Software Review and Security Analysis of the ES&S iVotronic 8.0.1.2 Voting Machine Firmware

Alec Yasinsac David Wagner Matt Bishop Ted Baker

Breno de Medeiros Gary Tyson Michael Shamos Mike Burmester

Security and Assurance in Information Technology Laboratory
Florida State University
Tallahassee, Florida

February 23, 2007

Final Report
For the Florida, Department of State

Software Review and Security Analysis of the ES&S iVotronic 8.0.1.2 Voting Machine Firmware

Table of Contents

Section	Title	Page #
1	Executive Summary	3
2	Project Information and Background	4
3	iVotronic Operational Overview	9
4	Assumptions	17
5	Activities That are Specifically out of Scope for this Analysis	19
6	Findings	21
7	Security-Related Findings	36
8	Analysis of Hypotheses	45
9	Conclusions	53
10	Acknowledgments	53
11	Team Endorsement	54
12	References	55
Appendix A	CD13 Screen Shots	56
Appendix B	Technical Analysis of the PEB Virus Threat	57
Appendix C	Virus-Safe and Unsafe Operations	62
Appendix D	Passwords	66
Appendix E	Non-Pertinent Flaws	68
Appendix F	Analysis of Anomalous Audit Log Messages Regarding Voter PEBs	78
Appendix G	Anonymization of cast vote records in the ES&S iVotronic 8.0.1.2 firmware	82

Software Review and Security Analysis of the ES&S iVotronic 8.0.1.2 Voting Machine Firmware
Final Report

1 Executive Summary

On December 15th, 2007 the Florida Department of State (FLDoS) commissioned an independent expert review of the ES&S iVotronic 8.0.1.2 firmware, as documented in the Statement of Work [1]. The team, led by Florida State University's (FSU) Security and Assurance in Information Technology (SAIT) Laboratory, was commissioned to conduct a static software code review as part of the state's audit of the 2007 Florida Congressional District 13 (CD13) election between candidates Vern Buchanan and Christine Jennings. This report is the culmination of that review.

1.1 Administrative Overview

This report describes the findings after an intensive analysis. The subject code was delivered to the review team and active preparations began the day the statement of work was signed. Outside code review members arrived in Tallahassee within three days and intensive code review commenced. A relatively large team, whose members were chosen because of their complementary skill sets, performed the review. SAIT Laboratory members bring strong theoretical and applied information security and electronic voting credentials. Two non-SAIT FSU Computer Science faculty members contribute computer architecture, compiler, and hardware interface expertise. Three outside members with distinguished records in secure software, voting system security, and code review round out the team.

1.2 The Analysis' Scope

Our investigation was limited to the scope specified in the Statement of Work:

> *The sole purpose of this project is to conduct a scientifically rigorous static software analysis on the iVotronics version 8.0.1.2 firmware source code to determine and identify flaws, vulnerabilities or anomalies, if any, that may have potentially caused, contributed or otherwise created the higher than expected under-vote rate in the District 13 Race. [1]*

We focused our efforts on finding voting machine software problems that may have contributed to the CD13 undervote. We received all requested access to iVotronic terminals, PEBs, elections officials, ballot definitions, development engineers, and documentation. Where we needed additional hardware information to understand the software operation, we were given that data. We methodically examined undervote symptoms and followed the evidence to our findings. We considered possible causes hypothesized in the press and Internet sources, as well has others of our own design. We used standard software tools for manual code review and used static analysis tools to automate some of the analysis. In accordance with our plan, the team worked together throughout the intense code review cycle, cross-checking and corroborating hypotheses and findings. We documented our findings during the course of our work, and referred to our daily notes as we prepared this report. While there are no guarantees in this type of analysis of a system as complex as the iVotronic, we examined all aspects of the software that we believed may have contributed to the CD13 undervote.

