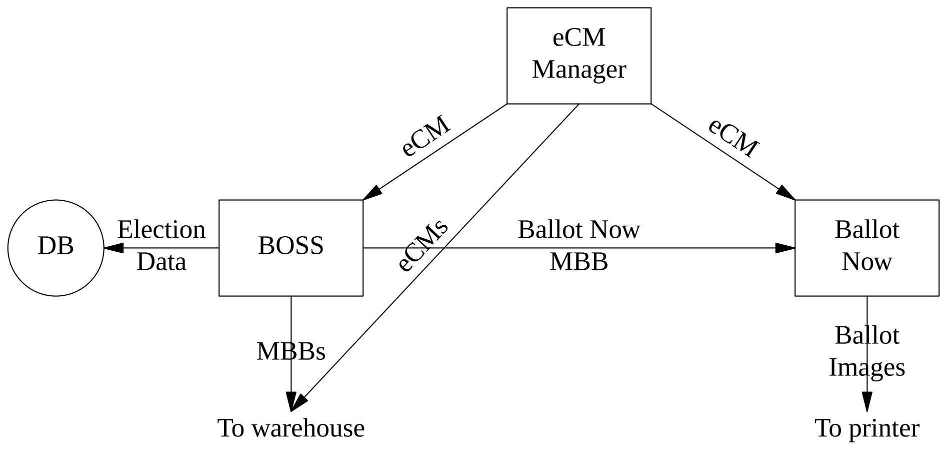

Figure 4.1: Hart Election Setup

Figure 4.1 shows an overview of setting up an election with the Hart system.

The first step in creating a Hart election is to create a cryptographic “master key” that is used to detect attacks against the integrity of data throughout the election. This key is stored on *eSlate Cryptographic Modules* (eCMs), which are PKCS#11 USB cryptographic tokens, about the size of a pack of gum. The master key is generated using the *eCM Manager* software, which runs on a standard Windows-based PC. The eCM Manager loads the key onto each eCM.

A Hart election is set up with the *Ballot Origination Software System* (BOSS) application, running on a standard Windows-based PC. BOSS is used to define the election, i.e., to create the list of candidates running for each office, and to define how precincts and party primaries create many different variants on each ballot. When complete, this produces an “election database.” The underlying database is sourced from Sybase, one of the major commercial database vendors.

In order to export election information to the precinct voting devices, BOSS writes the election definition to a *Mobile Ballot Box* (MBB), which is a standard PCMCIA type-1 flash memory card.

The ballot definition is digitally signed[1] using the cryptographic master key.

Hart supports both optical scan and direct recording electronic (DRE) voting. The optical scan ballots are prepared using the Ballot Now program, which runs on a standard Windows-based PC. Ballot Now exports the ballot image in electronic form for export to an external ballot printer.[2] The election definition is transferred between BOSS and Ballot Now on an MBB.

4.2 Preparing Voting Devices

The Hart precinct voting system consists of three separate devices:

eScan. a self-contained optical scanning vote counter

eSlate. a DRE voting device

Judge's Booth Controller (JBC). the control terminal for the eSlate.

All of these must be prepared before the election can take place. Preparation is performed at the central warehouse and consists at minimum of zeroing the vote counters and installing the per-election master key. It may also involve verifying the device firmware and installing the MBBs.

Device preparation and management is performed with the *System for Election Records and Verification of Operations* (SERVO), which is a program running on a standard PC. In the warehouse, SERVO is connected to the devices as shown in Figure 4.2.

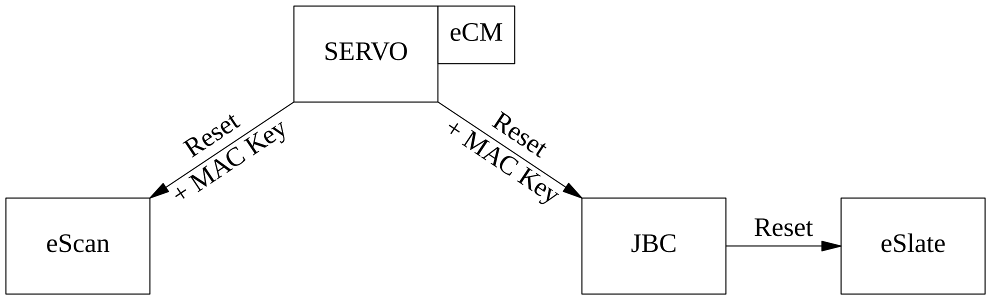

Figure 4.2: Hart Device Setup

SERVO is connected to the eScan by means of an Ethernet cable and to the JBC by way of a parallel data cable. SERVO cannot directly talk to an eSlate and must use a JBC as a go-between. An eCM containing the master key is plugged into SERVO, which installs the key on the JBCs and eScans (but not the eSlates) and zeroes the vote counters. SERVO also serves an inventory-control function, tracking the serial numbers of every device owned by the county and can be used to verify device firmware.

At this time, MBBs may also be installed into the JBCs and eScans, in which case they are tamper-sealed at the warehouse. Alternately, the MBB may be shipped separately to the precinct. In either case, the prepared devices are then shipped to the precincts or the local poll workers.

[1]Hart's terminology. More precisely, it's a shared-key message authentication code, with the same key used throughout the election.

[2]Ballot Now can also be used for on-demand in-precinct ballot production and scanning, but the California use procedures [21] specifically prohibit Ballot Now operation outside of the central office, so we do not consider in-polling place applications of Ballot Now.

4.3 Election-Day Setup

Hart supports two varieties of in-precinct voting: optical scan voting using the eScan and DRE voting using the eSlate/JBC. Figure 4.3 shows the setup.

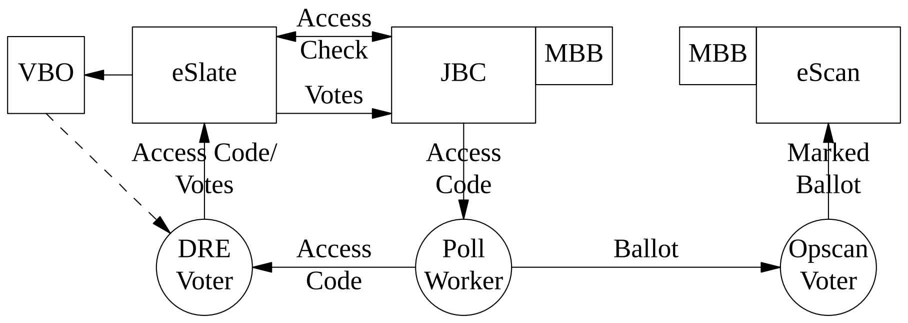

Figure 4.3: Hart Polling Place Setup

Once an eScan is loaded with an MBB it operates as an island unto itself. When ballots are inserted, it can validate and tabulate the votes, or it may reject malformed (e.g., over-voted) ballots. The *cast vote records* (CVRs) are recorded on the MBB.

DRE voting is supported using one or more eSlate systems connected with a local area network to a single JBC. Larger precincts or voting centers may well have multiple JBCs, each with its own group of eSlate machines. As with the eScan, the JBC must be pre-loaded with an appropriate MBB.

4.4 Authorizing and Casting Votes

Unlike the eScan, the eSlates rely on their communications with the JBC that controls them. In a typical scenario, a voter is first validated as being a legitimate voter for the local precinct. Then, they enter a queue that leads to the poll worker operating the JBC. This poll worker selects the proper ballot definition, where appropriate (e.g., giving the voter the Democratic or Republican primary ballot), and then the JBC will use its built-in thermal paper printer to output a random four-digit access code. The voter takes this printed code and approaches any open eSlate. The voter is granted access by the eSlate to cast his vote by entering the access code.

When the voter has completed voting, the eSlate prints out a summary of his selections using the *Verified Ballot Option* (VBO) printer (the VBOx). The voter then has an opportunity to accept or reject the ballot. Once the voter has accepted his ballot, the VBOx marks the paper “BALLOT ACCEPTED” and the ballot (CVR) is stored inside the eSlate, inside the JBC, and on the MBB stored within the JBC.

4.5 Vote Collection and Tallying

At the end of the election day, results need to be collected and tallied. Hart systems allow for a number of different procedures, as shown in Figure 4.4.

We will describe several of the possibilities. First, the JBC and/or the MBB from within it must be carried via courier to a processing facility. This might be Election Central, or in larger counties it might be a regional processing center. Similarly, an eScan might be couriered in its entirety, or the eScan might be shipped separately while the MBB is sent to be processed.

Either way, the MBBs are loaded into a Windows PC running Rally or Tally. The software extracts the contents of the MBB, verifies that they have not been seen before, and stores them in a

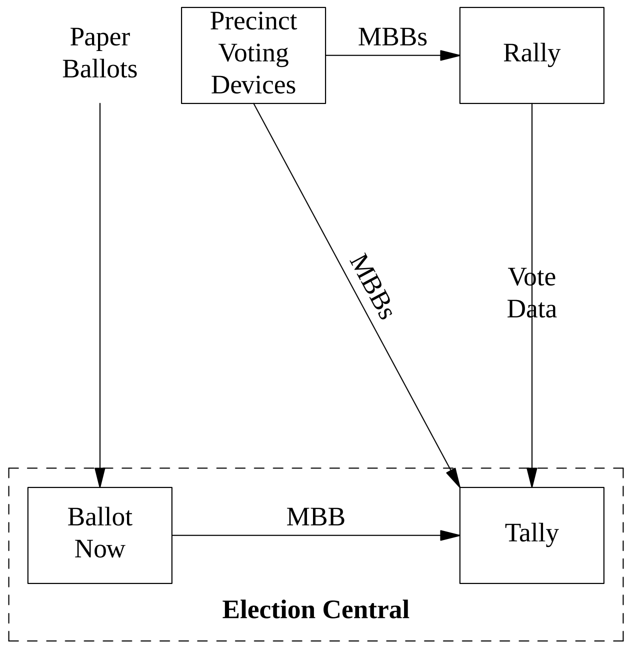

Figure 4.4: Hart Vote Collection and Counting

local (Sybase) database. In the case of regional processing centers, they will be running Rally and their computers must either be connected to a network or be connected to modems and configured to accept phone calls. Election Central, running Tally, will make phone calls to each regional center, running Rally. Tally will authenticate itself to Rally and then, over an SSL-encrypted session, will download all the MBBs stored within Rally.

Absentee votes are scanned using the Ballot Now software and a COTS scanner, which provides detailed procedures to allow the operator to disambiguate ballots. Once this process is complete, Ballot Now writes out an MBB which can then be accumulated alongside the other MBBs. In some counties, the eScan is not used and all ballots are scanned with Ballot Now.

Once all the MBBs are present within Tally, either because they were loaded locally or because they were copied from remote Rally systems, Tally can then compute the vote totals and produce a wide variety of reports. We note that both Rally and Tally require the presence of an eCM, in order to obtain a copy of the election's cryptographic master key, so that any tampering with MBBs (or, any inadvertent use of MBBs from other elections) can be detected.

4.6 Post-Election Auditing

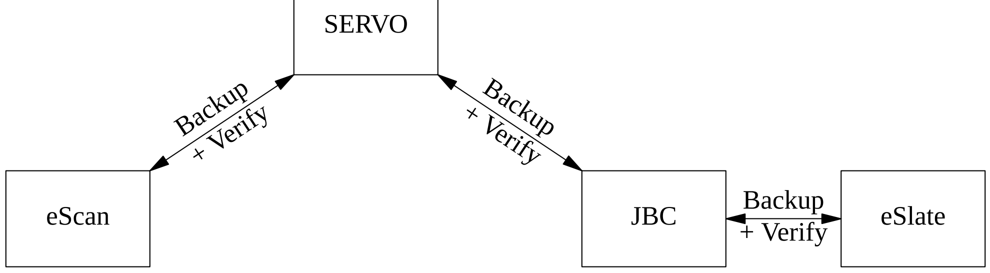

Figure 4.5: Hart Post-Election Auditing

Once the eSlate, eScan, and JBC machines have been returned to Election Central, they may be again connected to SERVO, as shown in Figure 4.5. SERVO can extract their internal copies of the vote data and store backup copies. These copies can subsequently be used to produce a recount MBB which can be fed to Tally to perform a machine recount which can be compared against the official totals. This process is optional and may not take place until well after the official election results have been certified. Likewise, the paper ballots that had been originally scanned using the precinct-based eScan system can be rescanned with Ballot Now. There do not appear to be any procedures for automatically performing an audit. Rather, the audits are performed and then the results manually compared.

At this time SERVO can also be used to verify the device firmware. However, the recommended procedure is that the devices not be zeroed in case a subsequent audit is required.

CHAPTER 5

Architectural Issues

As discussed in Section 1.2, the focus of this review was on architectural issues. Unlike simple programming errors, architectural errors pervade the entire system and therefore can be difficult to eradicate. Our review identified four major architectural errors that are made throughout the Hart system:

Authentication Failures. Components of the Hart system routinely assume that any input they receive from an entity speaking the Hart protocol is authorized and will act on it without question.

Least Privilege Violations. In many cases the Hart protocol allows peer agents to perform actions that are unnecessarily powerful.

Lack of Input Validation. The Hart software frequently fails to check input values before using them internally.

Misuse of Cryptography. Cryptography is not used in places where it should be; where it is used, it is used in brittle ways.

All of the issues described above can be viewed as aspects of a single larger architectural issue: the Hart system fails to exhibit *defense in depth.* Secure systems should be designed in such a way that compromise of single elements does not lead to compromise of the entire system. Rather, the system should be designed in such a way that other elements act to contain compromise of a single element, so that the larger system continues to function correctly.

In many respects, Hart's system exhibits the opposite property: a *distributed single point of failure*. An attacker who compromises any of a large number of elements can leverage that attack to cause other elements of the system to misbehave, in many cases even when those other elements are functioning as designed. This is inconsistent with good security design.

The rest of this section discusses these general issues at an architectural level. The next section provides a detailed description of the issues we have found.

5.1 Authentication Failures

The various components of the Hart system routinely communicate between each other over a variety of network interfaces, including:

- SERVO to eScan (Ethernet)
- SERVO to JBC (Parallel)
- JBC to eSlate (EIA-485)
- Voter Registration Computer to JBC (RS-232)
- Rally to Tally (Internet or modem)

Because these channels are used for important communications, such as uploading firmware, authorizing votes, and returning votes they constitute an attractive attack target.

The fundamental starting point for designing a secure networked system is to assume that the attacker has control of the communications channel. This is particularly important in the case of a system like Hart's where the attacker has physical access to the communicating devices. Therefore, any communication must be assumed to be originated by or modified by the attacker until demonstrated otherwise. The standard technique for establishing the identity of the communicating party and the integrity of its messages is to use a secure communications protocol such as SSL/TLS [8].

With the exception of the Rally/Tally communications, which do use SSL/TLS[1], these remaining communications channels are completely unsecured. The underlying assumption here appears to be that because the interface specifications and protocol are proprietary, any node which speaks that protocol must be legitimate. This is only true to the extent to which the protocol remains secret. An attacker who has access to either the source code or the equipment can, with only a modest amount of effort, decipher the protocol and create his own implementation. We ourselves have developed a partial implementation of the protocol and used it in attacks on the Hart system.

Vendors of proprietary network systems often assume that because they do not publish protocol specifications or source code, no attacker will be able to reverse engineer their protocols. This assumption is largely untrue. With enough effort and access to a system, it is generally possible for an attacker to reverse engineer most protocols. Probably the most famous example of such an effort is Samba[2], an independent reimplementation of Microsoft's SMB server which was to a great degree (though not exclusively) developed via this kind of analysis. The protocol Hart uses is especially susceptible to this kind of attack because many messages can simply be replayed without modification and still have the desired effect. Therefore, reverse engineering the protocol may not even be necessary if what is desired is to simply repeat an action that was observed in the past.

Repairing this issue most likely requires adding cryptographic authentication and encryption to all inter-device communication, which likely would entail reworking Hart's entire authentication model.

5.2 Least Privilege Violations

One of the fundamental principles of secure systems design is what's termed the "principle of least privilege": an agent should be given only the minimal capabilities required to complete the tasks it is expected to perform. Hart's systems violate this principle in two ways:

- The commands used to implement a given task are often significantly more powerful than required.
- The capabilities required to execute commands are often not limited to the agents and times when they are needed.

As an example of the former issue, Hart provides a general memory reading command that allows SERVO—or anyone pretending to be SERVO—to read arbitrary portions of the eScan permanent storage or of JBC/eSlate memory. SERVO uses this mechanism in an attempt to verify the device firmware (see Issue 11), however it can also be used to extract cryptographic keys and other sensitive information. This command could be limited to specific sections of firmware without sacrificing the desired functionality.

As an example of the latter issue, Hart provides a set of commands which can be used to update the firmware. These commands are only ever legitimately used by Hart representatives under tightly controlled conditions. However, our analysis suggests that these commands are available at all times to any agent which can access to the required interface. A superior design would be to lock down these commands entirely when the devices are in the field, with the unlocking requiring

[1]Though we have concerns about this usage as well, as described in Section 6.9.
[2]`http://www.samba.org/`

explicit action by an election official or Hart representative. It would of course be necessary to ensure that that unlocking could not be spoofed, which can be accomplished in a variety of ways.

Addressing this design issue would likely involve an analysis of the entire command set to determining the required scope and functionality of each operation.

5.3 Input Validation

We found numerous instances in the Hart system of failure to check inputs that are received from other entities. These inputs are then used in a variety of unsafe ways, such as copied into a fixed-size memory buffer or passed as a format string argument to `printf`. This violates the basic secure programming practice of assuming that any input received from outside is malicious and using it only after it has been rigorously checked. In the case of Hart's system, this leads to multiple remotely exploitable vulnerabilities, discussed in Issue 13 and elsewhere.

We wish to emphasize that these are not complicated errors where a tainted data item follows a tortuous path to the point of exploitation—though we identified this type of issue as well. The Hart system repeatedly takes data directly off the network and uses it in unsafe operations within a few lines of the point where it is read. This practice is totally unsafe and trivially exploitable with techniques taught in introductory security engineering classes.

A related error is the implicit assumption that peers are well behaved. For instance, we found errors where Hart would query a peer for the length of a value, allocate a buffer of that length, then query again for the length and use the second answer without checking that it was the same as the first. This, too, is inconsistent with the assumption that the peer can be malicious, and is a well-known type of error, known in the literature as a *time-of-check-to-time-of-use* (TOCTTOU) bug.

Addressing this issue requires that Hart perform a complete audit of every section of the code which reads and processes data from the outside world and in each case ensure that the data is thoroughly checked before any other operations are performed on it. This will require pervasive changes throughout the code.

5.4 Misuse of Cryptography

Although Hart does not use cryptography in a number of places where it would be helpful, they do use it in at least three cases:

- To protect communication between Rally and Tally
- To provide integrity protection for the MBB and
- To randomize vote data.

In the latter two cases, the techniques provide a far lower level of security than is desirable. The two major data items on the MBB which need to be protected are:

- The ballot data in transit from BOSS to the polling place.
- Vote data in transit from the polling place to Rally/Tally.

Logically speaking, these operations are quite separate. The first category of data is generated by Election Central and verified by the polling place. The second category of data is generated by the polling place and verified by Election Central. Basic cryptographic principles dictate that separate types of data should be protected with separate keys. Hart, however, uses a symmetric message authentication code (MAC), which requires that both sides have the same key. Moreover, the same key is used county wide.

The impact of this choice is that an attacker who has access to this single key can:

- Forge ballot data which will be accepted by any precinct.

- Forge vote data from any precinct which will be accepted by Election Central.

This choice is exacerbated further by poor key hygiene. Keys are centrally generated in software, then transferred to a hardware token, then transferred via an unsecured network interface to memory on the JBCs and eScans. This violates standard cryptographic practice with hardware security tokens, which is to use the key exclusively on the token. (Modern cryptographic tokens can perform a variety of cryptographic operations internally without ever divulging the keys used for these operations.) The result is that there is a large number of soft targets, compromise of any one of which suffices to compromise the entire cryptographic scheme.

Even without the device management issues described in Section 6.1, extracting these keys would not be particularly difficult. An attacker with physical access, such as a poll worker, can simply open the case of a single eScan or JBC and directly read the key from its internal memory card. Even if the violation of the tamper seals is detected, he can simply claim that the seals were broken when he received the device. According to Hart's suggested procedures, this would require notifying the chief election official, but at most would require taking the compromised unit out of service ([21], page 26, see also page 45), which does not prevent the attacker from using the extracted key.

A more robust design would use public key cryptography, with each device issued its own key. This would stop a compromised polling place device from impersonating either Election Central or another polling device. This design could be further hardened by distributing all keys on secure tokens with the keys never exported and all cryptographic computations performed on the token.

Hart also uses cryptographic-style techniques in the pseudorandom number generator which generates the voter codes. Design of cryptographically strong pseudorandom number generators is a well-studied problem in the literature, but home-grown techniques are typically quite weak and Hart's is no exception, as discussed in Issue 7. This issue can be relatively easily fixed by replacing the PRNG with a strong one from the literature.

CHAPTER 6

Detailed Analysis

We now consider individual attacks that we have discovered on the Hart InterCivic voting system. For each attack, we describe, briefly, how the attack works, and what capabilities are necessary for an attacker to perform the attack. We also discuss the impact the attacks may have and what mitigations may be available. Because the terms of the review forbid publishing full details of attacks on the Hart system, some of the descriptions below are incomplete. More complete descriptions, including, where relevant, the tools required to exploit the issues, were made available to the State in a private annex to this report.

6.1 Device Management

The polling place devices (eScan, JBC, eSlate) are all designed to be managed by programs running on ordinary computers:

- Hart uses a special program to update the firmware on these machines.[1]
- SERVO is used to backup election results and audit logs, verify firmware, and reset the ballot counters in preparation for the next election.

In both cases, management is performed by connecting the device to be managed to the management station via a physical cable. In no case did there appear to be any security measures applied to prevent unauthorized personnel from managing the devices themselves. This represents a significant security issue in that an attacker could use it to compromise voting equipment.

All network communications between Hart's devices, whether over TCP/IP, Parallel, or EIA-485 serial interfaces, use a common data protocol. Understanding this protocol is the key to successful analysis and exploitation of the Hart system. A large number of our attacks depend on being able to read and/or write protocol messages. We describe the protocol here in order to give readers a sense of how the protocol works and of the types of capabilities we use in implementing our attacks.