1.3 Findings Summary

The team's unanimous opinion is that the iVotronic firmware, including faults that we identified, did not cause or contribute to the CD13 undervote. We base this opinion on hundreds of hours of manual code review complemented by automated static analysis and extensive study of the problem symptoms and the execution environment. We traced program execution from terminal

initialization, through voter selection, to ballot image creation, to ballot image collection. We also investigated the possibility of asynchronous system faults not associated with any particular phase of voting. Our investigation provided no evidence that an iVotronic software malfunction caused or contributed to the CD13 undervote.

We do not claim that these results extend beyond the scope of our investigation. We emphasize that these findings are neither an endorsement nor a repudiation of the iVotronic, the larger class of Direct Recording Equipment (DRE) systems, nor any other form of electronic voting system. We specifically do not contend that these systems are correct or secure beyond the specific opinions that we give herein. This report is concerned solely with the question posed to us regarding the cause of the CD13 undervote in Sarasota County in November, 2006, and we do not claim that these results extend to a broader context.

2 Project Introduction and Background

2.1 Report Organization.

This document represents the total project report. It contains all of our pertinent findings and conclusions and the technical analysis that supports these conclusions. The document is written in two parts. The public part (Sections 1-12 and Appendices A, B, C, and D) constitutes the public report in its entirety; it contains our findings and the analysis to support these findings and is intended for public dissemination. In accordance with the terms of the Statement of Work, we have avoided revealing proprietary information in the public part of the report, and we are careful to avoid revealing information that would describe how to attack an election. The public report stands on its own and reflects the totality of our findings regarding the CD13 undervote.

The private part consists of Appendix E (Non-Pertinent Flaws), Appendix F (Analysis of Anomalous Audit Log Messages Regarding Voter PEBs), and Appendix G (Anonymization of cast vote records in the ES&S iVotronic 8.0.1.2 firmware). Appendices E and F are confidential, as required by the Statement of Work, because they contain vendor-proprietary information; also, Appendices E and G are confidential, as required by the Statement of Work, because they contain information about potential defects that could not have caused or contributed to the CD13 undervote and thus that are not relevant to this investigation. We are providing Appendices E, F and G to the state to allow the state to thoroughly evaluate the iVotronics and to pass on pertinent information to the vendor that will facilitate future improvements to these voting systems.

As indicated in the Statement of Work, we provided some details to the FLDoS and the vendor during the course of our work. We emphasize that the public part of this report contains everything we learned during this review that is relevant to the CD13 undervote.

The main document first gives background information about the undervote observed in the CD 13 race, the investigation, the voting system, and our assumptions. We follow these by describing our findings and conclusions.

2.2 The Software Review Team

2.2.1 The Senior Investigators

2.2.1.1 Ted Baker. Dr. Baker is a Florida State University Professor of Computer Science. For thirty years he has conducted systems-related research and taught hundreds of technical classes regarding machine interactions. He is an expert in device drivers and hardware-software issues.

2.2.1.2 Matt Bishop is a Professor of Computer Science at the University of California at Davis. He

is an expert in secure software and electronic voting systems, having participated in several widely recognized electronic voting software systems code reviews. His computer security textbook, *Computer Security: Art and Science*, is the acknowledged benchmark against which all others related to this topic are measured.

2.2.1.3 Mike Burmester is an FSU Professor of Computer Science and a co-Director of (SAIT) Laboratory. He is a renowned expert in information security and cryptography, with over thirty year's research experience in computer security related issues.

2.2.1.4 Breno de Medeiros is a Florida State University Assistant Professor of Computer Science. He is an Information Security expert with extensive software experience.

2.2.1.5 Michael Shamos is Distinguished Career Professor of Computer Science at Carnegie Mellon University. He has performed over 115 electronic voting certification examinations for six states, including Pennsylvania and Texas. He frequently testifies before the US Congress and various state legislatures on electronic voting issues.

2.2.1.6 Gary Tyson is a Florida State University Associate Professor of Computer Science. He is an expert in computer architectures and compiler technology.

2.2.1.7 David Wagner is an Associate Professor of Computer Science at the University of California, Berkeley. Like Professor Bishop, he is an expert in secure software and electronic voting systems, having conducted several widely recognized electronic voting software code reviews.

2.2.1.8 Alec Yasinsac is a Florida State University Associate Professor of Computer Science, a co-Director of SAIT Laboratory, and is the lead Principal Investigator on this project.

2.2.1.9 The Statement of Work (SoW) listed Dr. Edward Felten of Princeton University as an initial team member. Dr. Felten made significant contributions to project planning and was invited to participate, but he ultimately did not join the full team.

2.2.2 Team Organization

2.2.2.1 Internal Team Structure and Operation. As detailed in the project plan, six team members (Baker, Bishop, de Medeiros, Tyson, Wagner, and Yasinsac) conducted hands-on code review. Two members (Burmester and Shamos) contributed to project plans, reviewed the process documents, and participated in report preparation. The final report reflects the team's cumulative and unanimous opinion.

2.2.2.2 External Communication and Coordination

2.2.2.2.1 Florida Department of State (FLDoS). As noted in the SoW, FLDoS was entitled to observe the code review process at their discretion; they chose to limit their interaction. FLDoS only interacted with the team at our invitation and they proved to be a valuable information resource, providing precinct reports, election configuration files, general election knowledge, and hardware demonstrations to support our analysis. Their support was consistently prompt and complete. The FLDoS placed no restrictions on our activities within the SoW.

2.2.2.2.2 Florida State University. FSU and SAIT Laboratory hosted the code review and provided invaluable analysis resources and administrative support beginning the first active SoW

day, extending through the holiday periods (including while the University was officially closed for both the Christmas and New Year's holidays) and into the new year. The spaces were ideal for this type of review and the resources were excellent.

2.2.2.2.3 Election Systems & Software (ES&S). ES&S was an active and effective information resource for this team. Two ES&S iVotronic software developers with intimate knowledge of and experience with the firmware spent one and a half days answering questions and accelerating our understanding of the software structure and flow. We subsequently conversed with these developers by telephone several times.

Additionally, an ES&S hardware specialist met with the team to clarify information and confirm our observations of the hardware architecture and hardware-software interactions. We also had subsequent telephone conversations with other ES&S hardware engineers to answer follow-on questions.

When we sought technical detail, documentation, or clarification, ES&S responded promptly and comprehensively. For example, when we sought compiler information, they provided a listing of source code and the corresponding assembly language side-by-side. These interactions were undoubtedly an important contribution to the project that facilitated our work, accelerated our progress, and heightened confidence in our findings. The vendor offered to provide equipment and resources to allow us to construct proof of concept demonstrations of our hypothesis, but the team declined this invitation. We address specific vendor input and interactions throughout the report.

2.3 The Investigative Process

In accordance with our project plan, the investigation began with a short collaborative planning phase. The team met in the SAIT Laboratory and spent several days examining code, documentation, and symptomatic evidence to understand the problem and to formulate an investigative approach. The resulting plan relied on parallel investigation of reliability and security issues that may have caused or contributed to the CD13 undervote. The team composition provided a natural investigative partitioning. Professors Baker and Tyson focused on hardware interaction, low level software, and architectural issues. Professor Wagner focused on security considerations, Professor Bishop and Professor de Medeiros investigated software faults and security issues, and Professor Yasinsac investigated gap issues not covered by the natural team partitioning. Early in the process we produced an extensive list of scenarios that might have resulted in the observed undervote, and this list formed our investigation to systematically rule out each scenario.

Each code investigator took two complementary research approaches in their specialized area. Each investigator conducted unrestricted code examination. They each spent time analyzing code and following their instincts, with no external limits imposed upon them. This leveraged investigators' analytic strengths and offered the opportunity to reveal subtle or non-intuitive faults.

Additionally, investigators carefully and collaboratively examined evidence and Sarasota-specific symptoms within their areas. Investigators received data from Sarasota that defined the environment, triggered symptom analysis, and validated configuration assumptions. During our investigation, we reviewed problem logs produced on Election Day by Sarasota County poll workers. We also reviewed published studies, press reports, and court proceedings that aided our review. These symptoms led to many observations that constitute the bulk of our findings.