At the link layer, each message is framed with a link layer header and a checksum:

- Command byte (fixed)
- Length
- Payload
- Checksum

[1]We were not provided with a copy of Hart's firmware upgrading tools, but we were able to reconstruct some of their functionality from a protocol trace captured by the Red Team and by analyzing the source code.

The link layer strips off that framing to reveal the application layer message.[2]

Above the link layer, the Hart protocol is a reasonably conventional datagram protocol. Each node on the network is assigned an identification number. In the case of the JBCs, eScans, and management stations this is fixed. In the case of eSlates, each eSlate receives an identifier from one to twelve, denoting the eSlate's voting booth ID.

Every message shares a common header, which contains:[3]

- Source address
- Destination address
- Response address
- FrameID
- Length

One somewhat unusual feature is the response address, which indicates the node to which a response should be sent. Many protocols simply reply to the from address. We have not investigated this in detail, but it appears to be mostly used to indicate no response is needed.

The FrameID contains the command being executed or responded to. The rest of the message is a type-specific payload. Describing all the commands is out of the scope of this report, but a few of particular interest include:

File access direct read and write to arbitrary portions of the eScan file system. We used this capability in the firmware replacement attack described in Issue 3. which could be used to completely take over the device.

Read access to arbitrary memory locations which could be used to read cryptographic keys or vote records from the device. For the eScan this only applies to static storage. For the eSlate and JBC this appears to apply to all of main memory.

Write access to arbitrary memory locations which could be used to write MBB data and, in the case of the eSlate/JBC, take over program execution. It could also be used to tamper with any audit logs currently on the device. As with read access, write access applies to static storage on the eScan and to all of main memory on the eSlate and JBC.

Reset vote counters which could be used to reset the private protective counter in flash memory.

Generate a system error which could be used to reset the device.

Reset used to restart devices.

All three devices speak variants of this same binary protocol, although some commands have different behavior depending on which device they are addressed to or do not exist on some types of devices. The same basic protocol is also used for ordinary communication between the eSlate and the JBC, as described in Section 6.2.

Because we had source code, we did not need to reverse engineer the protocol but were able to build encoders and decoders based on the source code. However, cursory binary analysis suggests that an attacker with access to a device, a copy of the management software, and an appropriate sniffer could reverse engineer substantial portions of the protocol. If we consider briefly the case of an attacker who observes a firmware install, we realize that he observes:

- The commands necessary to install the firmware image.
- The firmware image itself.

[2]The control channel for the the eSlate EIA-485 interface uses the same link layer framing but a different upper layer protocol.

[3]Note: not all fields are shown

It's relatively straightforward to determine which parts of the communication are the firmware and which are the protocol wrapper. At this point, the attacker both has access to a delivery mechanism and to the binary which decodes it in one convenient package. It's straightforward (though time consuming) able to disassemble that to reverse engineer the binary and recover the rest of the protocol.

Access to the management interface is therefore extremely powerful and in the current design appears to lead to complete control of the device being accessed.

Issue 1: The JBC is managed via an accessible parallel port

The JBC is managed via a parallel interface: a DB-25 connector on the rear of the device labeled "Printer." This interface does not appear to be protected by any kind of door and we were not able to determine whether there is any requirement that it be tamper sealed or taped. Protecting this interface in the field would likely be fairly difficult because it is very close to the DE-9 port used to talk to the eSlates, which is used in the field. An attacker who can access this port can impersonate a management station and potentially subvert the JBC.

Detailed Description When the JBC boots up it installs an *interrupt service routine* (ISR) on the parallel interface. When data comes in on that interface the ISR reads a full message and then puts it in a message queue for the "service task". The service task determines whether the message is for the JBC or for a connected device (an eSlate). If it is for an eSlate it sends it out the serial interface to the eSlate. If it is for the JBC it is processed locally.

The JBC service task will process—without any authentication—any message it receives via the parallel interface. Thus, an attacker who understands the protocol can simply connect to the parallel port and start issuing instructions which the JBC will obey. These instructions include the management commands indicated above.

Prerequisites In order to exploit this issue the attacker would need:

- To have decoded the protocol used by Hart.
- Access to the parallel port on the JBC.

As indicated at the beginning of this section, we believe that an attacker who had an opportunity to observe a running system could decode substantial portions of the protocol.

Currently, any poll worker has access to the parallel port on the JBC. The amount of time required is quite small. We have observed JBC software upgrades (reflashing) in the field and they appear to take less than a minute. Other commands would presumably be even faster since little data transfer is required. In addition, the attacker may need to reset the JBC to load the new code. This can likely be done remotely but can also be done by simply pulling the plug. How noticeable this is depends on how often such failures occur in normal usage. Note that the JBC will recover gracefully from a power failure of this type. It may also be possible to patch the running image in memory.

Impact An attacker who took control of a JBC would have direct control of every eSlate in the precinct. In particular, the eSlates get their information about ballots and vote authorization from the JBC. Moreover, as described in the next section, we believe the JBC can use the management interface to subvert the eSlates. See Section 7 for a range of possible attacks using this access.

Mitigations The primary JBC-specific mitigation is to somehow secure the parallel port. There does not appear to be any legitimate use for this port in the polling place. General mitigations for all of the issues discussed in this section are discussed in Section 6.1.

Future versions of the Hart hardware might add doors with locks or other suitable mechanisms to protect the parallel port from tampering while the JBC is in the field.

Status This issue was discovered by examination of the source code. An incomplete understanding of the hardware interface has stopped us from replicating it with a real JBC.

Issue 2: The eSlate is managed via a serial port connected to the JBC

The eSlate is managed via a serial interface on the rear of the device. In order to manage the eSlate, one plugs the eSlate into the JBC and connects the management station to the JBC. All messages between the management station and the eSlate are sent through the JBC. This interface is not protected by any kind of door, nor is it tamper sealed. In fact, Hart's curbside voting feature explicitly contemplates that the eSlates will be unplugged.

An attacker who can access the eSlate interface can impersonate a management station to the eSlate and potentially subvert the eSlate. Mounting this attack would be extremely quick, on the order of seconds.

Detailed Description The eSlate is connected to the JBC via a serial interface (see Section 6.2 for details). On startup, the eSlate "network task" begins and waits for instructions over this interface, either from the JBC or from a management station routed through the JBC. In either case, the eSlate obeys those instructions without question.

By design, all communication to the eSlate is via the JBC. However, an attacker who had the specifications to this interface could connect directly and impersonate the JBC (and hence the management system) to the eSlate. Thus, controlling this interface would allow an attacker to manage the eSlate, as discussed above.

Prerequisites In order to exploit this issue the attacker would need:

- To have decoded the protocol used by Hart.
- Access to the serial port on the eSlate.

As indicated at the beginning of this section, we believe that an attacker who had an opportunity to observe a running system could decode substantial portions of the protocol.

Currently, any voter has access to the serial port on the JBC and can easily remove the connector. The eSlate is designed to be disconnected to enable curbside voting and therefore it is possible to remove the last eSlate in the chain. (See Section 6.2.3).

In order to install new firmware on the eSlate, it would probably need to be rebooted, however this could likely be done without disturbing the JBC. It may also be possible to use the memory commands to patch the running memory image. Any of these attacks could plausibly be mounted by a voter and could certainly be mounted by a poll worker.

Impact A successful attack of this type would likely lead to complete control of the eSlate. An attacker who controlled an eSlate would be able to access or control every vote performed on that eSlate. He might also be able to leverage this access into control of the rest of the eSlates in the system, as described in Sections 7.3 and 6.2.

Mitigations The natural mitigation is to secure the connection between the eSlate and the stand via some kind of seal. However, we saw no obvious sealing points. In addition, the red team informs us that the eSlates need to be demountable in order to install potential disabled access devices such as sip-puff devices. In addition, each eSlate is connected to the next eSlate, which provides an additional cable attachment point. See Section 6.2 for more details on the difficulties of securing this interface.

Status This issue was discovered by examination of the source code. We have verified that we can observe communications between the JBC and eSlate and were able to verify the lack of cryptographic security. We have not attempted to impersonate a JBC or management station to an eSlate.

Issue 3: The eScan is managed via an accessible Ethernet port

The eScan, like the other devices is remotely manageable. The eScan is equipped with an RJ-45 Ethernet jack and automatically adopts a fixed IP address. It can be remotely managed via TCP/IP.

Detailed Description When the eScan boots it automatically starts a "service task" which listens on TCP port 4600. The eScan seems to have fixed IP address of `192.168.0.1`. The eScan service handler treats any TCP connection on this port as the management console and will process commands issued via the TCP connection without question.

Prerequisites In order to exploit this issue the attacker would need:

- To have decoded the protocol used by Hart.
- Access to the Ethernet port on the eScan.

As indicated at the beginning of this section, we believe that an attacker who had an opportunity to observe a running system could decode substantial portions of the protocol. This is particularly easy with TCP/IP connections because tools such as Wireshark and `tcpdump` for analyzing TCP connections are readily available and widely used for precisely this sort of analysis.

Currently any poll worker has access to the Ethernet interface on the eScan. We do not know if voters would be able to access the interface in the field. This depends on the physical mounting of the eScan on its cart as well as what sort of supervision voters are under.

Impact A successful attack of this type would likely lead to complete control of the eScan. The attacker could cause any vote counts he chose to appear as well as rejecting "invalid" ballots of his choice. See Section 7 for other attacks based on controlling an eScan.

Mitigations The natural eScan-specific mitigation is to secure access to the Ethernet port on the eScan. There does not appear to be any need to have it available in the polling place. General mitigations for all of the issues discussed in this section are discussed at the end of Section 6.1.

Status This issue was discovered by examination of the source code. We have successfully used a computer under our control to communicate with the eScan and extracted the memory information, the initialization file, and the cryptographic keys. This channel is used by Hart to manage the firmware on the eScan and based on the source code and a capture of a firmware upgrade transaction (provided by the red team) we were able to write our own tool to update the firmware. We worked with the red team to tested this tool with the red team's eScan and were successfully able to load our own firmware, which displayed a revised Hart logo.

Issue 4: The JBC voter registration interface can be used to generate voter access codes

The JBC contains a DE-9 male connector on the rear of the device labeled "Modem". This is a serial interface which is internally referred to as a "Voter Registration Interface" (hereafter VRI)[4]. A VRI device can send instructions to the JBC to issue voter access codes. As with the JBC's parallel interface, this serial interface does not appear to be protected by any kind of door and we were not

[4]While there is no certified Hart InterCivic product in California that connects to this port, the support for it is clearly present in the JBC source code.

able to determine whether there is any requirement that it be tamper sealed or taped. An attacker who can access this port can impersonate the VRI device and generate voter access codes that can be used for voting on one of the eSlates managed by this JBC. Further, if the JBC is in Early Voting mode, access codes can be obtained by an attacker without causing records to be printed to the JBC's printer.

Detailed Description After the JBC boots up the and low level initialization is finished, the "main task" is started. The main task listens on the serial interface (labeled "Modem" on the back of the JBC, referred to as the VRI in the source code) for voter access code requests. If the JBC is in Early Voting mode, then access code requests can include an instruction to not print any records to the JBC printer. Otherwise, access code requests trigger a receipt being printed.

The JBC "main task" will process—without any authentication or cryptographic integrity checking—any message it receives via the VRI. Thus, an attacker who understands this protocol can simply connect to the serial port and start issuing voter access code requests.

Prerequisites In order to exploit this issue the attacker would need:

- To have decoded the voter registration protocol used by Hart.
- Access to the VRI (i.e., the serial port labeled "Modem") on the JBC.

Similar to the case with a firmware upgrade, an attacker who had an opportunity to observe data exchanged between a VRI device and a JBC could decode substantial portions of the protocol. This protocol is also partially described in the *JBC Functional Specification* [15].

One question worth asking is the duration of required access. The Red Team was able to connect to a JBC in Early Voting mode and extract several voter access codes in a very short amount of time.

Impact An attacker who could generate valid voter access codes could vote as many times as the number of access codes generated. The documentation indicates a limit of 150 outstanding access codes but we have not verified how well this is enforced.

If a JBC was used with a barcode reader, then an attacker would not need to connect an external computer to the serial port. Instead, the attacker could pre-print suitable barcodes and expose them to the reader.

Mitigations The primary JBC-specific mitigation is to somehow secure this serial port. If no computer or barcode reader is intended to be connected to the VRI (and, we are not aware of any reason why one should be), then the serial port should be suitably sealed to prevent access to it during the election.

General mitigations for the issues discussed in this section are also discussed in Section 6.1.

Status This issue was discovered by examination of the documentation and source code. The Red Team was able to successfully connect to a real JBC's VRI and generate voter access codes.

Mitigations

All of the issues discussed above stem from the same underlying architectural issue: all three devices assume that any device which talks to them and issues management-style commands is a legitimate management console. This assumption is incorrect and the failure to authenticate attempts at management allows any attacker who can speak the protocol to manage the device.

As noted above, it is conceivable to secure the ports on the eScan and the JBC but perhaps not the eSlate. To the extent that the seals are secure (see Section 3.5), this provides protection against voters and poll workers, but not election officials with access to unsealed devices. A sealing strategy implies that no one who is not completely trusted can ever be allowed unsupervised access to a device which does not have its ports sealed. This is difficult operational bar to clear. However, a sealing strategy may be useful in addition to other hardening strategies as a backup for potential programming errors in the software doing access control.

There are a number of modifications to the software architecture that would mitigate these issues. First, if management consoles were required to authenticate cryptographically to the devices they managed this would substantially reduce the risk of fake management consoles. Note, however, that the existing keying material management strategy (see Section 6.7) makes this difficult because compromise of any single device would allow the attacker to recover the key and make a fake management console. Public key authentication of the consoles would be safer but would be a significant architectural shift.

Even if the management consoles were to be authenticated, there would be substantial residual risk from an attacker who compromised a management console. This risk could be mitigated by substantially reducing the range of operations the console could mount. In particular, the console can currently load arbitrary firmware. If firmware needed to be signed by Hart or a third party (and this was enforced by the device) then even a rogue management console would not be able to load malicious firmware. In addition, a number of the operations (generic memory reading and writing in particular) appear to be stronger than required for a management system. Strictly trimming this list to the operations which are absolutely necessary for management would mitigate the risk.

Finally, the command set could be partitioned into commands which were accessible in the warehouse and commands which were accessible in the field. It is not clear that there is any actual need for field management, Special access (e.g., a PIN entered on the console) could be required to put the unit into warehouse mode, and perhaps to reactivate it. This would limit the exposure from unauthorized field management.

6.2 eSlate-JBC Communication

The eSlate has no MBB of its own. This means that it must get election information from the JBC to which it is connected and must return voting results to the JBC for storage. This communication that occurs over the eSlate-JBC interface using the standard Hart protocol (See Section 6.1). As discussed earlier, this protocol has no security features, which leads to a number of issues.

In the Hart voting system a number of eSlates are connected to a single JBC in a daisy chain configuration, with the first eSlate connected to the JBC and each subsequent eSlate connected to the one before it, as shown in Figure 6.1. This network is used by the JBC both to communicate with and supply power to the eSlates.

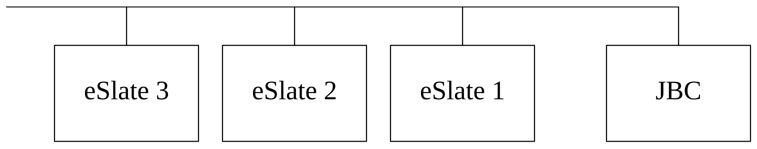

Figure 6.1: JBC/eSlate Network

This broadcast network. All of the devices are electrically connected and any message sent from any device to any other device can be seen by all the other devices in the network. The way that devices know which messages are intended for them is that there is additional addressing information (again, visible to all). This is a fairly common network design. Classic Ethernet, for instance, has some similar properties.

6.2.1 Detailed Description

Physically, the cabling is as follows:

- The JBC has a female DE-9 connector on the back.
- A cable with a male DE-9 on one end and a female HD-15 on the other end runs to the back of the eSlate stand for the first eSlate.
- The eSlate stand has a male HD-15 connector on the back and an internal cable that terminates in a female HD-15, which plugs into the eSlate.
- The eSlate has a male HD-15 for the above cable.
- A cable terminating in a female HD-15 comes out of the eSlate and is run to the back of the next eSlate stand.
- In the last eSlate, the cable is curled up and stuffed into a compartment in the stand.

Electrically, the network contains:

- One pair of wires for an EIA-485 [9] control channel[5]
- One pair of wires for an EIA-485 data channel
- +24 V power wires
- Ground wires
- A wire used to signal "last eSlate"

One thing that is confusing here is that the network is superficially a daisy chain: wires come into an eSlate and then come out. It's natural to suspect that eSlate One forwards packets to eSlate Two. This is not the case. Rather, all the devices are electrically connected, it is simply that the connection takes place inside each eSlate rather than externally as is the case with, for instance, 10Base2 (Thinnet). Effectively, this is a pair of broadcast networks, with one network used for control and the other network used for data.

Access to the network is mediated by the JBC which serves as the network master. The JBC uses the control network to tell the other nodes when to read and write. Only the JBC writes to the control channel and other nodes write to the data channel only when told to over the control channel. Signaling on both channels is clocked at a nonstandard 938 kbps.

Note that the fact that the JBC is the network master means that no eSlate can ever say anything that's unsolicited. Since eSlates do sometimes need to initiate communication with the JBC—for instance to record CVRs—the JBC polls each eSlate periodically to see if it has anything to say. If it does, it sends it a query to get the data.

6.2.2 Tapping the Interface

Because this is a broadcast network, tapping it is straightforward. We were successfully able to connect to this interface using an Aaxeon MSC-102B dual-port EIA-485 card. Using a pinout supplied by Hart and supplemented by our own analysis of the interface we built a cable to bridge the HD-15 output of an eSlate to our own PCI EIA-485 card. The maximum rated clock speed of the card is at 921.6 kbps, but this is apparently within error limits because clocking our card at this rate enabled us to capture both the control and data channels. Because we were able to capture messages between the eSlate and the JBC while sniffing after the final eSlate in the chain, this experiment confirms that all nodes receive each message. Note that the cable from the final eSlate is exposed and therefore can be tapped without unplugging any devices.

[5]EIA-485 is a differentially signaled multipoint serial interface (hence the need for two wires).

6.2.3 Hijacking the Interface

Tapping the network allows the attacker to have access to the contents of messages flowing over it, but this only allows a small number of attacks. More attractive to the attacker is to *hijack* the network. I.e., we would like to:

- Pretend to be an extra eSlate
- Pretend to be an existing eSlate to the JBC
- Pretend to be the JBC to the eSlate

This would allow you to send messages and affect the behavior of other nodes, rather than just broadcasting.

The difficulty here is that because messages are broadcast, the attacker has to compete with whatever node he is impersonating. So, for instance, if he is pretending to be eSlate Two, he has to worry that messages that are coming back to him are *also* read by eSlate Two and that it gets confused by the unexpected messages. Even more difficult is that the the eSlate being impersonated is trying to transmit at the exact same time the attacker is, which causes conflicts.

There are two major approaches to attacking such a network.

Using a Proxy

Because what makes attacks difficult is that the network is broadcast, the natural approach is to remove that property.

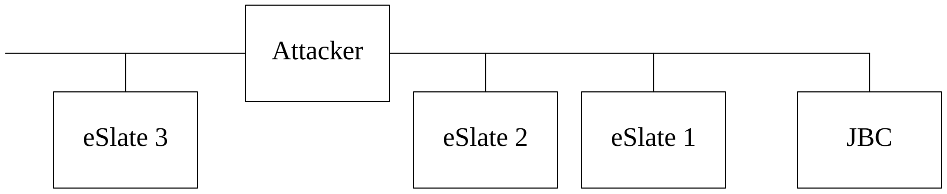

Figure 6.2: JBC/eSlate Network Proxy

In Figure 6.2, the attacker interposes himself between eSlates Two and Three and separates the network into two networks[6] He receives all messages directed to and from eSlate Three from and to the rest of the network and can pass on, change, or drop any messages he chooses. As far as the rest of the network is concerned he is eSlate Three and as far as eSlate Three is concerned, he is the rest of the network.

This allows complete control of all transmissions crossing the Attacker's device, but has two disadvantages from the perspective of the attacker. First, it cannot be mounted purely from a compromised eSlate. The attacker must leave a device in place for the entire duration of the period he wishes to mount his attack. The device could probably be fairly small (about the size of a matchbox) and could easily be put in place on the input port of an eSlate. Nevertheless, it might still be noticed.

Second, the attacker can only impersonate the JBC to devices to the left of his device (and vice versa). This limits his control and motivates placing the device between the JBC and eSlate One, which may be more noticeable since eSlate One is closest to the poll workers.

Installing such a proxy also requires disconnecting the cable leading into the eSlate, which might be noticeable. The Hart curbside voting feature appears to make this practical: once a voter code has been entered into an eSlate, if the JBC loses contact with that eSlate, it assumes that it is in curbside voting mode, logs the fact, and waits for its return. Even if the JBC does notice that a node has failed and disables it, the attacker can probably convince poll workers it was an innocent failure and have them correct it, possibly by rebooting the entire system. The Hart documentation [16]

[6]Technically, four networks, with two EIA-485 networks on each side.

describes how to clear such errors and does not treat them as security critical. Note that once the attacker has compromised the eSlate he can cause it to display any error message of his choice.

Sharing the Network

From attacker's perspective, a more attractive alternative would be to take over a single eSlate and then use it to impersonate other eSlates to the JBC and to each other. This avoids the need for any long-term device which might be detected. A related technique would be to plug into the dangling end of the cable from the final eSlate and pretend to be one of the existing nodes.

Technically, however, this is a more difficult problem. The most likely avenue would be to "co-master" the network, i.e., pretend to be the JBC. The general idea would be to issue one's own network control messages at times when the JBC was silent (most of the time). Analysis of the source code suggests that the JBC ignores control channel messages not directed to it, therefore it should be fairly oblivious to attempts to control the network. More difficult is stopping the JBC from receiving data messages from the eSlates directed to it (recall that we are impersonating the JBC) or data messages which we are using to impersonate it.