2.4 The CD13 Undervote Details

The CD13 undervote has been the subject of several lawsuits, news articles numbering in the triple digits, and uncounted blog commentaries. While this produced a mountain of information about the

undervote, facts were elusive. We know that there are approximately 18,000 undervotes, which is more than 13% of the total CD13 vote and is three to ten times the average undervote in other races. There is no dispute that this undervote is abnormal and unexpected and that it cannot be explained solely by intentional undervoting.

The FSU team spent many hours investigating election related documents and information and documented many symptoms that might indicate possible causes. Among these, we noted that the abnormal undervote rate was present in both early and Election Day voting, with a higher undervote rate observed during early voting. The Sarasota Supervisor of Elections (SoE) responded to complaints from voters about problems voting in the CD13 race during early voting by asking poll workers to remind voters to review their ballots. The undervote subsequently diminished on Election Day, suggesting that raising heightened voter attention may have reduced the undervote rate. Precinct logs, in which poll workers make notes, show repeated entries that poll workers reminded voters to give special attention to the CD13 race.

Recorded voter complaints also offered information that contributed to the software analysis process. Precinct logs indicate that voters offered three classes of pertinent comments.

1. The voter selected a candidate in the CD13 race, but claimed that the selection did not appear on the summary page.
2. The voter did not notice the CD13 race at all until it was shown as an undervote on the summary page.
3. Many machines responded slowly (five seconds or more) or not at all, to voters' touches.

These three reported symptoms suggested many hypotheses regarding possible software faults. We investigated numerous other reported symptoms as well. For example, during our review of the Sarasota iVotronic event logs (audit logs), we noticed an anomalous event log entry containing the message "Invalid Vote PEB". We hypothesized causes for this event and traced through the code to find its cause, as detailed in our findings below. We similarly traced other symptoms that we discovered through review of evidence and records such as:

- Event logs
- Ballot image files
- Ballot definition files
- Polling place logs
- Newspaper articles
- Court documents, particularly expert reports
- Blog entries
- Standard software flaw guides
- Standard security flaw guides
- Independent Test Authority reports
- Other historical documents

During our work, we analyzed many hypotheses. These activities included exercising iVotronic terminals, testing personal hypotheses, judging touch and display properties, analyzing machine timing and performance characteristics, and confirming configuration assumptions. The team was given two demonstrations of the iVotronic machines, and several team members later had the chance to experiment with iVotronic equipment configured with the ballot style used in Sarasota

County in November, 2006. We returned to the hardware several times to compare the machine behavior to our analysis of the source code.

2.5 Speculated Causes for the Undervote

Several papers have been written proposing theories about what may have happened in the CD13 race. We present a few representative theories in this subsection. They are not exhaustive, nor are they mutually exclusive. It is theoretically possible that all of these factors contributed, that none of them did, or that any combination of them did.

2.5.1 Machine or Software Malfunction. This is a general category that includes total machine failure; machine problems that created difficulty for voters; and subtle, even undetectable faults that may have contributed to the undervote. Some political and computer science experts have raised the possibility that a software fault or intentional software intrusion may have caused or contributed to the undervote. For instance, computer security and electronic voting expert Dr. Dan Wallach identified a number of hypotheses regarding potential software or system malfunctions that may have led to the CD13 undervote [2]. The FSU team considered Dr. Wallach's hypotheses in our analysis process.

Similarly, in Ms. Jennings' contest to Congress [3], her team hypothesizes that a software error may have interfered with the transfer of information between the volatile memory where votes are stored during the vote selection process and the non-volatile memory where the votes are retained for extraction at poll closing. The team specifically investigated each of the hypotheses mentioned above, as well as others identified by the team, during this investigation.