We believe that this can be dealt with by the attacker. First, the JBC appears only to pay attention to responses from nodes it expects to receive data from, so with careful timing it may be possible to minimize exceptions on the JBC (which is reasonably resilient in any case). Second, First, in many cases it is possible to send messages with a response address of 0, which means that there should be no response. In such cases there is no need to suppress the JBC from reading the responses. Third, the attacker could transmit a jamming signal in time with the response in order to generate a bogus message checksum, which will cause it to be ignored by the JBC. This works best with messages which we plan to ignore in any case but which might confuse other units.

Clearly, all of these techniques are more complicated than the simple proxy approach described in Section 6.2.3. The advantage is that they can be mounted from a compromised eSlate without any special hardware.

Issue 5: eSlate-JBC communication is insecure

The JBC and the eSlate are in constant communication. This communications channel is used for a number of purposes, including:

- Management of the eSlate (see Issue 2).
- Transmitting ballot information to the eSlate.
- Allowing the eSlate to check the validity of a voter access code.
- Transmitting CVRs to the JBC from the eSlate.

This channel is not cryptographically secured in any way, thus allowing an attacker who can access the interface to impersonate an eSlate to a JBC or vice-versa.

Detailed Description There is no cryptographic protection for messages on the eSlate-JBC network and therefore there is no authentication, message integrity, or confidentiality. Any attacker who can access the network can transmit any message he wishes on the interface and have it accepted as coming from any source address he chooses. He can also view any messages which are transmitted on the network.

In particular, an attacker who controlled this network could:

- Pretend to be a JBC and "accept" a CVR, thus preventing it from being recorded by the JBC.
- Pretend to be a JBC and lie to the eSlate about the validity of a voter code, thus allowing an illegitimate voter to vote or blocking a legitimate voter from voting.

- Pretend to be an eSlate and send spurious votes to the JBC (this is potentially enhanced by Issue 8).
- Watch each vote as it is cast.

As an example, consider the first attack. The JBC periodically polls every eSlate to see if it has a CVR to deliver. When it gets a positive response it sends a message to retrieve the CVR and then resets the eSlate's state.

An attacker who controlled the network could pretend to be the JBC and retrieve the vote, thus leaving the eSlate thinking it had delivered the vote while the true JBC has not in fact stored a copy. It could simultaneously pretend to be the eSlate to the JBC and store votes of its choice, replacing the real eSlate votes. The other attacks proceed in a similar fashion, with the attacker pretending to be one side or the other. Note that vote observation can be done purely passively, i.e., by snooping on the shared communication network. Any eSlate can observe all the communication from any other eSlate connected to the same JBC.

One difficulty here is that because this is a broadcast network, and therefore impersonating an existing node may cause problems for other nodes. Earlier in this section we outline some potential ways to bypass this obstacle. An attacker who is present at the beginning of the election can simply pretend to be a nonexistent eSlate, provided that fewer than 12 eSlates are in use. This avoids issues about hijacking for eSlate impersonation.

Prerequisites In order to exploit this issue the attacker would need:

- To have decoded the protocol used by Hart.
- Access to the serial port on the eSlate during an election.

As indicated in Section 6.1 we believe that an attacker who had an opportunity to observe a running system could decode substantial portions of the protocol.

Currently, any voter has access to the serial port on the eSlate and can easily remove the connector. The eSlate is designed to be disconnected to enable curbside voting and therefore it is possible to remove the last eSlate in the chain without generating alerts. In addition, an attacker could potentially connect to the outgoing cable on the last eSlate in the daisy chain, which is simply stored in a compartment in the eSlate stand.

The attacker has to have some sort of access to the serial port for the entire period of the attack. This does not imply that the attacker is physically present the whole time, merely that his device is attached. We anticipate that the device could be made relatively small, on the order of a matchbox. Nevertheless, this presents some risk of discovery.

Impact The impact of these attacks depends on which device is being impersonated. However, in general it would be possible to inject false votes into the JBC (which would not match the VVPAT or the eSlate memory), remove votes from the eSlate (but not from the VVPAT or JBC memory), or allow an attacker to appear to vote multiple times. This final attack would create matching entries in the VVPAT, eSlate memory, and JBC, though the logs of how many individuals had voted would not match the number of votes cast.

Mitigations One potential mitigation is to severely restrict access to the eSlate serial port. However, this is difficult for a number of reasons, including the desire to enable curbside voting. It may be possible to restrict voter access, but because the poll workers assemble the eSlate/JBC daisy chain at the polling place, it is likely to be extremely difficult to secure their access. Note that a small device placed between the eSlate serial cable and the connector is sufficient to mount this attack.

Another mitigation would be to cryptographically authenticate the eSlate to the JBC and vice versa. This could be done in a number of ways. One possibility is for the eSlate and JBC to use the global eCM key to authenticate each other. This has several difficulties. First, that eCM key needs to be installed in the eSlate, which currently would have to be done over this interface. However, it might be possible to do that installation securely in the warehouse and then set the eSlate to use cryptographic protocols from there on. Second, compromise of any unit in the entire county potentially leaks the eCM key and would allow this attack to proceed.

Another possibility would be use public key cryptography to authenticate the eSlate and JBC to each other. As with symmetric cryptography, one would securely install knowledge of the expected peer's public key on each device and they would refuse to connect to any other device. One difficulty here is that this severely reduces the flexibility to mix-and-match devices within and between precincts.

An additional problem with any cryptographic solution is that it requires some mechanism to reprogram the keying material on the device. That mechanism then becomes a potential target of attack and so must be designed carefully.

Status This issue was discovered by examination of the source code. We have verified that we can observe communications between the JBC and eSlate and were able to verify the lack of cryptographic security. We have not attempted to impersonate either side of the communication.

Issue 6: Format string vulnerabilities in JBC report mode

A format string vulnerability in the JBC's write-in summary report may allow an attacker who had caused the JBC to record specially-formatted votes to compromise the JBC after polls are closed.

Detailed Description At the end of voting, poll workers instruct the JBC to close the polls, after which no further votes are accepted. Once polls are closed, the JBC allows poll workers to produce, on the JBC's internal printer, any of three reports: a voter code summary; a vote tally; and a write-in report.

For each precinct, for each party, and for each contest, the write-in report lists the candidate names included in write-in fields and how many votes each received. The JBC produces this report by examining the votes in its internal CVR log.

While write-in candidate names are massaged, for example to remove leading and trailing whitespace, it appears that no attempt is made to filter out `printf` format specifiers (such as " `%d`"). What's more, the `PrintWriteIns` routine, which is called for printing each write-in entry in the summary, passes the candidate name to `printf`[7] as the format string, rather than using an idiom such as `printf("%s",str)`. This makes the JBC vulnerable to a format-string attack [13]: a specially crafted write-in candidate name recorded in the JBC CVR log, when processed as part of the write-in report, allows memory overwrites arbitrary code execution.

Prerequisites In order to exploit this issue the attacker would need:

- To cause a vote containing a specially crafted write-in candidate name to be recorded in the JBC's CVR log.
- To cause a poll worker to request a write-in report.

It is not normally possible to input format specifiers using the eSlate on-screen keyboard. An attacker could nevertheless cause votes containing the appropriate specifiers to be recorded either by compromising an eSlate (Issue 2) or by tapping the JBC-eSlate communication network and impersonating an eSlate (Issue 5).

[7]More precisely, to `pprintf`, which passes it to `vsnprintf`

Impact A JBC can only be compromised using this vulnerability once polls have closed. This is, of course, too late to have an effect on the system's behavior towards voters. Nonetheless, a compromised JBC can still do any of the following:

- modify its internal audit and CVR logs;
- modify the audit and CVR logs on the MBB, either to match the modified logs on the JBC or to exploit a vulnerability in Tally (cf. Issue 14);
- leak the cryptographic key (cf. Section 6.7); and
- use the JBC-eSlate interface to modify the logs on the eSlates or install new software on them (cf. Issue 2).

A compromised JBC can also be used to mount attacks on back-end systems, for example on SERVO (cf. Issue 13).

Mitigations Until Hart fixes this vulnerability, poll workers could disconnect eSlates from the JBC before requesting any JBC reports; this ensures that a JBC compromised through vulnerabilities related to its CVR parsing cannot modify the audit and CVR logs stored on the eSlates. This does not, however, mitigate the risk of SERVO being compromised through a compromised JBC (see Issue 13).

Status This issue was discovered by examination of the source code. We have not attempted to verify it.

Issue 7: The JBC access code generator is insecure

Voter codes, printed at the JBC, are intended to prevent voters from registering unauthorized votes at an eSlate. These codes are predictable: anyone who sees a single voter code can compute the sequence of all subsequent voter codes.

Detailed Description To prevent unauthorized voters from using an eSlate, Hart employs voter codes—four-digit pseudorandom numbers printed on the JBC that the voter types into the eSlate.

A voter enters his voter code on an eSlate to begin voting. The eSlate checks with the JBC that the code has been issued and is unused. If this is the case, the JBC marks the code as used, and the eSlate allows the voter to make his selections. Casting a ballot invalidates the voter code.

The algorithm that generates these voter codes uses a poor design that makes it possible to predict codes. The JBC picks a random initial index, between 0 and 9999. The JBC generates a voter code by applying an unkeyed permutation to the current index; it then increments the index.

Permutations, of course, can be inverted. Applying the inverse permutation to a voter code gives the current index. All subsequent voter codes can be determined by applying the permutation to the computed index plus one, plus two, and so on.

(This issue was first discovered by Proebstel et al. [26] in an analysis of Hart systems where they did not have access to any source code.)

Prerequisites In order to exploit this issue the attacker would need to observe a single voter code, for example by voting. Also, the attacker would need to know the permutation, which is fixed for the entire Hart system.

Impact An attacker could predict voter codes. Using these, he vote in place of another voter: He must wait for the other voter to be given a voter code from a poll worker. Before that voter reaches an eSlate and signs in using the code, the attacker signs in on another eSlates. The attacker can now vote as he pleases, whereas the legitimate voter will receive an error message indicating that his voter code is already in use.

Mitigations Poll workers should be vigilant for instances where newly issued codes appear to be already being used. These occurrences could be indicative of this attack or of voters who attempt to double-vote by using the same code twice. Poll workers should also be aware of voters who spend an unusually long time in the voting area.

Hart could revise the software running on JBCs to produce cryptographically unpredictable voter codes. A simple way to do this is as follows. Start an index at 0, and increment it with each vote. To produce a voter code, apply a *keyed* function to the index, where the key is chosen at random at startup. A good choice is to apply AES (with a randomly-chosen 128-bit key) to the index, and output the result modulo 10,000. The key should be generated using a source of strong randomness; the C library's `rand` function is insufficient.

Status This issue was discovered by examination of the source code. We have developed an access code predictor and verified it against a real access code sequence from a JBC, provided by Proebstel.

Issue 8: The JBC will accept votes from eSlates that are not in an authorized state

Voter codes, printed at the JBC, prevent voters from registering unauthorized votes at an eSlate. As currently engineered, they do not prevent a compromised, malicious eSlate from registering arbitrarily many votes without a voter code and without a voter present.

Detailed Description To prevent voters from registering unauthorized votes using an eSlate, Hart employs voter codes—four-digit pseudorandom numbers printed by the JBC that the voter types into the eSlate.

The eSlates send to the JBC their most recent voter code as part of each status message; status messages are solicited by the JBC once every second.

The JBC keeps a list of valid voter codes. When a voter types a voter code into an eSlate, the eSlate checks with the JBC that the code is valid by setting a flag in its status message request field, which prompts the JBC to verify that the code is valid and not yet assigned to assigned to another eSlate. The JBC then logs the code as assigned to the eSlate and communicates to the eSlate to proceed with voting.

Once the voter has finished voting, the eSlate signals that it has a vote ready by setting a second flag in its status message request field, which prompts the JBC to ask the eSlate for the CVR and to record the CVR in its CVR and audit logs. The JBC then removes the voter code associated with that eSlate (the code that was transmitted with that eSlate's most recent status message) from the active code list. If the code is not not found on the active code list, no error is raised, and the vote is still recorded.

The JBC has enough information in to determine (1) whether the voter code was generated by the JBC; (2) whether the voter code was already used by the eSlate voting using it; and (3) whether this is the first and only vote being recorded by this eSlate using this voter code. Our inspection of the source code suggests that the JBC does not perform any of these checks.

A compromised, malicious eSlate can thus record arbitrarily many votes, each with an arbitrary voter code. (Note that voter codes are included in the audit logs but not in CVR logs; however, the two logs can be correlated. See Issue 25.)

Prerequisites In order to exploit this issue the attacker would need to have subverted or be impersonating an eSlate.

Impact The subverted or fake eSlate can record arbitrarily many votes and thus engage in ballot-stuffing.

Mitigations Election officials can analyze the audit logs to verify that each generated voter code corresponds to at most one recorded vote, and that the eSlate to which the code was assigned is the one that records the vote. This analysis would allow officials to detect the attack, but it would be impossible for them to distinguish which of several votes recorded from a single eSlate using a single voter code is valid; see Section 7.2.2.

Hart could revise the software running on JBCs to automatically perform the checks recommended above, and to warn poll workers of attempts to register unauthorized votes.

Status This issue was discovered by examination of the source code. We have not attempted to verify it by impersonating an eSlate.

6.3 Software Integrity Checks

Hart appears to have been aware of the possibility of malicious firmware replacement and to have taken steps designed to prevent it. In particular, the following four mechanisms are used:

- The eSlate and the JBC both run internal memory consistency checks designed to detect changes in the running firmware.
- The JBC checks its own version.
- SERVO can be used to verify the firmware image on the JBC, eSlate, and eScan.
- The JBC checks the version of connected eSlates.

All of these mechanisms appear to function correctly for damaged firmware; however, they appear to be vulnerable to attack by *malicious* firmware.

Issue 9: eSlate/JBC internal CRC checks do not detect attacks

eSlate and the JBC run a "background task" whose job is to check memory integrity. This task starts with a list of memory regions and their expected CRC-16 values and compares the actual CRCs of the regions to the expected CRC. This checking is likely to be easily bypassed.

Detailed Description The eSlate and the JBC both run a background task which performs memory checking. It is provided with a list of code sections to check[8] and their CRCs, presumably computed during the binary build process. Every ten seconds it walks through the list and compares the CRCs of the regions to the expected CRCs.

We believe there are a number of avenues for bypassing or otherwise avoiding this check. These include:

- Inserting new code with the same CRC-16 value. This is easy to do if you have even a small amount of control over the instructions generated, as CRC-16 is not a secure hash function. Approximately 16 locations where there are two alternative instructions are enough.
- Overwriting the check list. This is likely to be possible in the 10-second interval between checks.
- Patching the code that runs the checks so that they are no longer run.
- Patching the firmware and section list in firmware and not in memory. Once the error is caught it causes a system crash. When the system restarts it will have the new firmware and section list, so it's not clear this can be distinguished from a simple crash.

We emphasize that we have not attempted any of these approaches because we did not have an opportunity to use a development kit to generate new firmware for the JBC or eSlate and only had limited access to both devices. However, these are standard techniques and seem likely to work.

[8]We do not have a list of those sections.

Prerequisites Any attacker who has the ability to load a new binary image on one of these devices could exploit this issue.

Impact This issue does not itself allow compromise of a system. However, it allows an attacker who has compromised a voting device to avoid detection. Hart's tamper detection routines are not themselves tamper resistant.

Mitigations The problem of having a program verify its own integrity is very difficult against a clever attacker. The "matching CRC" approach could be mitigated by using a superior hash such as SHA-1, but this would not prevent the other avenues of attack. A variety of technologies, such as Trusted Platform Module chips [25, 33], are now commercially available. Future versions of the Hart system could investigate the use of such hardware features. Likewise, Hart could investigate software attestation techniques such as PIONEER [30, 12].

With the present Hart software and hardware, there are no effective mitigations against this attack, beyond procedural measures aiming to limit an attacker's ability to install malicious software.

Status This issue was discovered by examination of the source code. We have not tested it on a running system.

Issue 10: JBC internal version checking is broken

The JBC has some internal version checking, but it is unclear how it is intended to work. It appears to only check two compiled-in numbers against each other and then log an error rather than exiting if there is a failure.

Detailed Description On bootup, the JBC attempts to verify its software version, ultimately fetching a a compiled-in value. This value is then logged and "success" is indicated as a return code. If failure were ever indicated, the system would exit, but since it cannot, this function always succeeds. Perhaps the implementation of this feature is incomplete.

Impact We don't understand the purpose of this set of checks. An attacker can compile in any version number they want and it's not clear that the audit logs are checked for version numbers in any case. However, whatever behavior this check is supposed to prevent, it seems unlikely that it does so.

Mitigations Unknown.

Status This issue was discovered by examination of the source code. We have not tested it on a running system to see if we misunderstand its function.

Issue 11: SERVO-based device firmware checking can be spoofed

SERVO can be used to verify the firmware of voting devices against an official firmware integrity file provided by Hart. This file contains SHA-1 digests of the firmware images. SERVO downloads the firmware images from the target device and then computes the digest and compares it with the recorded hash. If they do not match, SERVO signals an error. However, because the firmware image is provided by the running program, a malicious image can simply provide data that fools the check (e.g., by maintaining a backup copy of the legitimate firmware image for purposes of computing these SHA-1 digests).

Detailed Description SERVO maintains a database of all known devices owned by the county, partly for inventory control purposes. When a new device is introduced into the system (this is determined by the 32-bit device ID), it offers an option to "Verify" that the device contains the expected firmware image. Verification is done by comparing the hash of the device's actual firmware to the known-good hash. SERVO stores known-good hashes for each type of device in its database; these are loaded from an XML file supplied by Hart using SERVO's "Import Firmware Data File. . . " menu item.

The verification process is as follows:

1. Interrogate the device for its device ID.
2. If the device is already in the database, exit.
3. Retrieve the device version number and compare it to the expected version.
4. If the device isn't in the database, download its firmware. On the eScan this is done by reading the file `eScan.exe` using the `FILE_CMD_GET` command. On the JBC/eSlate this is done by using the `MEM_READ` command to read the appropriate memory block.
5. Once the entire firmware image is recovered, it is hashed with SHA-1[9] and the hash is compared to the expected value.

This technique has several problems. The first is that the firmware is only checked the first time a device is loaded. We do not know if the database is zeroed between elections, but if it is not then subsequent compromises will not be detected. Second, it is dependent on the device's providing the correct device ID. It is unclear what happens if the device provides the device ID of another, already checked, device.

These issues could presumably be dealt with by forcing a check every time. However, the more serious issue is that the commands that SERVO uses to download the firmware image are executed by the program running on the eScan, JBC, or eSlate. If that program has been compromised it can simply hand back the contents of a valid binary even though those do not reflect the program that is actually running. Another way to bypass this check would be to have the last act of the malicious image upon election close be to copy the correct image back onto the machine.

The current implementation of firmware checking also leaves SERVO vulnerable to compromise by a malicious eScan; see Issue 13.

Prerequisites Any attacker who has the ability to load a new binary image on one of these devices could exploit this issue.

Impact This issue does not itself allow compromise of a system (see, however, Issue 13). However, it allows an attacker who has compromised a voting device to avoid detection.

Mitigations The problem of verifying that a particular piece of software is running on a remote system (attestation) is known to be extremely difficult. The obstacle is exactly as seen here—you need to communicate with the system but you are doing so through a piece of software you do not yet trust. One commonly suggested strategy is to have a supervisor process which intercepts the communication and vouches for the binary. The supervisor itself is not field replaceable. This is difficult in this setting for two reasons.

First, the JBC and eSlate have no such supervisor process so implementing this approach would require very significant rearchitecture. eScan runs Windows CE and so it might be possible to install such a supervisor. However, the history of operating system security suggests that once attackers are able to run code at all on a target system they are typically able to escalate their privileges to supervisor status.

[9]The file format allows some flexibility in this, but SHA-1 appears to be what's used.

Second, this does not solve the problem of an attacker's completely replacing all the software on the machine, including the supervisor. Depending on the architecture, this might involve hardware hacking but is generally not impossible. In the limit, the attacker can simply gut the machine, install his own processor, and run a virtualized version of the Hart software.

Another suggested approach would be to move the firmware checking to the device by restricting it to loading only trusted firmware. This would require superior memory checking to that used by Hart (see Issue 9), which may or may not be practical. It would also not provide resistance to complete replacement attacks. The history of attempts to build restricted loaders is not particularly good [32].

The only known workable strategies for remote attestation involve trusted hardware on the target system (see Issue 9). That hardware can verify the running memory image and cryptographically signs the results for consumption by the verifier. This approach would require extensive modification of Hart's devices, both from a hardware perspective and in order to introduce the cryptographic infrastructure used for verification.

An alternative to remote attestation is local attestation, by having SERVO read the firmware of the target device directly rather than through the device's software stack. This would either require interfacing the device non-volatile memory to a new external port (a potential source of security holes) or physically removing the non-volatile memory in the warehouse and verifying it directly. The Hart devices do not appear to be designed to make either approach easy.

Status This issue was discovered by examination of the source code. We have developed a tool which runs on an ordinary Linux PC and will accept a connection from SERVO, pretend to be an eScan, and return a binary taken from a real eScan. This binary produces the same hash as that provided by a real eScan. We have not run through the entire backup and reset process because this would involve writing handlers for other eScan commands. However, we believe this is practical to do given modestly more time.

Issue 12: JBC-based eSlate firmware checking can be spoofed

The JBC attempts to check the integrity of the eSlate firmware. As with Issue 11, this check depends on the target device's telling the truth, whereas a compromised device can and will lie.

Detailed Description When a JBC connects to an eSlate, it gets the eSlate device information. The response contains the eSlate's software version and the "private ID" which appears to be some sort of CRC value. It then compares the major version of the eSlate software (ignoring the minor version) against its own version, which is hardcoded. If that matches, the eSlate is judged to be acceptable. Because this value is under the control of the eSlate software, this may be useful as a compatibility check but is useless against a malicious eSlate. In addition, this check is only done at startup, so an eSlate compromised during an election would go undetected.

Prerequisites Any attacker who has the ability to load a new binary image on one of these devices could exploit this issue.

Impact This issue does not itself allow compromise of a system . However, it allows an attacker who has compromised a voting device to avoid detection.

Mitigation See the mitigation discussion for Issue 11.

Status This issue was discovered by examination of the source code. We have not tested it on a running system.

6.4 Buffer Management Vulnerabilities in Back-End Systems

The back-end election management systems (BOSS, SERVO, Tally, Ballot Now) are programs running on standard Windows-based PCs. It is expected that physical access to these systems will be restricted to authorized personnel. Despite this, there still remain several avenues through which attacks can be mounted, including:

MBBs transported from polling places are connected to back-end election systems. These MBBs could have been tampered with, as described in Section 6.8.

Polling place equipment could be connected to the SERVO system post-election. This equipment could have been tampered with (as described in Section 6.1).

Connections from a Tally system to a Rally system could be made for the purposes of transferring election data. If the Rally system were compromised, it could be used to propagate attacks to Tally. (See also Sections 6.9 and 6.6.)

Malicious use of election management systems by authorized personnel.

As a result, great care should be taken to secure the back-end systems at the points where they interact with other election systems.

Issue 13: Multiple buffer overflows in SERVO

After an election, SERVO is used by election officials for verification and backup of the polling place devices. In particular, it is used to:

- Verify the firmware on the JBC, eSlates, and eScan
- Back up the Cast Vote Record logs from the JBC, eSlates, and eScan
- Back up the audit logs from the JBC, eSlates, and eScan

Each of these routines contains buffer overflows which we believe to be exploitable, allowing an attacker who has compromised the connected device to execute arbitrary code on the machine running SERVO.

Detailed Description All of these overflows follow roughly the same pattern: SERVO allocates a buffer of a given size to read data from the device. It then reads a block of data from the device but allows the device to specify the size of the data, without checking whether it will fit in the buffer. This allows an overflow which corrupts the `malloc` arena, which can be exploited using known techniques [23, 2].

As a specific example, consider an overflow in the firmware verification routine. SERVO invokes `FILE_CMD_GET` to read 1000 byte extents of the eScan executable[10] (stored as a file on the eScan flash memory) into a local buffer. However, it allows the eScan to tell it the length of the returned buffer (likely in order to detect the end of file by a short read). The eScan can cause an overflow of the buffer by returning a length greater than 1000. The buffer is allocated on the heap and is promptly freed, which is easily exploitable due to the details of the `malloc` implementation used.

This particular attack appears to work only with the eScan because a different command is used to read the eSlate and JBC firmware; however, the CVR log and audit log backup issues appear to apply to all three devices.

We are aware of other potential issues that appear to be somewhat harder to exploit.

[10]Strangely, the comment for this size definition reads `16k bytes`.

Prerequisites In order to exploit this issue, an attacker needs to take control of a device which will later be verified or backed up with SERVO. We have already described a number of issues which should allow this (see Section 6.1).

Note that the exploit described above requires that the device appear to be a "new" device in order to activate the verification routine. However, as the eScan can provide a device ID of its choosing, this should be easy to do even with a pre-verified device. The other routines do not suffer from this limitation.

Note also that while running the verification and backup procedures is not required for correct functioning of the Hart system, it appears to be Hart's recommended procedure and checking the "verify" checkbox is simply treated as part of the backup procedure in the SERVO documentation [19].

Impact The immediate impact of this attack is to allow the attacker to run arbitrary code in the SERVO process. This will generally also allow the attacker to run a program on the SERVO machine as the SERVO user, as well as to modify SERVO as desired. It is not necessary to escalate to Administrator privileges; nevertheless, techniques for doing so are well-known.

For the long-term impact of this attack see Section 7.4.

Mitigations These attacks could be directly mitigated by fixing the specific overflows. However, the fact that we were able to readily discover three simple overflows in a very small section of the SERVO code base suggests that SERVO is insufficiently careful about checking the input it receives from devices it is supposed to be checking. It would not be surprising to discover other remotely exploitable vulnerabilities of the same type. See Sections 5.3 and 6.11 for more on this point.

Status These issues were discovered by examination of the source code. We have directly exploited the first of the three attacks (through the firmware verification routines) against our own copy of SERVO installed on Windows 2000 SP4 in a VMWare image running on one of our Linux machines. Our exploit installed a Windows "bind shell" that we were able to remotely access. We have not yet attempted to exploit it against a Hart-supplied machine.

Issue 14: An improperly formatted MBB will cause Rally or Tally to crash.

A mobile ballot box (MBB) can be carefully crafted to cause the Rally or Tally applications to crash when an MBB is read into the system. If a compromised Rally feeds this information to Tally, Tally will likewise crash.

Detailed Description The MBB file format makes extensive use of 16-bit CRCs, which serve to detect minor corruption within the file. (16-bit CRCs do not serve any security function to protect against tampering. The MBB format separately uses HMAC for that purpose, as describe in Section 6.8.) The low-level routine that verifies these CRCs uses a 32-bit length field that comes from the header of the MBB. While the true length of the buffer might be quite small, the header of the MBB can specify a much larger buffer. Such a malformed MBB will cause the `crc16` routine to read memory beyond the end of the buffer.

Given a large-enough length field, `crc16` will eventually attempt to read a virtual memory address where no physical page has been mapped. This will result in a segmentation fault, causing Tally to crash. Every time Tally is restarted and asked to process the MBBs, it will consistently crash.

Impact This attack will not destroy any data, but it can cause either Rally to Tally to crash when reading an MBB. If a Rally machine has been compromised through other means, then the communications path between Rally and Tally would allow for Rally to deliver a corrupt MBB to Tally and cause Tally to crash.

While the corrupt MBB would never be written to the database on disk, this could cause a fair amount of confusion for the election administrators.

Prerequisites An attacker must be able to introduce a corrupt MBB into the legitimate flow of MBBs from the precincts back to Election Central. This attack could be performed by a malicious poll worker (or group of poll workers, depending on local procedures). This attack could also be performed if any individual eScan, JBC, or Rally server had been corrupted at some point in the process, as all of them ultimately feed MBBs to the Tally server.

Mitigations An attack of this form might not corrupt *every* MBB but only a subset of them. Election officials would be able to determine a specific MBB (or set of MBBs) which cause the crashing condition. This would allow the election officials to tally the non-corrupt subset of MBBs; votes from the effected precincts would need to be hand-tallied from paper records.

Future versions of the Hart software could be engineered to more carefully track the true size of any given buffer, rather than trusting the length fields in potentially untrusted data to be truthful (see Section 5.3 and Issue 36 for more details).

Status This issue was discovered by examination of the source code provided to the code analysis team. It has not been directly verified against Rally and Tally servers.

6.5 Privilege Issues in Back-end Systems

All of the Hart back-end software runs on a general-purpose operating system (Windows) with its own security mechanisms intended to force users to only use the official applications and block direct access to the underlying data, even by an election administrator. However, in many cases we find that the Hart applications are designed in such a way that allows such direct access. A particular point of concern is the security of the databases that store much of the data on the Hart system.

Issue 15: Database passwords are stored insecurely

The user names and passwords used by the back-end election management systems to access databases containing election data are stored in configuration files. The configuration files, as a whole, are not protected using any cryptographic techniques. Within the files, the password is obfuscated, but the method of obfuscation is easily inverted to recover the password.

Detailed Description When one of these systems (Boss, SERVO, Ballot Now, Tally) is started, a user name and password is read from a configuration file and used to connect to the database server. The password value contained in the configuration file is the characters of the original password XOR'd with the character 'x'. Anyone who has access to the configuration file can easily recover the database password and use it to connect directly to the database server containing election data, reading any data present, as well as making arbitrary changes to the database.

Prerequisites Any attacker who has access to the file system of the PC running a back-end election management system could exploit this issue.

Impact Since the username and password contained in this configuration file are used by the election management system code to read, modify, and delete election data, possession of this password configuration file allows the same privileges.

Mitigations One of the most common reasons to store passwords in a file is to avoid requiring user input upon program startup or having the user type multiple passwords. If this is a necessary product feature, then there are a number of ways to partially mitigate the risk of password exposure. One of the most common is requiring more read and write permissions on the configuration file than the user logged into the PC owns. The election management applications are then configured to Run As[11] a user that has the appropriate read and write permissions on the configuration files. The reason that this can only be considered a partial mitigation is that if the normal user is able to escalate his privileges, then the passwords can be recovered. Alternatively, an administrative user could still recover the database password. See Section 6.6 for more issues related to Windows security.

A more aggressive approach would be to have the database encrypted under a password that the user typed in, avoiding the entire issue of file access. This password can be integrated with the general Hart login system to avoid multiple password management issues.

Status We discovered this issue by examining the source code. We then developed tooling to extract the password and were able to use the password to connect directly to the database.

Issue 16: Ballot Now counters are stored in database

The Ballot Now system is designed to maintain a private counter of all ballots processed by a given Ballot Now installation as well as a public counter of the number of ballots written to a particular MBB. When Ballot Now is installed, the private counter is expected to be zero. Further, only the Ballot Now system is supposed to have the ability to increment the private counter. We found that the ability to access and modify the Ballot Now database suffices for changing the values of the public and private counters.

Detailed Description The private and private counters are implemented as database table entries in the database used by Ballot Now. Anyone who can connect to the Ballot Now database with the privilege to update the table containing the private counter can change the counter value. As described above, the username and password of an account that has such privileges is stored on the file system of the PC running Ballot Now.

Prerequisites Any attacker who has access to the Ballot Now database could exploit this issue.

Impact Modifying the public or private counters of a Ballot Now system could bring into question the number of ballots that it processed, either in a given batch of MBBs or for its lifetime, potentially requiring manual examination of election data to resolve discrepancies.

Mitigations This problem can be partially mitigated by requiring more read and write permissions on the counter files (and the database files) than the user logged into the PC possesses. Ballot Now would then be configured to Run As[12] a user that has the appropriate read and write permissions on the counter files. The reason that this can only be considered a partial mitigation is that if the normal user is able to escalate his privileges, then the counters can be modified. Alternatively, an administrative user could still modify the counters. See Section 6.6 for more issues related to Windows security.

Future Hart software could be designed to leverage the hardware counters featured in TPM chips [33] that are increasingly installed in standard PCs. This would allow counters to be stored in a tamper-resistant chip rather than on the accessible file system.

Status We discovered this issue by examining the source code.

Issue 17: The Tally interface allows a Tally administrator to "adjust vote totals." This can create inconsistencies in the reported vote totals.

[11] `http://support.microsoft.com/kb/294676`
[12] `http://support.microsoft.com/kb/294676`

Detailed Description In the course of running an election, there may be a variety of reasons for an election administrator to need to adjust the totals being reported by Tally. One reason, cited in the Tally User Manual, is that there may be an additional voting system in use, entirely outside of Hart's voting system, whose votes need to be included in the final tally.

The "adjust votes" feature allows the election administrator to select a specific precinct, race, and/or party (for primary elections), and to make changes to the bottom-line totals for that particular race. The implementation of this feature just changes the relevant "total" fields in the underlying database without adding any corresponding records that the adjustments had been made. This could result in vote totals inconsistent with the number of cast votes. This also does not allow an administrator to easily undo or modify such changes after they have been performed. If an election administrator adjusts the vote totals then realizes there was an error, the original adjustments can only be found in the audit log. The "adjust votes" dialog box will not show previous changes, nor is there an easy way to back out changes after they have been made.

In addition to the opportunities for an administrator to make legitimate mistakes, this functionality could also be used to tamper with the election results.

Prerequisite The user of the "adjust votes" dialog must have "administrator" privileges on the Tally application. Users may either be "administrators" or "operators" and operators are not allowed to adjust vote totals. Any user who either knows or can guess the administrator username and password and had brief physical access to the Tally machine could perform the attack.

Impact The reported vote totals could be inconsistent with the genuine cast ballot totals. The vote total adjustments are included in the audit logs, which would allow these adjustments to be observed.

Mitigations Users of Hart election systems could be discouraged from using the "adjust vote totals" feature of Tally.

Tally offers a variety of report formats, including some which are intended to be easy to load into tools like Microsoft Excel. California could require that a standard set of reports, including a complete report of the audit log, be produced in machine- and human-readable formats. This would simplify the process of detecting abuses of this feature.

In future software releases, Hart could change this feature to follow basic accounting principles, explicitly reporting the adjustments alongside the totals rather than simply overwriting the totals.

Status This issue was discovered by examination of the source code and verified by the Red Team on actual Tally systems.

Issue 18: Databases are not encrypted

Election-related data, stored in the databases used by these back-end election management systems, is stored in plaintext. As a result, election data can be read off without knowing the database access password.

Detailed Description The election-related data stored by these systems (Boss, SERVO, Ballot Now, Tally) are stored in Sybase databases. These databases make use of the file system for persistent storage. Even though logging into the database and issuing SQL commands requires knowledge of the database password, the database entries can be recovered by simply opening the database file directly with a binary editing tool. With a little extra effort, the database entries themselves can be altered by directly modifying the database file.

Prerequisites Any attacker who has access to the file system of the PC running a back-end election management system could exploit this issue.

Impact Sensitive election data can be obtained or potentially modified.

Mitigations One natural mitigation is to employ a database encryption scheme. Alternatively, a partial mitigation would be to restrict read and write access to the database file, in a fashion similar to that described in Issue 15.

Status We discovered this issue by examining the source code. The absence of encryption was verified by examining database files generated during a simulated election.

Issue 19: New users can be added via the database

An operator of one of the back-end election management systems, who has restricted access, can create a new account for herself with higher privileges. In general, being able to access the databases for one of these systems permits the creation of new user accounts.

Detailed Description The distinction needs to be drawn between the password used for logging into a system database versus the passwords used for logging into a Hart application. The database password, in addition to being stored within the database, is also stored by the application (see Issue 15). The applications also maintain *user passwords*, for users of the applications. These passwords are also stored inside the database and, unlike the database access passwords, are not stored in the clear but are instead hashed with a random salt (a fairly standard technique, used by Unix and many other systems; an attacker who can read the salted/hashed passwords will not be easily able to learn the passwords).

While an attacker cannot directly read the plaintext passwords, the attacker can either reuse existing passwords or compute new ones if he has the ability to write to the file. Likewise, an attacker can program a computer to methodically try common passwords (e.g., words in dictionaries) against the hashed passwords in the database. If a user chose or was given a weak password, the attacker could discover this.

Prerequisites Any attacker who has read-only access to the file system of the PC running a back-end election management system could extract the password file and reverse-engineer poorly chosen passwords. An attacker who has read-write access to the file system (particularly the database) could install his own users and give them administrative privileges.

Impact Privilege escalation for operators of back-end election management systems.

Mitigations As with other database-related issues, the root issue is that normal users will have sufficient privileges to access the database without going through the Hart applications. The same mitigations, as described above, apply here.

Status We discovered this issue by examining the source code. We manually started up the Tally database server, logged in, and issued SQL commands to add new Tally users with the same password as existing users. We were then able to successfully log in using these accounts.

6.6 Windows-related Vulnerabilities

Many of the components of the Hart election system run on standard Windows-based PCs, including BOSS, Servo, Rally, Tally, and Ballot Now.

The Hart documentation provides only minimal guidance in how their Windows installations should be configured. There are a wide variety of security issues that occur with any use of Windows machines. This section focuses on issues that may apply with Windows as used in a typical Hart customer installation.

Issue 20: Back-end Windows systems may be insecure

Because all of the Hart back-end components run on Windows-based PCs, an attacker who is able to subvert Windows may be able to subvert the Hart components. The user need not start out with administrator level access because standard configurations of Windows-based computers often are susceptible to privilege escalation attacks, especially if they have not been locked down or kept up to date. Therefore, the security of the Windows environment is critical.

Detailed Description Hart provides only minimal guidance for securing these machines. They do not provide detailed instructions on how to lock them down. Moreover, they encourage or require practices that are likely to compromise security. Section 3.5 of the Hart use procedures [21] state:

> Operating system upgrades to the computers on which the Hart Voting System applications are only performed by Hart InterCivic personnel, and only then, after certification by the Secretary of State.

We are unsure if this is a certification requirement, but in any case, unpatched versions of Windows are generally insecure and become increasingly so as new vulnerabilities are discovered.

It is likely that an attacker with even short-term physical access to one of the back-office machines could subvert it and escalate to administrator privileges. This would not be a long process since he could load malware on a USB stick, set it off, and then leave.

Prerequisites An attacker needs physical access to one of the Windows-based computers used in Election Central for at most a minute. The attacker could use a username and password to log into the Windows machine; or obtain access to a machine that is already logged in, perhaps left alone; or, with more time, access the chassis and access the machine's hardware directly.

Impact Election insiders with physical access to any of the Windows-based back-end machines could most likely subvert them.

Mitigation First and foremost, all Windows systems should be regularly updated with security patches from Microsoft.[13] This will reduce exposures to widely understood vulnerabilities.

Another important mitigation is to "lock down" Windows so that the user accounts used to operate the Hart systems have limited privileges rather than the full privileges of the Windows administrator. A suitably-configured user account would be unable to run *any* programs beyond the official ones necessary to operate the Hart software. Microsoft offers some advice for how to best configure Windows XP with least-privilege user accounts[14] and has added a variety of features along these lines to Windows Vista. For Windows 2000, which seems to be common for Hart servers, Microsoft has extensive resources online to describe how to lock down such a machine[15]. There are also good guides elsewhere online[16]. This will not necessarily stop privilege escalation but may delay it.

Finally, physical access to these machines should be highly limited. It should not be assumed, merely because a user does not have a login to a machine, that it is secure in his presence.

Status This issue was discovered by examination of the Hart manuals. Red team analysis of the systems provided by Hart determined that they were running Windows 2000 SP4, with the "Licensed User" granted full administrative privileges.

[13] Installing updates to Windows and to anti-virus systems may have implications for the certification process, which typically considers a specific version of every element of the software stack, including COTS components.

[14] `http://technet.microsoft.com/en-us/library/bb456992.aspx`

[15] `http://www.microsoft.com/technet/security/prodtech/windows2000/secwin2k/default.mspx`

[16] See, e.g., `http://www.cites.uiuc.edu/security/by_os/win2000.html`

Issue 21: Many Hart systems are connected to internal networks or modems, opening them to attacks against Windows' vulnerabilities.

Detailed Information Windows systems, particularly those which do not have Microsoft's security patches regularly installed, are well-known for having network-exploitable security holes. While none of the Hart Windows-based systems should ever be directly connected to the Internet, these vulnerabilities may still be exploitable.

An attacker wishing to exploit one of these vulnerabilities needs to connect his own computer to the same network as one of the Windows machines running Hart software. The attack would only take seconds to perform, perhaps installing a "backdoor" account on the Windows machines, allowing them to be accessed later in an arbitrary fashion. Such attacks would likely override any locking down of user account privileges (as described above), since many of the vulnerable Windows system services run with full administrative privileges.

An attack such as this could be mounted against Rally (when used at Election Central), Tally, BOSS, Servo, and eScan machines. In addition to launching an attack from an unofficial laptop computer, the attacker could possibly have compromised an eScan machine while it was in the field; eScan machines are connected to the local network at Election Central to extract their votes, providing an opportunity for the eScan machine to mount attacks against other machines on the same network.

When Rally and Tally communicate using modems (see also, Section 6.9), an attacker can also dial the phone number used by the Rally machine. If the Rally machine is configured in the fashion specified in the *Rally User's Manual* [18], then no username or password will be required before the Rally machine gives the attacker a "local" network address (i.e., $192.168.x.x$) allowing the attacker to communicate with *any* service on the Rally machine, not just the Rally application itself. This means that any network-exploitable vulnerability in Windows is now a modem-exploitable vulnerability in a Rally machine. When Tally connects to Rally, a compromised Rally can then attack the Tally system, which in turn can attack other Windows-based systems at Election Central.

Hart does recommend that anti-virus software software be installed on its Windows-based systems (see the *Hart Product Description* [17], pages 27-30), however such software typically requires an active Internet connection to download the latest updates, as does Microsoft's own Windows Update system. As Hart's Windows-based systems should never be connected to the Internet, this presents some practical complications to keeping both the anti-virus software and the necessary Windows security patches up to date. Hart's *System 6.2 Use Procedures* [21] (Sections 10.2.3 and 10.2.4) states that "anti-virus software is only installed and configured by Hart InterCivic personnel" and that "installation of software and firmware upgrades is performed only by Hart InterCivic personnel if or when necessary." This requires Hart's customers to rely on the vendor for a variety of services. As a major election approaches, the vendor may be unable to provide this service to all its customers in a timely manner.

Furthermore, most anti-virus software focuses on searching for well-known viruses ("signature matching"); security compromising software, engineered specifically to attack Hart systems, would most likely not match the signature of any known virus. Hart suggests that such updates should be performed through removable media [21] (Section 10.2.3).

Prerequisites An attacker must either have access with a laptop computer to the local network at Election Central for at most a minute, or the attacker must have previously compromised any one computer used in the local network, including the eScan machine. Once one machine is infected, the infection can spread to every other machine without the direct involvement of the attacker. Infections from traditional computer worms and viruses might also spread, inadvertently, in the same fashion, for example, if an election administrator's personal, Internet-using, laptop was accidentally connected to the internal network at Election Central.

Impact Once any one Windows-based system in a Hart election, including an eScan system, has been compromised in any fashion, it can then use well-known Windows vulnerabilities to attack all of the other networked Hart systems. Such attacks would have full system privileges, allowing them to modify or delete votes and other records. Likewise, the attack can spread either directly from the initial compromised machine, or it might spread via viral propagation.

Mitigations Windows XP and Vista have a built-in firewall tool.[17] This should be configured to block all ports except those required for the Hart system to function. For example, Rally and Tally machines communicate on port 4500. No other ports should be necessary, although this will require careful testing to determine whether there are unexpected dependencies. Regardless, the firewall can be configured to allow connections on only this handful of ports while dropping all other traffic.

Hart can and should develop a step-by-step checklist for installing its systems on Windows and for configuring Windows to minimize exposures to network-based vulnerabilities. This could be applied to the current, certified versions of Hart software. Likewise, the California Secretary of State's office could create such a Windows checklist and require Hart customers in the state to follow it. (A complete guide to Windows configuration is beyond the scope of this report.) The checklist should be mandatory, and must name the specific version(s) of Windows to which it applies.

A more aggressive mitigation would be to place each Hart system on a separate network, with airgaps between them. This would significantly reduce the risk that compromise of one system would affect others. This would be useful against attacks even if the machines were in the same room, since it would make viral spread significantly more difficult.

The modem vulnerabilities with Rally systems can be mitigated by banning the use of modems. If regional vote processing centers are still desirable, Rally can still be used in a disconnected fashion, with a traditional courier transporting the Rally machines back to Election Central once all of the precincts' votes have been collected.

eScan runs Windows CE, an embedded version of the Windows operating system. This makes it difficult or impossible for an end-user to change the configuration of eScan to equivalently configure a firewall to disable connections to unnecessary services. When an eScan is in the field, its Ethernet interface already provides extensive capabilities to an attacker (see Section 6.1), and the mitigations taken to protect against those attacks would also apply against these ones. It is unclear that there is any reason for this port to be live in the field at all.

In future versions of their software, Hart could design the installers for its various software packages to lock down the Windows computer, configure the firewall, and delete unnecessary and unused elements of the Windows system, making the above checklist an automatic aspect of the software installation. For Rally's modem issue, Hart could modify the way that modems are used such that dialing into the Rally system yields a direct connection to the Rally software rather than a general-purpose network connection to the Rally machine (i.e., Rally's design should follow the principle of least privilege to reduce the power of an attacker who might connect via the modem).

Status This issue was discovered by examination of the source code and other documents provided to the source code analysis team. Red Team analysis has determined that a number of unnecessary network services are enabled on the Windows 2000 machines provided by Hart. The Red Team did not discover any vulnerabilities in these services, implying that the machines given by Hart had all of the latest security patches installed. This might or might not be the case with Hart's customers using these systems.

Microsoft currently only provides "extended support" to Windows 2000, meaning they are providing security fixes but are not adding new functionality. Microsoft has stated that this extended support will end in 2010. Hart customers using Windows 2000 must migrate to newer versions Microsoft before Microsoft ceases maintenance of Windows 2000 security patches.

[17] Windows 2000 also has firewall functionality. See, e.g., `http://homepages.wmich.edu/~mchugha/w2kfirewall.htm`

6.7 Cryptographic Key Management

Hart makes extensive use of cryptography to protect data on their MBBs. In particular, ballot data and CVR data are both cryptographically integrity protected. Although the Hart documentation and source as well as the Symantec report [4] refer to this as a "signature" it is actually a symmetric *Message Authentication Code* (MAC), specifically HMAC-SHA1 [5]. Throughout this document we will refer to this as a MAC rather than a signature.

Because MACs are a symmetric technique, the generator and the verifier of a MAC must share a MAC key. This implies that if three parties, Alice, Bob, and Carol, all wish to use the same MAC key, then there is no cryptographic way for Alice to distinguish a MAC generated by Bob from one generated by Carol. Bob can impersonate Carol to Alice, and indeed any member of the group can impersonate any other. In Hart's system, a single MAC key is used for every machine/device in a single administrative domain, and so compromise of any entity allows the attacker to impersonate any other entity.

This key is distributed on *eSlate Cryptographic Modules* (eCMs) and is generated by the eCM Manager application. The eCMs are USB-based Spyrus cryptographic tokens about the size of a typical USB flash drive. The eCM manager uses the Windows cryptographic random number generator to create a secret 128-bit key and a *Globally Unique Identifier* (GUID) identifying that key. The key is then stored on each individual eCM. These eCMs are then used to program the JBCs and eScans using SERVO.

It should be noted that standard practice for the use of cryptographic modules is that the keys should be generated on the module and never leave it. Hart violates this practice in two respects, first by generating the key outside the module and second by exporting the key from the module. This leads to a number of issues, as detailed below.

Issue 22: The same symmetric eCM key is used county-wide

In the Hart system, all message integrity is performed using a single county-wide MAC key. This key is made available to the back-office applications such as Tally and SERVO by plugging an eCM containing the key into the machines running them. It is made available to JBCs and eScans by programming them using SERVO. This key does not appear to be used by eSlates. This practice implies that an attacker who compromises any single keyed device (most likely a JBC or an eScan) or has access to an eCM and its PIN will be able to forge MBBs that will be accepted by any other device in the system.

Detailed Description There are three major ways to recover the secret key

- Extract it from the eCM directly (which probably requires the eCM PIN).
- Extract it from the eCM manager (see Issue 23).
- Extract it from an eScan or JBC (see Issue 24).

Once an attacker has the key, he can mount a number of attacks. The two most interesting are: forging incorrect ballot information on MBBs that are consumed by eScans and JBC/eSlates, and forging MBBs containing fake votes (or modifying the votes on real MBBs) and sending them back to the central office for tallying.

Note that although Hart's procedures indicate that one should not reuse keys between elections, we did not find any technical controls preventing it.

Prerequisites A voter is unlikely to be able to access the secret key. A poll worker would be able to extract it (see Issue 24) if he had temporary unattended access to an eScan or JBC. An election official responsible for running SERVO or Tally could extract the key directly from the eCM. Though a PIN is required to access the eCM, this PIN also allows extraction of the eCM secret key, because extraction is necessary to export it to the eScan/JBC.

The attacker would also need physical access to the target MBB. It appears that the poll workers have such access at least in some environments and that election officials generally do.

Impact An attacker who forged MBBs for consumption by the eScan/JBC would be able to impose a ballot of his choice. This might allow him to affect election results, for instance by removing candidates, flipping the order of candidates, or breaking the binding between candidates and opscan markings or vote records. It is unclear whether this would be detected by those familiar with the correct order of the election.

An attacker who forged MBBs for consumption by Tally would be able to modify votes for that MBB. Those votes would be accepted by Tally. This attack would be detected if a direct audit (via SERVO) were performed or a manual paper recount—such as is done with the one percent recount—were performed.

Mitigations A number of mitigations to this threat are possible. In some cases forged ballot data could be detected by visually comparing the election interface to the expected UI. Forged MBBs sent to Tally could be detected by doing a complete recount based on the onboard memory of the JBCs, eSlates, and eScans. We do not know if this is standard practice. More extensive use of paper-based recounts would also aid in detection of this type of attack.

Restricting master keys to a single election, as Hart's procedures require (though the software does not appear to enforce) would reduce the risk level somewhat. More severe restrictions on access to the secret keys, such as those proposed as mitigations to Issues 23 and 24, would make this attack more difficult to mount.

Moving from a MAC-based system to a digital signature-based system with separate keys for every device would significantly mitigate this vulnerability by separating the ability to verify from the ability to sign. The effect would be that compromise of an eScan or JBC would no longer allow forgery of MBBs that would be accepted by other eScans/JBCs. Similarly, it would not allow forgery of MBBs that were accepted as from other eScans/JBCs, provided that Issue 27 were also addressed.

Status This issue was discovered by examination of the source code. We have developed a tool that will check signatures on MBBs using an extracted key. We have made some initial attempts to forge MBBs but do not yet have a working tool; due to time constraints have not yet been able to complete this.

Issue 23: ECM keys are stored insecurely on the eCM manager

The eCM manager creates a key in software using the Windows `CryptGenRandom` call, which is a standard Windows call and appears to generate high quality randomness. The generated key is 128 bits long, which is adequate. The eCM offers the user the option to save the module data to a file. When that data is saved it is saved "obfuscated," which means that each byte of the data is XOR'ed with the letter 'x'. An attacker who controls the computer on which this has been done can trivially extract the secret key.

Detailed Description An attacker who has access to the computer running eCM manager can simply find the `.eCM` file and XOR the contents with 'x' to recover the key.

Prerequisites The attacker simply needs access to the computer.

Impact Once the attacker has access to the eCM key he can mount any of the attacks described in Issue 22.

Mitigations One mitigation would simply be not to store the MAC key to a file at all. This would require initializing all the eCMs at once or having eCMs with different keys, which would be necessary in any case if public key cryptography were being used.

Another mitigation would be to replace the obfuscation with encryption under a user-supplied PIN or passphrase [24]. This would still potentially be susceptible to dictionary attacks if the attacker *also* had access to an MBB protected with that key or if the encryption were done poorly. However, this is a significantly more difficult attack, especially if a good passphrase is chosen.

Status This issue was discovered by examination of the source code. We have developed a tool to extract the key from an .eCM file and have used it to verify an MBB.

Issue 24: eCM keys are extracted and stored insecurely

Although the keys are transported on the eCM, they are then copied (over the management channel) to the JBC and the eScan, where they are stored on the onboard memory. Any attacker who controlled either of these devices would be able to extract the keys.

Detailed Description SERVO is used to program MAC keys onto the eScans and JBCs via the usual management interface (see Sections 6.1 and 6.2). This involves passing the cryptographic key in the clear over the Ethernet or parallel cable using the `NET_CMD_SET_SIGNING_KEY` command, which stores it unprotected the flash on the device. It can then be extracted in a number of ways, including:

- Reading the key directly out of memory with `MEM_READ`.
- Loading new firmware onto the device which allows key export.
- Opening the case and reading the flash directly.

We have tested the first of these approaches.

Prerequisites Reading the key directly out of memory would require only direct physical access to the device. Loading new firmware would require similar access, though if firmware were made more difficult to load (see Section 6.1), then this attack would be much harder. Opening the case requires extensive physical access to the machine.

Impact All of these methods would allow raw access to the MAC key and enable the attacks described in Issue 22. Note that even if asymmetric keys were used, it would still be possible to forge MBBs from the compromised device.

Mitigations The first avenue of attack (reading the key directly out of memory) could be mitigated by removing `MEM_READ` access to that portion of memory. Loading new firmware could be made more difficult, as described in Section 6.1. The case opening attack is substantially harder to thwart, but the use of a tamper-resistant cryptographic module such as the eCM on the eScans and JBCs would make this attack significantly more difficult.

Status This issue was discovered by examination of the source code. We have successfully used a computer under our control to communicate with the eScan and extract the cryptographic keys. We have not attempted the other two avenues of attack.

6.8 MBB Vote Storage

Both the eSlate and the eScan store their vote records in *Mobile Ballot Boxes* (MBBs), which are generic PCMCIA memory cards. Each eScan has an MBB and each JBC has a single MBB which serves all the eSlates under its control. Integrity for MBB data is intended to be provided by an HMAC-based [5] message authentication code over some of the contents. As discussed in Section 6.7, the MAC is keyed with a key common to all the precincts in a county.

Issue 25: Vote order can be determined

In order to preserve voter privacy, it is necessary that an attacker who has access to the MBB not be able to determine how individual voters cast their votes. Although the votes recorded on the MBB do not contain the identity of the voter, it is possible to determine both the order in which votes were cast, and in the case of the eSlate, the order in which voters were issued access codes. Because the order in which voters vote is fairly easy to determine, being able to determine the order in which votes are cast provides a large amount of information about how any individual voter votes, even without the MBB explicitly containing voter identities.

Detailed Description Each vote is stored on an MBB in two ways:

- The vote itself is stored in a *cast vote record* (CVR) which is stored in a randomly selected location in a large memory block.
- An entry representing the vote is appended to the audit log.

This method has two problems. The first is that the randomization of the CVR location is poor and leaks a significant amount of sequence information. The second is that the in-sequence audit log leaks the order of the CVRs.

CVR Randomization Before any votes are cast, the eScan or JBC allocates a large random block on the MBB. It then selects a random point approximately in the middle of that block and initialized two pointers, $start$ and end with the $start$ pointing at the selected point and $end = start + 1$. Whenever a CVR needs to be stored, the device selects a random 1-bit value and uses that to select whether to store at the beginning or end of the log. If the beginning, the CVR is stored just before $start$. If the end, it is stored at end. The appropriate pointer is then moved away from the center of the buffer and the process is repeated for any new vote.

Given this algorithm, if the initial value of $start$ (referred to here as $start_0$) is known, it is easy to derive a partial vote ordering for two CVRs at positions p_1 and p_2 provided that either $p_1 < start_0$ and $p_2 < start_0$ or $p_1 > start_0$ and $p_2 > start_0$. In either case, the CVR closer to $start_0$ must have been cast first. Even for votes on different sides of $start_0$ it is possible to determine a rough ordering if their distances from $start_0$ are sufficiently large.

The above algorithm requires determining $start_0$. This can be approximately determined by taking the center of the used portion of the buffer. However, a more precise estimate may be possible by observing that that algorithm for selecting $start_0$ selects a random position from a 100 byte region centered on the middle of the buffer. The end of the first prepended vote must fall within this region, as most the beginning of the first appended vote. Because the CVR header alone is 17 octets long, this narrows $start_0$ down to within at most 6 values, and in fact less because the CVR data must also be stored. In one sample election run by the Red Team, the smallest CVR (with no write-ins) was 32 bytes.

Derandomization via Audit Log Although the audit log does not contain CVRs, each audit entry contains a CRC-16 of the CVR data. This allows the attacker to determine a small set of CVRs (most likely all containing the same votes) which correspond to a given audit log entry. Because the audit log entries are in sequence, this allows an attacker to completely recover the sequence of cast votes. In addition, when voters vote by eSlate, the voter access code appears in the audit log and it is therefore possible to determine how voters voted by the order in which they were authorized, particularly if voters keep their access code printouts, which are produced by the JBC.

The SERVO Device Audit Log report also appears to display the audit log data, containing the CRC and the voter code, which may allow mapping of voters to vote data. Whether this is possible depends on whether the CRCs can be completely predicted from the voter's choices, without access to the CVR log data. We have not determined whether that is the case.

Prerequisites In order to exploit this issue, an attacker would need read access to an MBB or to the contents of the MBB. Because MBBs are not encrypted, no access to keying material is required. Election officials are likely to have access to MBBs or MBB contents. Whether poll workers have access to MBB contents depends on whether the MBBs are sealed into the eScan/JBC at the central warehouse and then shipped back to the warehouse inside the JBC or whether they are removed prior to shipping. If the latter, then poll workers would also be able to mount this attack.

Poll workers could also access the CVR and audit logs internal to the eSlate, JBC, and eScan using the management interfaces; see Section 6.1.

If the SERVO audit logs can be mapped back to vote results, then this attack may be practical with only public information.

Impact The result of this attack is that anyone who has access to the contents of an MBB can determine which votes were cast in which order. If the attacker also has access to the order in which voters entered the polling place, the order in which they cast their votes, or access to voters' access code printouts, this leaks a significant amount of information about how individual voters voted.

Mitigations Poll workers can be blocked from exploiting this issue by restricting their access to MBBs and to the network/management interfaces on polling place devices. In particular, if the MBB is sealed in the JBC/eSlate whenever the unit is in the poll worker's possession, then this attack becomes more difficult to mount without being detected. Future versions of Hart software could routinely encrypting the MBB contents, which would also block poll worker attacks. The contents could be encrypted under a public key owned by Election Central, thus preventing leakage of the key by device compromise.

Because MBB CVR information is routinely stored by the back-end election systems, it is more difficult to restrict access by election officials.

Merely improving the CVR randomization algorithm does not significantly mitigate this issue because the CVRs can still be derandomized by examining the audit log. Future versions of the Hart software could remove the audit log vector by either further restricting access to the audit log or making it significantly more difficult to tie audit log entries to CVRs, most likely by removing the CRC-16 hash of the CVR from the audit log. We have not determined whether the CRC-16 is used by any of Hart's auditing mechanisms, but as it's not cryptographically strong, it is only useful as a check for data corruption, not malicious changes.

Status This issue was discovered by examination of the source code. We have developed tools to reconstruct the vote order and used them with an MBB image provided by the red team to extract the vote order in a given election. We also verified that the first vote (representing location of $start_0$) was in the expected location.

Issue 26: MBB is not protected during voting

In order to protect against tampering, Hart's MBBs are cryptographically protected using HMAC [5].[18] In theory, this protects the relevant structures on the MBB from tampering by anyone who does not

[18]The material provided to us refers to this as a "signature" but it's actually a symmetric message authentication code (MAC).

have the relevant MAC key. However, the HMAC is not present on the vote data on MBB during the election, but is computed and recorded either when the election is locked to accommodate a pause in early voting or when the election is closed. Therefore, it may be possible to remove, modify, and reinsert the MBB without being detected.

Detailed Description The eScan and JBC both maintain vote data in two locations:

- In internal static storage.
- On the MBB.

As indicated above, each location has two data structures, the CVR data and the audit log.

This data is maintained unprotected up to the point where MBB is closed or locked. The internal memory does not appear to be accessible without opening the case of the JBC or eScan. Assessing the difficulty of this is outside the scope of this work. The MBB, however, is readily accessible, but tamper seals are intended to detect removal.

If an MBB is removed, modified, and reinserted, the result is that the internal and external data structures will not match. However, the JBC and eScan do not appear to verify that the internal and external audit and CVR logs match and therefore voting proceeds as normal.

JBCs and eScans both run a watchdog task that triggers an audit log message and halts the device if the MBB is removed. The attacker can power down the device, remove, modify, and reinsert the MBB, then power the device back up. The reboot will cause the JBC, at least, to check that the inserted MBB is the same as the one it originally read the election information from, but this check does not appear to extend to comparing the internal CVR and audit logs to those stored on the MBB.

When it is time to “sign” the MBB, the following steps are performed:

1. Generate an unkeyed hash for the internal CVR data and log it to both the internal and external audit logs.
2. Generated a keyed MAC (HMAC-SHA1) of the external CVR data and log it to both audit logs.
3. Generate an unkeyed hash of the internal audit log and log it to both audit logs.
4. Generate a keyed MAC (HMAC-SHA1) of the external audit log and log it to both audit logs.

When the MBB is verified by Tally, it verifies the MACs but does not have access to the internal audit or CVR data and does not attempt to compare them to the data on the MBB.

Prerequisites The major obstacles to this attack are obtaining access to the device long enough to remove and reinsert the MBB and the tamper sealing on the MBB insertion point. A poll worker would have appropriate access to the devices though a voter would not. See Section 3.5 for a discussion on the limitations of tamper sealing.

Impact An attacker who was able to obtain access to the MBB would be able to change existing votes on it and then reinsert it into the JBC or eScan and have that unit proceed as normal, including inserting the MAC. This MBB would have votes partly of the attacker’s choice and would be accepted by the central tabulation system.

Mitigations Several possibilities for mitigating this attack exist.

Election administrators could verify the internal audit logs (collected via SERVO) against each MBB to verify that they match. We have no information as to whether this is currently routinely done, but it could be done with the current Hart software.

In future versions of Hart software, the JBC/eScan could check on each MBB insertion that the contents of the MBB match the internal memory and fail if they did not. One minor difficulty here is that the randomization of the CVRs on the MBB and the internal memory is not the same;[19] however, this could be fixed by lexically sorting the CVRs prior to verification.

Status This issue was discovered by the Red Team against a live system and verified by reference to the source code.

Issue 27: Handling of precinct IDs in CVRs

Each CVR contains a precinct ID field indicating the precinct for which the CVR was voted. This allows a single MBB to contain consolidated records for a number of different precincts. It may be possible to generate an MBB which contains votes for unauthorized precincts. In addition, it may be possible for an attacker who has access to the election database on Tally to silently suppress votes.

Detailed Description Each CVR in an MBB is an independent record, with a descriptive header containing meta-information about the votes in the CVR. This meta-information includes:

- A 16-bit ballot type identifier
- A 16-bit precinct type identifier
- A 16-bit party identifier

In theory, any given piece of equipment is only relevant to a small number (most commonly one) of precincts, and under ordinary circumstances when Tally converts votes into the internal storage it refers to the election database to determine what precincts and parties are authorized for a given MBB. It then cycles through each valid precinct/party combination in order to count votes. This prevents an MBB from precinct X from injecting votes in precinct Y. However, it has the side effect that it is possible for votes to get ignored if the database on Tally is inconsistent with the MBB.

The one exception to this rule appears to be that SERVO is allowed to create recount MBBs, which contain records from multiple polling places. If Tally thinks it is processing such an MBB, it executes `tallyCvrLogForServo()` which appears to iterate through all possible precincts and parties. Tally makes this decision based on the MBB header, which is under control of the precinct polling devices. Thus, it may be possible for an attacker to forge a SERVO MBB which contains arbitrary precinct IDs.

Prerequisites Mounting a suppression attack would require access to the Tally database in order to modify the precincts assigned to a given polling place. This is election worker style access.

Mounting this attack requires the ability to forge MBBs. This could either be done by extracting the master MAC key (see Section 6.7) or by using the attack described in Issue 26. An attacker who had arranged to replace the firmware on an eScan or JBC could also mount this attack.

In any case, the attacker would need access to the MBB, such as may be available to a poll worker.

Impact The impact of a suppression attack would be to allow someone with temporary access to the election database to suppress votes from a given precinct/polling place pair.

The impact of an injection attack would be to allow an attacker who had partial control of equipment in one precinct to affect the outcome of voting in another precinct by injecting his votes into that precinct's data.

[19] This is also suboptimal because it leaks additional ordering information.

Mitigations One might attempt to mitigate this issue in a number of ways. First, one might attempt to restrict physical access to the MBB, perhaps with extensive tamper sealing, thus preventing it from being rewritten. The security level of this defense depends on the attack vector. If the attacker wants to write the MBB with his own computer, this provides some level of security. If the attacker has compromised the JBC/eScan, then he can arbitrarily rewrite the MBB without removing it from the JBC/eScan.

Auditing the MBB contents against JBC/eScan internal memory might also detect this attack if the attacker has only altered the MBB. However, it will not detect the attack if the attacker has compromised the JBC or eScan and therefore can modify the internal memory.

Finally, one might attempt to enforce this on the Tally side. This could be done in two ways. First, one could compare the number of recorded votes in each precinct against the number of voters signed into the precinct and flag discrepancies. Second, one could keep records of which MBBs were assigned to each precinct and flag events where precincts which should not appear on a given MBB in fact do. However, this might interfere with consolidation features and the creation of recount MBBs by SERVO.

In future versions of Hart software, these records could naturally be stored in the MBB header. However, because the header is integrity protected with a key available to the JBC/eScan, this will not defend against an attacker who has compromised them without a stronger cryptographic key infrastructure (see Section 6.7).

Status This issue was discovered by examination of the source code. The MBB-reading code is extremely complicated and may include sanity checks that prevent either of these attacks, although we did not see any such checks. Testing on a real system would be required to confirm or reject this issue.

Issue 28: Users can read unclosed MBBs or MBBs with invalid MACs

Although the data on the MBB is integrity protected with HMAC, this HMAC is only applied when the election is closed. When Tally is asked to read an MBB which isn't closed or when the cryptographic check fails it shows a dialog allowing the user to accept the MBB anyway. This provides a potential way to bypass the MAC check.

Detailed Description When Tally reads in an MBB it checks the MAC value. Based on the documentation, if the MAC check fails, it complains that the MBB is corrupt and allows the user to accept it anyway. Similarly, if the election is not closed (and therefore there is no MAC), Tally pops up a dialog giving the user the option to accept the MBB anyway. This provides an easier attack target for an attacker who has possession of an MBB but cannot forge the MAC: they send in the MBB with a bogus MAC or simply don't close the election or strip off the closure information in the audit log. There is no guarantee that the MBB will be accepted, but experience in other environments, where users are asked to enforce security rules [29, 34], suggests that there will be a high rate of users accepting invalid MBBs.

Note that the concern here with respect to Tally users is inattention, not malice, since a sophisticated Tally user can forge MBBs.

Prerequisites To exploit this an attacker would need to be able to either modify an existing MBB or substitute/add his own. This probably means a poll worker or election official.

Impact This attack would potentially allow an attacker to inject false votes.

Mitigations One potential mitigation for future Hart software versions would be to simply refuse to ever accept un-closed MBBs. However, this would have the effect that if poll workers make any mistakes, perhaps forgetting to close a precinct before shipping off the MBB, recovery would be difficult.

This could perhaps be addressed with a separate tool that Hart would supply to election administrators which could be used to sign unclosed MBBs. This tool would require supervisor access and produce extensive paper logging. Such MBBs should perhaps be flagged for a higher probability of post-election investigation since they are more likely to be corrupt.

Status This issue was discovered in the source code and verified in the documentation but we have not tested it ourselves.

Issue 29: The protective counter is subject to tampering

A voting machine are required to keep a private protective counter that is incremented with each vote cast and that reflects the total number of votes ever cast on that machine. However, on the JBC, eSlate, and eScan, the private counter can be tampered with by software running on the machine and through the management interface.

Detailed Description Each of the JBC, eSlate, and eScan keeps the private counter in memory as a 32-bit integer (along with a 16-bit CRC, which is ineffective at detecting tampering). On each of these devices, the counter is kept stored in flash memory. The eScan uses its "estore manager," which keeps two copies at location `STORE_VOTE_COUNTERS`. This flash copy appears to be updated only when the logs are cleared; the current private counter is the sum of the counter in flash and the votes recorded in the audit log. The JBC and eSlate store two copies interleaved directly in flash, at `ADDR_PRIVATE_COUNTER`, i.e., `0x0000c000`.

Since the software on the JBC, eSlate, and eScan can modify the stored private counters directly, any malicious software loaded onto these devices can modify the protective counter. What's more, it is possible to use the management interface to modify the protective counter without replacing the software. For example, the JBC processes a `NET_CMD_CLR_PRIVATE_CNT` message that resets the private counter. In addition, an attacker can rewrite the private counter directly using the `MEM_WRITE` option to the `NET_CMD_MEMORY` message.

Prerequisites To exploit this an attacker would need either to replace the software running on a JBC, eSlate, or eScan, or to be able to connect to the device's management interface.

Impact This attack would allow the attacker to modify the protective counter.

Mitigations Implementing a counter that can be incremented but not reset or otherwise modified requires specialized hardware support, which Hart would have to add to the JBC, eSlate, and eScan.

Status This issue was discovered in the source code, but we have not attempted to verify it.

Issue 30: If two MBBs have the same ID, Tally only reads the first one

Tally maintains a database which stores each MBB. If an attacker can load a bogus MBB that has the same ID as a real one, before that real MBB was loaded, the real MBB will be rejected by Tally.

Detailed Description Every MBB has an identifier that should be unique within the county. Before a new MBB is inserted into the database, the database is first queried to see if the MBB ID is already present. `MBBValidateCls::InsertSubLog` is responsible for this process, ultimately calling into `MbbDBInfoCls::AlreadyProcessed` which does the SQL query.

Nowhere in the process is there any check to make sure that the given MBB ID is the proper ID for a given precinct. As a result, an attacker could produce a malicious MBB for a target precinct, perhaps as a result of compromising a Rally server or a JBC, and then Tally would load that MBB.

So long as the attacker's malicious MBB goes first, it will prevent the legitimate MBB from being loaded.

A suitable error message will be logged, but election officials would have no easy way to resolve the error condition. Even though the log indicates the identifier of the conflicting MBB, Tally has no MBB removal mechanism available from its user interface.

Impact This is an example of a "denial of service" attack, which will slow down the tallying process, possibly requiring assistance from the vendor in order to resolve the issue. This could delay election reporting for several days.

Prerequisites An attacker needs the ability to introduce bogus MBBs into the Rally/Tally system. This could be accomplished through attacks against eScan or JBC systems, in the polling place, through attacks against Rally, perhaps via modem, or through attacks against SERVO, if it were used to extract MBBs directly from the voting machines.

An attacker also needs to know the MBB identifiers for any particular precincts that he wishes to suppress. BOSS, which writes out MBBs, generates these identifiers sequentially as it writes the cards, starting out from one and working its way up. This means that an attacker can simply use small numbers for MBBs to perform the attack, if he has no particular preference for which precincts he wishes to suppress. If he wants to suppress a particular precinct, he would need to know the exact order in which the MBBs are written by BOSS.

Mitigations This attack must be performed after the ability to inject a bogus MBB has been accomplished. As such, the mitigations which might protect against those attacks would also mitigate against this attack.

Should an attack of this form succeed, no actual vote data would be lost. A vendor technician could potentially issue raw SQL commands to clean up the database. Likewise, VVPAT printouts and paper ballots could be tabulated by hand for the affected precincts.

Status This issue was discovered by examination of the source code. We have not attempted to verify it with a real Tally installation.

6.9 Rally/Tally's Use of TLS/SSL

When communicating over a modem (see Section 6.6), Rally and Tally encrypt their conversation using OpenSSL (the Secure Sockets Library), which, when used properly, could both prevent tampering and eavesdropping on the communication.

Issue 31: Rally and Tally use an old version of OpenSSL with known bugs.

Rally/Tally use OpenSSL version 0.9.7d, circa March 2004. This opens Rally and Tally servers to attacks against documented OpenSSL bugs.

Background When you visit a secure web site (i.e., any web site whose URL begins with "https") the web site and your browser communicate with the Transport Layer Security (TLS) protocol [8], often referred to by its earlier name, the Secure Sockets Layer (SSL, which we will use here). SSL is most commonly used for secure web pages, but this general-purpose technology and can be used for a variety of other tasks, as it is used, for example, by Rally and Tally.

SSL use encryption technologies to provide a variety of useful properties:

Integrity SSL uses message authentication codes (MACs) to ensure that the data received by one side of the conversation is precisely the same data that was sent by the other side of the conversation.

Privacy SSL provides optional encryption, using a variety of different algorithms, to ensure that eavesdroppers cannot learn the plaintext of the conversation.

Authentication SSL can validate that the party on the other side has a specific identity. This is done using *certificates*, which are digitally signed statements made by a *certificate authority* (CA) who is trusted by both parties. When the two parties start their SSL conversation, they will use cryptographic techniques to prove that they are the holder of a particular secret key which corresponds to a public key. Likewise, they will prove that the CA agrees that the public key corresponds to their name, organization, email address, or other identifying information specified in the certificate.

OpenSSL is a free, open-source implementation of the SSL protocol that is widely used, particularly by the Apache web server. Hart InterCivic has adopted OpenSSL to use for protecting the conversations between Rally and Tally when they are performed over a modem.

Detailed Description One of the supported configurations for Rally and Tally is to create "regional" centers, where poll workers would bring their MBBs at the end of the day. The regional centers would have PCs running Rally, which would be used to collect the MBBs together. Tally, running at Election Central, would make a phone call to each regional center, connecting to the computer running Rally, and would then download the MBBs. This connection is protected by the use of SSL and by the use of a username and password.

Alternately, Rally machines may be installed alongside Tally machines at Election Central. This would allow there to be multiple "processing desks" to handle incoming MBBs, speeding the process. In this configuration, the Rally and Tally machines are connected by a local area network (i.e., standard Ethernet), which Tally uses to copy the MBBs from each Rally machine. SSL and username/passwords are also used in this configuration.

While we were not given the source code to the version of OpenSSL used by Hart InterCivic, we were given the DLLs (compiled binary libraries) that contain OpenSSL. By examining the DLLs, we determined that Hart is using OpenSSL 0.9.7d, which was released in March 2004. The OpenSSL maintainers' vulnerabilities web page[20] documents seven known issues that apply to OpenSSL 0.9.7d. These issues include one attack which can be used to crash the remote system and another which allows a buffer overflow attack. The buffer overflow attack relies on a flaw within `SSL_get_shared_ciphers()`, which is presently unused by Rally and Tally. The other relevant OpenSSL flaws allow an attacker to crash the program or to induce an infinite loop (rendering the program unresponsive).

When Rally and Tally are used with modems to connect them (see Section 6.6), an attacker may also connect to the Rally machine via modem and can then use these known OpenSSL vulnerability to attack Rally.

Prerequisites Rally and Tally have two use modes: communicating over modems or communicating over a network[21]. This attack mode functions in any context where an attacker can access the network to which Tally or Rally is connected—in particular, if Rally is connected to a modem.

If, for whatever reason, a Rally system was connected via its network port to the Internet this vulnerability would also be exploitable by a remote attacker over the Internet.

Impact When Rally and Tally are used with modems, an attacker can cause an arbitrary Rally system to crash or become unresponsive, simply by dialing its phone number. Such an attack could be used to slow down or inconvenience election officials.

Mitigations One obvious mitigation to the risks posed by outsiders connecting to Rally/Tally modems is to avoid any use of modems. Rally servers can operate offline, collecting votes without

[20] `http://www.openssl.org/news/vulnerabilities.html`

[21] The use procedures [21] specify an intranet, but there is no technical reason it could not be the Internet

transmitting them to Tally. Once all the votes have been collected, the Rally server can be physically transported from the regional vote collection facility to Election Central.

Another possible mitigation is to train election officials how to respond to these attacks (e.g., how to kill and restart the Rally application) or to switch to physical transportation of the Rally machines if the modems are not working properly or are under attack.

Of course, future software releases from Hart could be engineered to address these concerns, most likely by upgrading to the current version of OpenSSL (and following subsequent OpenSSL upgrades as they become available).

Status This issue was discovered by examination of the source code and other files provided to the source code analysis team. It has not been directly verified against Rally and Tally servers.

Issue 32: Rally and Tally, when presented with an unknown certificate, will present the unauthenticated name and organization to the user for verification.

Detailed Description With Rally and Tally, there is no certificate authority (CA) in use. Instead, Rally and Tally have a memory of the certificate used "last time" and will present a dialog box to the user if the certificate changes. This dialog box presents the name, organization, city, state, and country from the remote machine's certificate. However, because no CA was used to sign the certificate, its contents may be set arbitrarily by an attacker. If a user is convinced to press the "okay" button to this dialog (on either side: Rally or Tally), then an attacker can potentially impersonate one side to the other.

Once the SSL connection is initialized, Tally authenticates itself to Rally by presenting a username and password inside the SSL connection. This username and password are configured in advance, before the Rally machines are deployed. This process is analogous, in some fashion, to how users authenticate themselves to secure commerce web sites, but it's not very good for use when two computers are connecting to one another on a regular basis. The use of human-chosen passwords is an unnecessary opportunity for poor passwords to be chosen.

Aside from the SSL certificate authentication, there is no other mechanism for Rally to authenticate itself to Tally. This means that an attacker who can intercept the connection between Rally and Tally, perhaps by climbing a telephone pole and clipping into the appropriate wires, may be able to impersonate a Rally server to Tally. The only mechanism preventing such an attack is the dialog box that will appear on the Tally machine asking the user to approve of the new certificate (which, as described above, can present seemingly correct information).

Prerequisites Rally and Tally must be configured to communicate over a modem or other network connection. The user of Rally or Tally (whichever machine is being attacked) must press okay to a dialog box that, by all appearances, is presenting good information. In the case of attacks against Rally, the attacker must know the phone number of Rally and must know the username and password. In the case of attacks against Tally, the attacker must be able to intercept the phone call from Tally to Rally.

If, for whatever reason, a Rally or Tally system was connected via its network port to the Internet, then this vulnerability would also be exploitable by a remote attacker over the Internet.

Impact If modems are used to connect Rally and Tally and if the usernames and passwords are chosen poorly, an attacker will be able to connect to Rally and download all of the votes stored in it. Likewise, if an attacker is able to intercept the outgoing phone calls from Election Central (e.g., by climbing a nearby telephone pole and attaching alligator clips to the appropriate wires), then the attacker would be able to insert his own computer in place of the true Rally machine, feeding back arbitrary vote information to the Tally server.

Mitigations One possible mitigation is to ensure that all Rally and Tally systems are "preloaded" with the necessary cryptographic certificates. This can be performed in the Election Central warehouse, prior to deploying the Rally systems to the field. This would allow for poll workers and election officials to be instructed, under all circumstance, to reject requests that Rally or Tally might pop-up on the screen to approve any new or previously unseen certificate, regardless of the certificate contents presented on-screen.

The use of strong usernames and passwords also mitigates against this threat, as an attacker who does not and cannot guess the username and password will be unable to make any further progress attacking Rally, but would remain unhindered in presenting a false Rally to spoof Tally. He might be able to leverage that attack to capture the Tally password and then impersonate Tally to Rally.

The username and password chosen within Rally and Tally to authenticate the systems to one another must be carefully chosen to have a high degree of randomness. A number of commercial and free software tools are available that can generate suitable passwords. Likewise, passwords could be selected by rolling dice. Every Rally machine can and should have a different username and password.

This issue could also be mitigated as part of a broader redesign of Hart's key management architecture (see Section 6.7), in which keys are managed centrally and less subject to user discretion.

Status This issue was discovered by examination of the source code provided to the code analysis team as well as reading the Rally and Tally user manuals [18, 20]. It has not been directly verified against Rally and Tally servers.

6.10 Verified Ballot Option Issues

As required by California law, Hart's DRE system has a Voter Verified Paper Audit Trail (VVPAT), which Hart calls the *Verified Ballot Option* (VBO), which uses the VBOx, a reel-to-reel printer which attaches to the eSlate. Once the voter has indicated that he wishes to cast his ballot, the ballot information is printed by the VBOx and presented to the user. The voter can then indicate either acceptance or rejection of the ballot.

If the voter accepts the ballot, the VBOx prints an "BALLOT ACCEPTED" message and a bar code and then scrolls to a blank page. If the voter rejects the ballot, the VBOx prints a "BALLOT REJECTED" message and the eSlate allows the user to adjust his vote.

The VBOx is intended to provide evidence of the voter's intent in a way that is not amenable to tampering by compromised software on the DRE. However, we found a number of implementation decisions that might allow an attacker who controlled the eSlate to construct bogus paper audit trails.

Issue 33: VBOx printing and scrolling is controlled by the eSlate

The VBOx itself is an extremely simple printer. The commands it receives from the eSlate just tell it to print arbitrary data to the paper and/or scroll the paper forward. This enables a compromised eSlate to exert a large amount of control over the VBOx.

Detailed Description The eSlate communicates with the VBOx by sending it a variety of commands. The two most relevant here are `Print` and `FormFeed`. These allow the eSlate to print arbitrary bytes and to scroll the paper forward. However, the general flexibility offered here plus the fact that there is no external user control on the VBOx, allow a compromised eSlate flexibility to mount attacks.

The simplest such attack would be forging votes. As described in Issue 8, a compromised eSlate can send votes to the JBC even if no acceptable voter code has been entered into the eSlate. However, this potentially produces an inconsistency between the recorded votes and the VVPAT. Because the eSlate controls the VVPAT, it can simply issue its own instructions to the VVPAT to print the vote along with the accepted banner.

Other attacks are also possible, such as waiting for a voter to confirm his vote, then printing "BALLOT REJECTED", printing a new, corrupt vote with the "BALLOT ACCEPTED" banner and then scrolling to a blank page.

Yet another possibility would be to simulate VBOx failures (e.g., formfeed through all the paper) to force the polling place to fall back to electronic only voting. We do not know how polling places deal with failures of the VVPAT.

Prerequisites To exploit this issue, an attacker would need to control the eSlate or somehow gain control of the serial connection between the eSlate and the VBOx. There are two points of attack here: the connection between the eSlate stand and the VBOx, which appears to be a simple serial cable, and the contacts in the eSlate stand which mate to contacts on the bottom of the eSlate. It is not clear that either are well sealed and the eSlate stand contacts are voter accessible, though clumsily nonstandard to access.

Impact An attacker can maintain consistency between tampered electronic vote records and a tampered VVPAT, thus decreasing the risk of detection.

Mitigations A partial mitigation would be to provide the VBOx with separate hardware controls so that the voter had to operate the VBOx in order to accept their ballot. This would prevent attacks where the eSlate ran off incorrect ballots. One approach that has been suggested in the past but would require somewhat extensive re-engineering would be to have the VBOx have a mechanical switch that when depressed cut off the ballot and dropped it into a hopper, thus both requiring manual engagement in the voting process and providing some measure of privacy for the sequence of votes (see Issue 35.)

Status This issue was discovered through examination of the source code. It has not been tested on the VBOx Printer hardware, which may not in fact support a rewind feature.

Issue 34: The VBOx code indicates a reverse function

A VBOx printer is connected to an eSlate device, which issues it commands to print data. It appears to be possible for the VBOx printer to be run backwards, thereby overwriting previous ballot records. While there are not commands implemented in the source code to perform this action, if an attacker were to gain control of a VBOx Printer (for example via a compromised eSlate) they might be able to cause the VBOx Printer to run backwards.

Detailed Description When a VBOx Printer receives a command from an eSlate, code on the printer is executed that instructs the hardware to perform the printing. Within this VBOx Printer source code, we found several commented out lines of code that seemed to indicate it was possible to instruct the printer hardware to run in reverse. So under normal operating conditions, running in reverse would not occur. However, if an attacker were able to take control of a VBOx Printer, e.g. by exploiting a buffer overflow, it seems he could instruct the printer hardware to run backwards.

Prerequisites An attacker who is able to execute code on the VBOx printer could be able cause the VBOx Printer to run backwards.

Impact Running the VBOx Printer backwards would overwrite previously written ballot records, thereby corrupting a source of data for performing manual recounts of ballots. It would also enable an observer to see previously-cast votes, violating the privacy of those voters.

Mitigations The natural mitigation for this issue is to disable backwards feeding in the VBOx Printer hardware. If this change were made, even a complete compromise of the VBOx Printer software would not result in the overwriting of ballot records.

Status This issue was discovered through examination of the source code. It has not been tested on the VBOx Printer hardware.

Issue 35: The VBOx votes are sequential and so compromise vote privacy

The VBOx is a reel-to-reel printer with votes printed sequentially. Therefore, any attacker who had access to the paper could determine the sequence of votes and how any individual voter voted.

Detailed Description Cast votes are printed out on the VBOx in sequence both in human-readable mode and as machine-readable bar codes. This allows an attacker who has access to the tape to determine the order of votes. With access to the order in which voters voted, this allows reconstruction of individual votes. Even if the paper tape has been cut, it may be possible with enough work to line up cut edges.

This attack is made more useful by the predictability of voter codes (Issue 7), because an attacker can identify where a particular voter code occurred to the sequence of voters.

Prerequisites The attacker would need access to the VBOx paper tape. This could potentially be available to a poll worker who violated the VBOx tamper seals, if any. It would be available to a number of election officials.

Impact This issue allows violation of vote secrecy.

Mitigation The primary mitigation here is to ensure that nobody ever has access to the paper tape in complete form. If a VBOx audit is performed, some mechanism should be used to blindly cut the paper apart, perhaps mechanically. This appears to be an issue for all current VVPAT-based e-voting systems.

Status This issue was discovered by inspection of the system and is generic to reel-to-reel VVPATs of this type.

6.11 Code Quality and Miscellaneous

In addition to the specific issues discussed above, we identified a number of problems related to the implementation of the Hart InterCivic voting suite that bear discussion.

Issue 36: Pervasive failure to follow commonly agreed safe coding practices

Writing secure code is an extremely difficult problem, especially in C and C++, which give the programmer an extraordinary amount of control over memory, and thus a large number of ways to make serious mistakes.

Detailed Description Specific classes of problems include:

Failure to check array bounds. In addition to the exploitable buffer overflows listed previously, we discovered numerous places in the source code where memory buffers were used without appropriate bounds checking. In some cases system crashes could potentially be triggered by an attacker taking advantage of these overflows.

Through a combination of manual and automated source code examination —using Fortify— we found a number of buffer overflows in the Hart source code. While most of them did not open the systems to control by an attacker, these overflows were suggestive of a fragile codebase and of a software development process that has room for improvement.

Format string vulnerabilities. The correct way to print a string using the `printf` family of functions is `printf("%s", str)`, not `printf(str)`. This because the latter form can lead to unexpected results if `str` contains format specifiers such as `%d`. In fact, a carefully crafted malicious format string, when interpreted by `printf`, can allow the attacker to execute arbitrary code.

The Hart source contains multiple instances where `printf` is used with a non-constant format specifier. One example is the JBC's report mode, which uses `printf` to print write-in candidate names; see Issue 6.

Signedness and integer overflow errors. It is important to watch out for arithmetic operations whose result is outside the range of values that can be represented by the result type. For example, $30,000 + 30,000$ as a signed 16-bit integer is -5536, not $60,000$. Wraparound of this sort can lead to incorrect results. It can also lead to security vulnerabilities of a class known as integer overflows [6]. A common variety of integer overflow arises from carelessness about the signedness of values, leading to (implicit) integer overflow when a signed quantity is type-cast as unsigned or vice versa.

The Hart source code we examined contains multiple integer overflows. It makes extensive use of 16- and 32-bit signed and unsigned integers, and frequently converts between these. This sometimes leads to exploitable vulnerabilities.

For example, consider Tally's MBB processing. For each MBB, Tally reads the extents of the CVR log from the MBB header, then computes the size of this log by subtracting the end offset from the start offset—as a 32-bit *signed* integer. If the difference between these offsets exceeds 2,147,483,647, it is represented as a negative number, which means that Tally's check that the size is less than the maximum allowed size is satisfied. The log is then read into a buffer, which causes `memcpy` to be invoked with the computed size as its size argument. But `memcpy` treats its size argument as an *unsigned* integer, which means it copies more bytes than are allocated for the input and output buffers, leading to memory corruption.

Failure to check and propagate error conditions. Detecting and appropriately acting on unexpected error conditions is important for secure programming. Attackers compromise systems by causing them to deviate from their intended behavior, and this deviation will be characterized by an error condition. Software modules should check that each of their assumptions about their environment is satisfied before acting. Having discovered an anomalous condition, modules should propagate an alert to higher-level code that can take appropriate action in response.

The Hart source code we examined follows neither of these recommendations. This sometimes leads to exploitable vulnerabilities.

For example, when a JBC records a vote received from an eSlate, the JBC calls a function to remove the eSlate's claimed voter code from the list of active codes. If this function determines that supplied the voter code is not a valid voter code, it takes no corrective action. It does not log an alert, it does not alert the operator, and it does not propagate the error condition up—in fact, the function is of type `void`. (Even if the function did propagate the error up, it is already too late to remove the vote from the CVR log.) The lack of error checking is crucial in enabling the attack described in Issue 8.

Failure to minimize trust between components. Software modules should be written to minimize reliance on the correctness of other modules. This is not only useful for robustness but crucial for security. If this principle is not followed, a compromise of one module can lead to a compromise of another that trusts it to behave reasonably.

The Hart source code we examined does not follow this principle. This sometimes leads to exploitable vulnerabilities.

For example, when SERVO is backing up the audit log of an eScan, JBC, or eSlate, it asks the device what the extents of the device's audit log are and uses the answer to size a buffer for its copy of the log. Then, as part of the actual log dumping, SERVO again asks the device for the audit log extents to decide how many bytes to transfer. If the the two answers are the same, the copy proceeds as expected. If, however, the connected device has been compromised and claims a different (larger) log size on the second query than on the first, SERVO's allocated buffer is overwritten, leading to a heap overflow, as described in Issue 13.

Poorly unified design. Any time a piece of software has multiple implementations of the same type of functionality, there is an opportunity for error. The Hart software includes a number of such cases, including:

Multiple copies of pieces of code in particular, between the HartLib library and the eScan. In many cases the files seem identical, in others they appear to be system-dependent, and in yet other cases they appear to be branched versions which have diverged. This programming practice substantially increases the risk that the files will diverge in important ways, for instance fixes get applied in inconsistent ways, or that programmers get confused about the behavior of the subroutine being invoked.

Multiple error return handling mechanisms between different sections of code. In some places, the Hart code uses exceptions, in others, they use explicit return values, and in others they simply ignore the return values. This makes it hard for one implementor to be sure of the error convention used by another section of code, which can lead to errors being inappropriately ignored. We have found cases where errors should not have been ignored but were, though we do not know the cause.

Inconsistent memory allocation. Hart uses both `malloc` and `new`. There is nothing inherently unsafe about this in C++ code, but it runs the risk of misusing data managed with one method with the other method, which can create serious problems.

These issues make the code hard to maintain as well as to review, since one must frequently ask "what section (or sections) of code is in play here?" A unified system would be easier to work on.

Prerequisites The prerequisites for an attack based on an instance of the issues described above depend on the specific setting in the system where that instance is found. The knowledge required to exploit standard C-language vulnerabilities like buffer overflows is widely disseminated.

Mitigations Hart could rework its codebase, using some combination of: following commonly-agreed safe coding practices; auditing the code base by hand or with automated static analysis tools; and rewriting in a safer language such as Java.

Status This issue was discovered through both manual and automated examination of the source code.

CHAPTER 7

Attacks on the Full System

The previous section described a number of attacks on individual components of the Hart InterCivic voting system. However, an attacker would not be limited to mounting individual attacks, but would most likely want to string them together into an attack plan which accomplishes a set of real-world objectives.

In the rest of this section, we consider attacks designed to accomplish a number of common objectives:

- Compromising voter privacy
- Altering the final vote count
- Subverting all DRE devices in a precinct
- Subverting all the devices in a county

It's important to note that in many cases we will only present one or a handful of avenues towards achieving the desired goal. This should not be taken to imply that there are no other ways to achieve these goals or even that we do not know of others. Rather, this section is intended as an illustration of how an attacker might choose to attack the system.

7.1 Compromising Voter Privacy

As discussed in Section 3.2.3, there are two major applications for attacks on voter privacy:

- Vote buying/voter coercion
- Information gathering

The attacks described in this section work for both applications.

7.1.1 Vote Buying on eSlate Systems

In a vote buying attack, the attacker needs to be able to verify that the voter voted the way that he was instructed. The attacker can readily verify that the voter entered the polling place and was authorized to vote because Hart issues a receipt containing the voter access code. While this does not demonstrate that the voter voted at all or that he voted the way he was instructed to, it is useful in combination with other issues to form a complete vote buying attack.

The way that this works is that the vote buyer instructs the voter to vote a certain way. When the voter exits the polling place he provides his access code printout receipt to the vote buyer. The vote buyer can then use this access code as a lookup key into vote records stored on the MBBs, eSlate, or JBC to verify the voter's actions (see Section 6.8 for details on vote storage). The problem then becomes obtaining a copy of that vote data.

This is easiest to do if the attacker is a malicious poll worker, using the following steps:

1. After the election is closed and the MBB has been unsealed he places the MBB into a PCMCIA card reader (e.g., a laptop) and obtains an image. This is the only part of the process that requires access to protected materials.
2. He searches through the audit log for an access code entry that matches the voter's access code.
3. He searches through the CVR data for an entry with a CRC that matches the entry in the audit log (see Issue 25).
4. He examines the entry to see if it matches the expected votes.

This allows the attacker with high probability to determine that the voter has voted as instructed. An election official could of course mount the same type of attack. Naturally, once the MBB data has been obtained and decoded, the attacker can verify how any number of voters voted.

Another approach would be to read the internal audit logs of the eSlate or JBC via the parallel management interface (Issue 1). The same derandomization procedure can be used.

A voter can also mount this attack. In order to do so, he places a monitoring device on the serial port of any of the eSlates of the system, either in between the eSlate and the stand or on the free connector of the final eSlate (see Section 6.2 for more details on what such a device might look like). This device would be able to observe every cast vote from every eSlate in order and would simply record them. Because the access codes are transmitted by the eSlate to the JBC prior to voting, it is very easy to tie the access codes to the CVR. Alternatively, a voter could potentially attach a device at the very end of the election, impersonate the JBC, and simply suck out the contents of the JBC's memory. Note that this does not require the attacker to be present at the precinct, he merely needs to obtain the receipt from the voter.

7.1.2 Vote Buying on eScan Systems

Vote buying attacks on eScan systems are more difficult because the voter is not issued with a linkable receipt. However, because the audit log is created sequentially and can be linked to the CVR log, a poll worker who has access to the MBB can still mount a vote buying or coercion attack. The attacker observes the order in which people feed their ballots into the eScan. He then looks up the entry for the appropriate voter in the audit log and uses the techniques described in Issue 25 to find the relevant CVR. This tells him how each voter voted, in the order their ballots were cast.

7.1.3 Information Gathering

Extending this attack beyond a single voter to a generic information gathering attack is relatively straightforward. Against the eScan, the attack looks exactly like that described in Section 7.1.2. This allows us to match votes to the order in which voters voted.

Against the eSlate, the attacker can do the same thing as with the eScan, but observing the order in which voters voted may not be simple because there are a large number of eSlates. However, because the eSlate audit logs contain the access codes he can do something more convenient: match votes to the order in which voters entered the precinct and received access codes.

In order to mount this attack, the observes the order in which voters approach the JBC. He then extracts the audit logs from the JBC, eSlate, or MBB, which contain each access code in order, and then uses Issue 25 to match the access codes to the cast votes. This allows him to reconstruct how each voter voted. Note also that even if the access code issuance were not in the logs, it would still be possible to reconstruct vote order by taking advantage of the predictability of access codes (Issue 7.)

This attack requires having a collaborator in every precinct to observe the order in which people voted—provided that polling places do not keep such records—which makes large scale information gathering difficult. However, it would be practical for small-scale attacks.

7.2 Altering the Final Vote Count

Another important attack is to modify the final vote count. In this section we consider attacks where an attacker who controls a single device can modify the vote count. These attacks are obviously more powerful if the attacker controls multiple devices. We consider those issues in subsequent sections.

7.2.1 eScan

The most obvious attack is to subvert an eScan, which can be done via the attack described in Issue 3. This would most likely require either poll worker or election official access, or that the attacker break into the location where the eScan is stored. An attacker who subverted the eScan could mount at least two attacks:

- Extract the shared MAC key and use it to write a fake MBB (see Section 6.7).
- Install his own firmware image which counts votes in an arbitrary fashion (see Issue 3).

The second attack is more powerful because it would result in the eScan internal vote data, the MBB, and the summary tape, all having identical (wrong) results. As noted in Issue 12, it is easy to subvert the external integrity checking and so electronic audit procedures would not detect this as long as the total number of votes remains the same. The one percent manual recount would detect this form of attack, provided that the victim eScan was randomly selected for recount.

7.2.2 eSlate

While attacking an eScan is most likely outside the reach of a voter, we believe that a voter could subvert an eSlate, as described in Issue 2. Such an attacker could install new firmware on the eSlate, which would act in an arbitrary fashion.

The difficulty with the eSlate is that there are four types of records which one wishes to have agree:

- CVRs/Audit logs on the MBB
- CVRs/Audit logs in the JBC's internal memory
- CVRs/Audit logs on the eSlate's internal memory
- The VVPAT records

The MBB data is what is ordinarily used for tallying, so modifying that is attractive, but our attacker may wish to survive an audit, in which case he must arrange to tamper with the other records as well.

If we assume that the JBC is uncompromised, then the first two types of records agree by definition. The difficulty is getting the first two types to agree with the third and fourth types.

We consider two example attacks. In the first, the attacker changes a voter's vote before it leaves the eSlate. In the second, the attacker introduces false votes into the system from the eSlate.

Changing Votes The basic vote changing attack is simple: the (compromised) eSlate simply sends whatever CVR data it wishes to the JBC. This produces consistent electronic records but the VVPAT will not match. It is not clear how serious a problem this is from the attacker's perspective, because in California the VVPAT is not routinely verified, except during the 1% manual tally. However, an attacker might wish to change votes in such a way that he could not be detected even in a 100% manual recount. This requires some social engineering to produce a VVPAT record that is incorrect but not noticed by the user.

A number of techniques are possible here:

- The simplest approach is for the eSlate to print out a VVPAT entry that matches the wrong candidate. If the voter does not notice, then the attacker has won. If the voter does notice, the eSlate simply allows them to change their vote, as if they had simply made an error, this time allowing the vote to go through. Everett's results indicating that few people detect changes in DRE confirmation screens [10] suggest that this attack has a high success probability.
- Another approach is to present the correct vote data on the screen, wait for the user to accept their ballot, and then display the UI indicator that shows their vote was accepted but not print "BALLOT ACCEPTED" on the VVPAT. Once the voter walks away, the VBOx prints "BALLOT REJECTED" indicating that the ballot was spoiled and then prints a new ballot of the attacker's choice. The instructions for using the Hart system only tell the voter to wait for the display to indicate done, not to wait for the VVPAT to scroll forward.
- The converse of this approach is to present the correct vote data in the VBOx, but when the user accepts the ballot, print "BALLOT REJECTED" instead and then quickly scroll the paper forward. Once the voter is likely gone the eSlate prints a new VBOx record with the candidates of their choice and mark that "BALLOT ACCEPTED". This is clearly possible in principle but we have no data on whether voters or poll workers will notice it. Even if they did, it would be hard for a voter to prove the problem to a pollworker because the VVPAT would have scrolled past.

In any case, the data recorded on the VVPAT would match the data recorded in the electronic records, with the result that the attack would not be detected by an audit. As with the eScan, SERVOs integrity checks would not detect the compromise of the eSlate (see Issue 11).

Electronic Ballot Stuffing An attacker who was not concerned with altering the total number of cast votes could also use a compromised eSlate to introduce false votes, as described in Issue 8 or by using the modem port to introduce access codes, as described in Issue 4.

As discussed above, we presume the attacker would want to minimize his vulnerability to detection. There are three issues here:

- Arranging that the VVPAT matches the electronic records.
- Avoiding detection by voters or poll workers.
- Making it difficult for an auditor to determine which votes were fake.

In order to satisfy the first requirement, the eSlate must print out the attacker's chosen ballot result on the VBO printer. Otherwise, the one percent manual recount might detect the cheating. However, because the VBOx is completely controlled by the eSlate software (Issue 33), this is straightforward. This leaves us with the second requirement, avoiding voter or poll worker detection of the attack. The general principle here is to print out the fake vote data as part of the process of clearing the VBOx screen after the voter leaves. This minimizes the chance of detection.

Note that this attack produces a situation in which the total number of votes in a precinct exceeds the total number of voters. This is easy to detect during the official canvass, provided that counties compare the number of votes cast to the number of voters. If it is detected, some attempts might be made to repair the damage by removing the fake votes. If the voter codes are generated using the modem technique of Issue 4, then they will presumably all be next to each other in the audit log and easy to to suppress. However, if the compromised eSlate sometimes swaps the access codes for legitimate and fake votes, then the repair problem becomes quite challenging.

The situation is more complicated if only the eSlate is compromised. However, it is still possible to make it very difficult for an investigation to determine which votes were the fake ones. The attack is as follows.

1. The eSlate waits for a voter to finish voting.
2. The eSlate generates a fake CVR.

3. The eSlate generates a random number r, either 0 or 1.
4. If $r = 0$, the eSlate prints the fake CVR entry to the VVPAT. immediately following the voter's genuine vote. Otherwise, it will be printed later.
5. The eSlate waits for a second voter to come, call that voter's access code X.
6. If $r = 0$, the eSlate immediately sends the fake CVR to the JBC with the new access code X. Otherwise, it will be sent later.
7. The voter votes and the eSlate sends the correct CVR to the JBC and prints out the correct CVR entry to the VVPAT.
8. If $r = 1$, the eSlate sends the fake CVR to the JBC with access code X and then prints out the fake CVR entry to the VVPAT.

The result is that there is a pair of votes, both of which have access code X, but the order will vary. Half will have the legitimate vote first while the other half will have the fake vote first. This will be consistent between the VVPAT and the electronic records.

This attack could be detected by examining the electronic records, although we found no evidence that the Tally code makes any attempt to detect multiple uses of the same access code. Even if this case were detected and even if the election officials knew exactly how the attack was designed, they would be unable to distinguish between the legitimate votes and the fake votes that could occur in either order. The election official would be forced to either accept both votes or disregard both votes, either of which could disenfranchise the voter whose access code was abused in this fashion.

7.2.3 Voting in Other Precincts

If we assume that an attacker is not concerned with producing a set of electronic records which can survive auditing, other attacks are practical. If an attacker controls either a JBC or an eScan he can simply add CVRs of his choice to the MBB. Because each CVR contains its own precinct ID (Issue 27), the attacker may be able to inject MBBs into the system which contain votes for any precinct of his choice, thus affecting a wider area than his own precinct.

The difficulty with this attack is that it creates an inconsistency between the vote counts for each precinct and the number of recorded voters. To the extent to which records are kept of which MBBs and devices are assigned to each precinct, it should be possible to determine the rogue devices/MBBs and suppress the bogus votes. We do not know the extent to which such checks are in fact done.

7.2.4 Denial of Service

Subverted machines can also be used to mount a variety of denial of service attacks. In the simplest such attack, the machine could simply be programmed to crash, misbehave in confusing ways, or operate extremely slowly. The scope of this attack depends on the number of machines compromised (see below for discussion of compromising multiple machines). This could block or at least severely impede an entire election.

More sophisticated attacks are also possible. Machines could be programmed to detect the voting patterns of voters in the precinct in which they are installed and malfunction if the voting patterns are unfavorable to the attacker. Machines could also be programmed to selectively fail for voters who voted a certain way, thus potentially causing them to give up. Once the machine is subverted, it is running software of the attacker's choice and so any of these behaviors are simply a matter of programming.

7.3 Subverting all DRE Devices in a Precinct

Because Hart's polling place architecture involves multiple machines connected in a network, we should ask the extent to which it is possible for an attacker who has subverted one machine in the polling place to subvert others.[1]

This attack is trivial for a poll worker or an attacker who breaks into the polling place (or other storage area) the night before the election. It is also trivial for an election official working in the warehouse. First, he is likely to have unattended access to all the machines. Second, he can subvert the JBC (using Issue 1) and then use that to take over all the remaining eSlates (using Issue 2).

The problem is more difficult for a voter, because he only has access to a single eSlate. However, if the attacker can cause the eSlate to impersonate a JBC (see Section 6.2), then he can use the eSlate to subvert the other eSlates in the precinct. This is made somewhat difficult by the JBC's being the bus master; however, we believe it is potentially possible, as described in Section 6.2.3.

Using the JBC write-in report vulnerability of Issue 6 it is possible to escalate access from an eSlate to the JBC, but only at the end of an election, potentially compromising the final MBB data or future elections.

7.4 Subverting all the Devices in a County

While subverting the devices in a single precinct is attractive for the attacker, it is also limited for reasons indicated in Section 7.2.3. A more powerful attack is to take over all the devices in an entire county. We believe that such an attack can be mounted by a poll worker, voter, or an attacker who has broken into the polling place/storage area before the election.

1. Subvert a single polling place device, whether eSlate, JBC, or eScan (see Section 6.1).
2. Install new firmware on that device. This firmware operates like the original software except that it contains an exploit for one of the SERVO issues described in Issue 13.
3. When SERVO connects to the compromised device after the election (to backup and/or verify), our firmware attacks SERVO and installs new software on the SERVO machine.
4. When each new device is connected to SERVO, our program installs compromised firmware on that device (again, see Section 6.1). Note that this happens immediately prior the next election when SERVO is used to zero the counters.

At the conclusion of this attack, most or all eSlates, JBCs, and eScans in the county would contain malicious firmware under the control of the attacker.

Effectively, this is a multi-stage, multi-platform worm, with the exploit for each stage being embedded as a payload in the stage before it. Writing such a worm is well within the capabilities of the average malware writer.

This attack allows an attacker who compromises a single device in one election to escalate up to an attack on all the devices in a county by the time of the next election. For instance, one could compromise a device in the primary election and have complete control of all devices in the county in time for the general election.

Once the attacker has control of all the devices in the county, he can of course mount attacks on vote integrity and secrecy that are difficult to detect and/or repair, including all of the attacks described in Section 7.2.

We have tested the following portions of this attack:

- Installation of new firmware on an eScan to which we have physical access.
- Remote compromise of SERVO from an eScan during the firmware verification process.

[1] This isn't a relevant issue for the eScan because they're not networked inside the polling place and there are likely only one or two eScans per polling place.

We have not produced an end-to-end demonstration of this attack, however delivering a payload of one's choice in the type of attack we mounted on SERVO is a well-understood problem. We have not demonstrated the ability to change the software on an eSlate or JBC. However, the installation of new firmware on the other eScans in the county from SERVO is essentially similar to the firmware installation we have already demonstrated.

In addition to this attack, note that if SERVO is installed in a network with the the other Hart back-office software, an attacker might employ Windows vulnerabilities to take over those machines from SERVO, leading to compromise of all county machines.

CHAPTER 8

Detection and Recovery

Because each vote in the Hart system produces records in a number of locations, in principle many of the less complete attacks can be detected and in some cases repaired. For instance the vote injection attack described in Section 7.2.2 changes the aggregate vote count, which could be detected simply by comparing the number of voters reported by a precinct against the number of votes reported by the precinct's voting machines. However, the extent to which the measures that are available are actually used is unclear and may vary dramatically.

The list of potential independent records which should be reconcilable includes, for the eScan:

- The voter book indicating the number of voters who voted.
- The number of paper ballots issued
- The actual paper ballots processed by the eScan
- The eScan's MBB
- The eScan internal logs

And for the DRE system includes:

- The voter book indicating the number of voters who voted.
- The JBC records of the number of issued voter codes (DRE only)
- The JBC's MBB
- The JBC's internal logs
- The eSlate's internal logs
- The VVPAT

In principle, any inconsistency between any of these records can be detected and potentially corrected. In practice, inconsistencies can be difficult to detect.

8.1 Detection

In this section we consider methods of detecting discrepancies. In the next we consider methods of recovery.

8.1.1 The One Percent Manual Recount

California election law requires that one percent of the ballots be manually recounted. In the case of eScan or Ballot Now centrally counted ballots, this means manually assessing each opscan ballot and comparing the count against that reported by the optical scanner. In the case of DRE voting it means comparing the VVPAT against the records from the JBC's MBB. 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.

The effectiveness of the one percent recount is strongly dependent on how discrepancies are treated. There are at least three plausible possibilities:

- Treat the manual results as the correct results and discard the electronic results.
- Investigate the precinct where the errors occurred.
- Do an investigation or a recount of a larger fraction of the votes, escalating up to a full paper recount.

It appears that in at least some counties the first option is taken [1]. While this is potentially sufficient to detect cheating by the precinct voting equipment, it is likely to be insufficient to recover from it outside the precincts which are actually recounted. Consider an attacker who increases the vote count for his candidate by 5% in 25% of precincts. Such an attacker has a .25 probability of having one of his precincts audited, but if the cheating in the other precincts remains undetected, he will have changed the vote totals by nearly 1.24%, having lost only a small fraction of his influence from the recount.

The other extreme is to take the 1% recount as a tripwire and follow up any discrepancies by escalating to a more significant recount, possibly up to a complete manual recount.[1] The difficulty is in deciding how sensitive the tripwire and how aggressive the escalation should be. If even minor discrepancies trigger full manual recounts, it is fairly easy for an attacker with limited resources to force the county into a full recount some reasonable fraction of the time. Moreover, strategies which are effective in detecting fraud are also susceptible to being forced into full recounts. It should also be noted that attacks that produce discrepancies are generally easier to mount than those that do not—much of our effort went into figuring out how to conduct attacks that leave behind no inconsistencies.

Note that in the optical scan case, escalation is substantially easier: the ballots simply need to be recounted with a scanner which is trusted not to have been compromised. This is inconvenient but far less inconvenient than a DRE recount, which requires manual recounting of the human-readable portion of the VVPAT—the bar code cannot be trusted for this purpose. One might imagine doing optical character recognition of the VVPAT, but this is a much more complicated problem. As far as we know, tools for this job are not presently commercially available.

8.1.2 Total Voter Counts

Attacks which simply inject false votes are in principle susceptible to simply comparing the number of voters to the number of electronically recorded votes. In both the optical scan and DRE cases, there is a physical record of the number of voters who signed in to the pools. In the case of the eScan, one could count the actual number of paper ballots used. None of these records are susceptible to electronic tampering[2] and they can therefore be used to detect vote injection, provided that these comparisons are actually done. We do not know how regularly they are performed or what the results of discrepancies are—as we expect that small-digit numbers of discrepancies will be common. Again, treating small discrepancies as a trigger for a recount creates an opening for a denial of service attack.

[1] Appel [3] provides a good description of the statistics here.

[2] If electronic pollbooks are in use, they may also be subject to attack.

8.1.3 Mechanical Log Comparisons

We note that several of the attacks we describe here produce distinctive signatures on the electronic logs produced in the various components. For instance, the eSlate vote injection attacks produce more votes in the MBB than the number of access codes issued to voters. Careful comparison of the MBB and the JBC access code records would reveal this. In principle SERVO provides reports that would allow an auditor to make these comparisons, however, we did not see any evidence it would do so automatically.

8.1.4 Tamper Seals

Finally, we may not have any evidence of actual cheating other than that a tamper seal is broken. This is obviously a reason to suspect that the sealed device has been tampered with, but it could also be a case where the seal was mishandled in some way. Determining which has happened is a difficult problem. In the case of all three of the polling place devices, an attacker with physical access can completely reprogram the device in a way that requires extensive forensics to verify, as discussed in Section 8.1.6.

The converse of how to respond to cases of tampered seals is the question of how effective such seals are. A number of different types of seals (plastic, wire, tape, etc.) are in use in California, and we have not made any systematic investigation of them; however, Johnston [22] reports successful and straightforward attacks on seals of the general type used in California elections. Therefore we must consider the possibility that seals can be broken without any obvious signs of tampering, especially with minimally trained pollworkers. See Section 3.5 for more on this point.

8.1.5 Parallel Testing and Logic and Accuracy Testing

One natural way to detect whether systems have been compromised is to test them to determine whether they are behaving correctly. Parallel testing involves selecting a random sample of DRE machines, taking them aside, and running a mock election on Election Day using the equipment. By preparing a known voting slate, one can compare the results from those machines against the inputs that mock voters entered. Typically, parallel tests are videotaped so that it is possible to go back and review any discrepancies. Parallel tests are one way to detect bugs or malice in DRE software, if the faulty software is widespread enough that the random sample is likely to pick at least one DRE that exhibits incorrect behavior. It is natural to wonder whether parallel testing might provide a way to detect large-scale viral attacks.

The reliability of parallel testing at detecting malicious code appears to be open to debate. The effectiveness of parallel testing is heavily dependent upon the details of how the testing is done. If malicious software can distinguish when it is being tested from normal operations, for instance by looking for mistakes that inexperienced voters would make but officials performing tests would not, then the malicious software can evade detection by behaving correctly when it is under test. Analysis of the effectiveness of parallel testing practices was outside of the scope of this study.

Ultimately, parallel testing becomes an arms race between attack designers and officials who plan realistic parallel tests. The defenders attempt to design testing procedures that mimic real elections as closely as possible, while we must assume the attackers will try to design methods to detect when they are being tested. It is not clear who has the advantage in this race. The problem with this kind of arms race is that it is difficult to know whether one is winning. Thus, there is a risk that an attacker might develop a secret way to defeat parallel testing, leaving the defenders with a false sense of security about election integrity.

Another way to thwart parallel testing would be to use a secret knock (a series of inputs known only to the attacker that would be unlikely to happen by chance) to control activation of the vote-stealing code. A secret knock could be used to activate the virus, though this would require the virus author to have conspirators who would need to access each of the voting machines where votes would be stolen. Alternatively, a secret knock might serve to deactivate the vote-stealing code, though this would require the help of an insider in the parallel testing process.

Parallel testing only defends against malicious software on the DRE system (eSlates and JBCs). It does not defend against malicious software at county headquarters.

Parallel testing is more effective at detecting attacks than at preventing them from disrupting the election. Suppose testing reveals that a small number of votes are recorded for the wrong candidate. If the test is conducted on or close to election day, there may not be enough time to determine the cause. As described below, it may be difficult or impossible to determine the correct vote totals if attack code is running on the machines on election day. Denial of service attacks present an even harder challenge. Officials have few recovery options if they discover shortly before the election that the machines will fail the next time they are used, and parallel testing on election day provides no warning of such a failure.

Finally, unless parallel testing is performed on a very large number of machines, it will have a low probability of uncovering attacks that are directed only at specific precincts or election conditions. Other mitigation strategies must be applied to control these risks.

All in all, it is difficult to predict with any degree of certainty how effective parallel testing will be. We would prefer a defense that we can be confident will detect problems over one whose efficacy is open to debate. Despite these limitations, parallel testing may still have value at detecting viral attacks. And, of course, we do not deny that parallel testing may have other benefits that are outside the scope of this study. We leave it to others to analyze the cost-effectiveness and appropriateness of parallel testing.

8.1.6 Firmware Forensics

In cases where a unit is suspected of having been tampered, it is natural to attempt forensic examination of the unit in order to determine whether it is running the correct firmware. As we discussed in Section 6.3, Hart provides techniques for running these checks remotely but they are ineffective.

The only plausible technique for running such forensic checks would be to directly examine the memory cards on the device, thus bypassing the potentially compromised software. However, a clever attacker could defeat such checks as well. The approach would be to retain a copy of the original firmware. When the "close elections" routine is run, the compromised firmware would scrub itself off of permanent storage and then restore the original firmware. This would defeat even fairly sophisticated forensic techniques.

A countermeasure to this attack is to set aside a random sample of devices before the election so that any attack software present will be preserved for analysis. Note that this technique does not detect malware installed on Election Day.

8.2 Recovery

As noted above, recovering from a clear discrepancy with an optical scan system is straightforward: rescan with a new scanner. eScan MBB compromises can potentially be recovered simply by auditing the eScan directly, although it isn't necessarily straightforward to determine which of the two is wrong.

Recovering from DRE compromises will be more complicated. If the VVPAT doesn't match the electronic records, then it is natural (and perhaps legally required) to use the VVPAT as the definitive record—although it too may in fact be incorrect. However, if they match, the problem is more difficult. The choices are effectively to abandon and rerun the election or to try to forensically determine what really happened.

The forensics option is problematic in two respects. First, it starts from the premise that the system is under attack from an attacker with limited capabilities who for some reason cannot control all electronic records. Inconsistencies are evidence of his lack of control over one or the other device and so reconstructing the real history is a matter of figuring out which device was compromised. This is a difficult proposition and bears with it the inherent risk of identifying the uncompromised device as compromised and therefore accepting the bogus set of results rather than the real ones.

This risk is enhanced if the forensic procedures are published in advance and the attacker therefore has an opportunity to tune his attack to counter them.

The second issue is that such a forensic investigation depends on maintaining more extensive and tamper-resistant records which tend to compromise the secrecy of the ballot. To consider two examples from the current system:

- The VVPAT is designed to be tamper resistant but inherently provides a linear record of all of the votes. This record can be easily mapped to voter codes (cf. Issue 35).
- In Issue 25 we observed that the votes are stored on the MBB in a way that allows reconstructing the voter code–CVR mapping. This represents a ballot secrecy issue but is also key to allowing recovery from the attack described in Section 7.2.2.

One can imagine even more aggressive record keeping measures, such as a separate hardware-based tamper-resistant record of every piece of user input. This would of course allow much better recovery, but at the expense of severely compromising ballot secrecy if an attacker obtained the record. Thus, there is an inherent tension between the ability to recover from compromise and the secrecy of the ballot. It may not be possible to recover from any significant attacks purely on the basis of electronic records.

This is not to say, however, that the Hart system represents an optimal trade-off in that design space, as it has both issues with ballot secrecy and minimal recovery capabilities.

CHAPTER 9

Recommendations for Future Analysis

The major difficulties we faced in this work were:

- Sharply limited time
- Inadequate information about procedures
- Insufficient access to materials
- Difficulty in testing hypotheses about system operation

9.1 Time Limitation

Although the project started May 31, the delivery of the Hart source code was delayed. We first obtained the source code code on Monday June 18. This left us with just over a month to perform our analysis and complete our report, which was due July 20. This timeline was extremely tight and forced us to strictly triage our investigation.

In particular, the short time combined with the coordination difficulties described in Section 9.4 prevented us from verifying most of the issues we discovered in the code. This is extremely problematic and should be a major focus of any additional effort.

As explained in Section 2, it is unrealistic to believe that any project of this type will produce an exhaustive audit of all the vulnerabilities in a system; we believe that a significantly more thorough job could have been done with more time.

We recommend that future projects provide the teams with at least three months to complete their work.

9.2 Inadequate Information about Procedures

The impact of a number of the issues we discovered depends significantly on the procedures followed in actual use of the system. For instance, Rally and Tally clearly have the capability to communicate results via the Internet or modem; however, the extent to which this capability is used in California—if at all—is still unclear to us.

It appears that many of the procedures vary substantially between counties, but we were generally instructed that these procedures were out of scope.[1] This made it very hard to assess which issues were most important and to focus the appropriate amount of attention accordingly.

This lack of information about procedures extends to ordinary election procedures. We recommend that in the future teams be allowed to observe a real election using the chosen equipment

[1] We were able to directly obtain some answers about procedures in Yolo County. We appreciate the assistance of Freddie Oakley and Tom Stanionis.

and that they have direct access to vendor representatives and county election officials to answer questions about procedures.

9.3 Insufficient Access to Materials

This project was significantly hampered by incomplete materials. Materials we were missing include:

- Detailed specifications for the network protocols used by the system.[2]
- The tooling that Hart uses to upgrade firmware on their own devices.
- A working build environment that would have allowed us to build our own binary versions of the Hart software.[3]
- Source code for significant portions of the system, such as the embedded MQX real-time operating system used by eSlate and JBC. It is our understanding that Hart owns and controls MQX [7].
- Windows systems configured with the back-end Hart applications as they should be configured in a genuine election environment.

Lack of access to build environments was particularly problematic for a number of reasons:

- Reading source code is generally a fairly inefficient method for determining how a program works. Running it, especially under a debugger, is more convenient, but only if you can build your own binaries so you can do source rather than binary debugging.
- A number of the issues we discovered involved detailed memory exploits which would have been easier to investigate had we had been able to to build our own versions of the software. Eventually we were able to acquire a copy of the binaries and resorted to binary level analysis.
- A number of our testing and attack tools required imitating one Hart component talking to another. In cases where this did not work it was very difficult to discover where the bug in our code was because the Hart component would generally just be silent. Were we able to build our own binaries and run them under a debugger we quickly would have been able to determine the issue. In addition, we could have used the Hart system as a basis for our own tools, thus greatly reducing the required effort level.
- Source code analysis tools like Fortify SCA are designed to process the code in the same build environments that the compiler does. In order to use these tools at all we had to mock up our own build environments, which was time consuming and may not have been representative of the real environment. In addition, we were not able to do this for all the components of the Hart system.

Some of our questions about the system still remain unanswered due to lack of the ability to build our own binaries.

We recommend that any future projects require that the teams be supplied with a full vendor development environment as well as all tools used by the vendor for maintenance, even if they are not supplied to ordinary customers.

[2]We finally received a small portion of this the week of July 1st, but most of our understanding came from reviewing the source.

[3]We understand that the tooling were finally sent to the Secretary of State on July 18, despite our earlier requests for these resources. As of the completion of this report on July 20th, we have not received them.

9.4 Difficulty in Testing Hypotheses about System Operation

Work on each system under study was divided between three teams:

- A documentation team
- A source code team
- A "Red Team" responsible for trying attacks

For the Hart system, the documentation team was located in Berkeley, the source team in Menlo Park, and the Red Team in Sacramento. The geographical separation plus the necessary security restrictions made it very difficult to transfer information among the three teams. For example, when we wished to transfer our attack tools to and from Sacramento, they had to be encrypted, burned onto a CD-ROM and shipped, with the key transferred via telephone. This, plus the ordinary difficulty of knowledge transfer across geographically separated teams, unavoidably created significant delays in testing hypotheses about potential issues.

We recommend that in the future there be one team which is responsible for both analysis of the source and mounting attacks and that all the work be done in a single location with access to both the source and the hardware. This would allow a much tighter feedback loop between hypothesis and test and would accelerate progress.

CHAPTER 10

Summary of Findings

Although we had only limited time to review the source code of the system, our review nevertheless uncovered what we believe to be a number of significant security issues. In many cases the Hart system does not incorporate defense-in-depth principles, which may allow individual attacks to be escalated up to much broader attacks.

The Hart software and devices appear to be susceptible to a variety of attacks which would allow an attacker to gain control of some or all of the systems in a county:

- The Hart eScan, eSlate, and JBC devices incorporate an unsecured management capability. We believe that given brief physical access to an eScan, eSlate, or JBC device, an attacker can subvert it and overwrite the existing software with malicious software of his choice.
- These attacks could be mounted by a poll worker or possibly by a voter while in the process of voting. The effects of such an attack are essentially permanent; once malicious software is loaded onto such a device, there is no realistic way to remove it.
- Subversion of single polling place devices can be used to mount a variety of vote forgery and ballot stuffing attacks.
- The mechanisms provided by Hart for detecting device subversion appear to be easy to bypass and therefore system subversion is likely to go undetected.
- The Hart back-end SERVO software contains multiple buffer overflows which appear to be remotely exploitable by a single compromised polling place device. We have exploited one of these in our test environment and used it to install software of our choice on the SERVO machine.

By combining the above attacks, a malicious pollworker could subvert an eScan, through that SERVO, and through SERVO all the machines in the county for the next election. We have tested what we believe to be the essential elements of this attack but not performed an end-to-end test. Furthermore, a malicious voter could subvert a single eSlate, through that SERVO, and through SERVO all the machines in a county for the next election. We have tested some but not all of the elements of this attack.

Beyond direct system compromise, we found that Hart's management of ballot and vote data is vulnerable to several attacks:

- Hart's cryptographic key management requires a county-wide symmetric key which is stored on vulnerable field devices. This key can be obtained by an attacker with brief physical access to an eScan or JBC.
- Compromise of this single key would allow an attacker to forge both ballot information and vote results.
- We found multiple avenues for compromising voter privacy, enabling both vote buying/coercion and wholesale information gathering attacks.

This list does not include all the issues discovered during our review and there may be other issues that would be uncovered with further review. We encourage the Secretary of State to undertake such a review.

We stress that due to limited time and access to Hart equipment, we did not attempt to validate all of the above issues. In the body of the report we clearly indicate the validation status of each issue. We encourage the Secretary of State and Hart to attempt such validation.

Some of these issues can be mitigated with stricter polling place procedures. Others may be repaired with minor modifications to Hart's systems, while yet others may require significant redesign. Providing a complete assessment of mitigation strategies was out of scope of this review, but we encourage the Hart and the Secretary of State to study these issues.

We have deliberately avoided addressing the broader issue of whether or how this system should be used for voting in California. Making that judgement requires assessing not only the technical issues described in this report but also the procedures and policies with which the system is used.

Bibliography

[1] Judy Alter. Report on the 1% manual recount for special election, Nov. 2005 L.A. County, July 2006. `http://www.bbvforums.org/forums/messages/2197/Manual_recount_report-33060.doc`.

[2] Anonymous. Once upon a free(). *Phrack Magazine*, 57(9), August 2001. `http://www.phrack.org/archives/57/p57-0x09`.

[3] Andrew W. Appel. Effective audit policy for voter-verified paper ballots in New Jersey, March 2007. `http://www.cs.princeton.edu/~appel/papers/appel-nj-audits.pdf`.

[4] Brad Arkin. Securing the eSlate Electronic Voting System: Application Security Implementation, January 2005. `http://www.hartic.com/files/HART_SYMANTEC_SECURITY_REPORT_White_Paper.pdf`.

[5] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *Proceedings of Crypto 1996*, volume 1109 of *LNCS*, pages 1–15. Springer-Verlag, August 1996.

[6] blexim. Basic integer overflows. *Phrack Magazine*, 60(10), December 2002. `http://www.phrack.org/archives/60/p60-0x0a.txt`.

[7] Compuware Corporation, Columbus, OH. *Direct Recording Electronic (DRE) Technical Security Assessment Report*, November 2003. `http://www.sos.state.oh.us/sos/hava/compuware112103.pdf`.

[8] Tim Dierks and Eric Rescorla. The transport layer security (TLS) protocol version 1.1. RFC 4346, April 2006. `http://www.ietf.org/rfc/rfc4346.txt`.

[9] Electronics Industry Association. Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems (ANSI/TIA/EIA-485-A-98) (R2003). EIA-485, March 1998.

[10] Sarah P. Everett. *The Usability of Electronic Voting Machines and How Votes Can Be Changed Without Detection*. PhD thesis, Rice University, 2007.

[11] Ariel J. Feldman, J. Alex Halderman, and Edward W. Felten. Security analysis of the Diebold AccuVote-TS voting machine. In *2007 Usenix/ACCURATE Electronic Voting Technology Workshop*, Boston, MA, August 2007.

[12] Ryan Gardner, Sujata Garera, and Aviel D. Rubin. On the difficulty of validating voting machine software with software. In *2007 Usenix/ACCURATE Electronic Voting Technology Workshop*, Boston, MA, August 2007.

[13] gera and riq. Advances in format string exploiting. *Phrack Magazine*, 59(7), July 2001. `http://www.phrack.org/archives/59/p59-0x07.txt`.

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

[15] Hart InterCivic. *Judge's Booth Controller (JBC) Functional Specification*, 2005. Document 6000-050, Rev. 42-62B.

[16] Hart InterCivic. *eSlate Polling Place System Election Day Desk Reference*, 2006.

[17] Hart InterCivic. *Hart Voting System, Product Description, System 6.2*, 2006. Document 6000-060, Rev. 62A.

[18] Hart InterCivic. *Rally Operations Manual*, 2006. Document 6100-114, Rev. 23-62A.

[19] Hart InterCivic. *System for Election Records and Verification of Operations, Operations Manual*, 2006. Document 6100-102, Rev. 42-62B.

[20] Hart InterCivic. *Tally Operations Manual*, 2006. Document 6100-049, Rev. 43-62B.

[21] Hart InterCivic. *Voting System Use Procedures for California, Hart Voting System 6.2*, July 2006. Part Number 000255, Rev. C.

[22] Roger G. Johnston. Tamper-indicating seals. *American Scientist*, 94:515–523, November-December 2006. Reprint available at `http://ephemer.al.cl.cam.ac.uk/~rja14/johnson/newpapers/American%20Scientist%20(2006).pdf`.

[23] Michel Kaempf. Vudo malloc tricks. *Phrack Magazine*, 57(8), August 2001. `http://www.phrack.org/archives/57/p57-0x08`.

[24] Burt Kaliski. PKCS #5: Password-based cryptography specification version 2.0. RFC 2898, September 2000. `http://www.ietf.org/rfc/rfc2898.txt`.

[25] John Marchesini, Sean Smith, Omen Wild, and Rich MacDonald. Experimenting with TCPA/TCG Hardware, Or: How I Learned to Stop Worrying and Love The Bear. Technical Report TR2003-476, Department of Computer Science, Dartmouth College, December 2003.

[26] Elliot Proebstel, Sean Riddle, Francis Hsu, Justin Cummins, Freddie Oakley, Tom Stanionis, and Matt Bishop. An analysis of the Hart InterCivic DAU eSlate. In *2007 Usenix/ACCURATE Electronic Voting Technology Workshop*, Boston, MA, August 2007.

[27] David Rohde. On new voting machine, the same old fraud. *The New York Times*, April 27, 2004. `http://www.nytimes.com/2004/04/27/international/asia/27indi.html`.

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

[29] Stuart Schechter, Rachna Dhamija, Andy Ozment, and Ian Fischer. The emperor's new security indicators. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 2007.

[30] Arvind Seshadri, Mark Luk, Elaine Shi, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla. Pioneer: Verifying integrity and guaranteeing execution of code on legacy platforms. In *Proceedings of ACM Symposium on Operating Systems Principles (SOSP)*, pages 1–15, October 2005.

[31] Michael Shamos. Oral testimony, Technical Guidelines Development Committee (TGDC), public data gathering hearings, September 2004. `http://vote.nist.gov/PublicHearings/9-20-94%20Panel%202%20SHAMOS.doc`.

[32] Michael Steil. 17 mistakes Microsoft made in the Xbox security system. In *Proceedings of the 22nd Chaos Communication Congress*, December 2005.

[33] Trusted Computing Group. TPM v1.2 Specification Changes, October 2003. `https://www.trustedcomputinggroup.org/downloads/TPM_1_2_Changes_final.pdf`.

[34] Alma Whitten and J.D. Tygar. Why johnny can't encrypt: A usability evaluation of PGP 5.0. In *Proceedings of the 8th USENIX Security Symposium*, Washington, D.C., August 1999.

APPENDIX A

System Components and Versions

Source code for the following components was submitted by Hart InterCivic for review:

- Ballot Now, version 3.3.11
- BOSS, version 4.3.13
  - BossUtil, version 2.5.8
  - Translate DLL, version 1.8.2
- eCM Manager, version 1.1.7
- eScan, version 1.3.14
- eSlate, version 4.2.13
- HartLib, version 4.0
- JBC, version 4.3.1
- Rally, version 2.3.7
- SERVO, version 4.2.10
- Tally, version 4.3.10
- VBO Printer Firmware, version 1.8.3

We did not receive source code for the following components:

- The version of the embedded MQX real-time operating system maintained by Hart for use in the eSlate and JBC devices.
- The software tools that Hart uses to upgrade firmware on their devices.

Breakdown of the number of Source Lines of Code (SLOC) per component:

Component	*SLOC*	*Language(s)*
Ballot Now 3.3.11	64K	(C++)
Boss 4.3.13 (excluding PowerBuilder code)	35K	(C++)
eCM Manager 1.1.7	2K	(C++)
eScan 1.3.14	79K	(C++, C)
eSlate 4.2.13	22K	(C++)
HartLib 4.0	56K	(C++, asm)
JBC	23K	(C++)
Rally 2.3.7	7K	(C++)
SERVO 4.2.10	24K	(C++)
Tally 4.3.10	52K	(C++)
VBO Printer Firmware 1.8.3	2K	(C++)
(total)	366K	

Table A.1: The number of non-blank, non-comment source lines of code in each voting system component, as counted by David Wheeler's `sloccount` 2.26. All numbers have been rounded to the nearest thousand lines of code.