2.5.2 Voter Discontent. Another possibility is that voter apathy may have contributed to the undervote. Some argue that the negative tone, both in the primaries [4] where reportedly neither candidates' opponents endorsed the eventual winner, and the subsequent bitterly contested general campaign, resulted in voter apathy in this race. A ballot review conducted by a local newspaper in early December [5] and cited by Electionline.org [6] supports the theory that voter apathy may have combined with the ballot design issues and thereby increased the magnitude of the undervote in the CD13 race. The newspaper article quotes one usability expert as suggesting that straight party voters may be looking for party affiliation rather than candidates, and thus may be less likely to realize that they did (or did not) vote for a specific candidate.

> *An analysis of the election published this week in the Herald Tribune further supports the findings of the Dartmouth study. The newspaper analyzed every vote cast and discovered that loyal party voters — both Republican and Democrat — were largely responsible for the undervote in Sarasota. Nearly 60 percent of the 18,000 undervotes in the race came from voters who otherwise voted along party lines.[5]*

Voter discontent does not explain the difference among the undervote percentages in mail ballots, surrounding counties, and the machine recorded votes. It is possible that voter demographics between more and less densely populated areas may account for part of these differences, but it is widely accepted that these factors do not account for all of the difference.

2.5.3 Event 18 Correlation. An academic study of Sarasota event logs [7] revealed a correlation between the undervote rate on specific machines and occurrences of a specific anomalous event in the audit logs for those machines: specifically, the "Invalid Vote PEB" message, which has also been identified as "Event 18" [7]. In the first week of January, before the report had been released, we had noticed the Event 18 messages in the event logs, investigated them, and established that the causes were (1) a software bug that did not affect the recording or tallying of votes for the Voter

PEB-Normal Ballot anomaly and (2) poll workers taking a specific action for Event 18 variations. The correlation noted in the paper is not due to any fault in the iVotronic firmware. Our detailed findings supporting this are reported in Section 6.2.1.2 below and in Appendix F.

2.5.4 Ballot Design Issues. Ballot design issues represent another possible cause that emerged soon after the election. Many people speculated that placing a race with only two candidates on the same page along with a race that has many more candidates, without a prominent race title block, could distract some percentage of voters. This theory may also explain voter complaints that they "...did not see the Jennings race..." until they noticed it on the summary page. A recent study argues that ballot design issues are the most likely undervote cause [8] in the CD13 race, a result also supported by an informal experiment reported in Electionline.org [6]:

> *Ted Selker, director of the Caltech/MIT Voting Technology Project, set up voting machines on the MIT campus and asked random people to vote. Selker told the paper that initial results indicate that the two-candidate race is missed 60 percent of the time when it's dwarfed by the list of gubernatorial candidates.*

A clearly confusing aspect of the Sarasota ballot is that the first page contained two long headers separated by a straight line, followed by a large, important race. This structure may pre-dispose voters to a pattern of two long headers separated by a line followed by a large race, leaving the CD13 race unnoticed on the second page. Screen shots of these pages are provided in Appendix A.

Another study [7] questions whether the ballot design theory can explain all the undervotes. That study hypothesizes that machine failures associated with the "Invalid Vote PEB" message (Event 18) may have contributed to the high undervote rate. Our analysis and code review conclusively refutes the Event 18 hypothesis, as detailed in our technical findings below. Unfortunately, neither statistical analysis nor code review can conclusively confirm or refute the ballot design hypothesis itself. Our findings are consistent with, but do not confirm, the ballot design explanation.

2.5.5 Age Variations. In December 2006, a Sarasota newspaper conducted an analysis examining the correlation between age and CD13 undervotes [9]. They found that in "...precincts where the median age was greater than 65, the undervote rate in the congressional race was 18 percent, 40 percent higher than in younger precincts." Some suggest that the undervote-age correlation supports the ballot design hypothesis and refutes most machine-related hypotheses since software cannot detect a voter's age. It may also explain the correlation between undervotes and voters associated with one party or the other. We attempted to identify fault hypotheses to explain this correlation, but we were unable to construct any machine-related fault hypotheses that would explain this observed effect.

3 iVotronic Operational Overview

The ES&S iVotronic is a highly configurable voting system. It provides a wide variety of configuration options that can be used to customize its operation according to local requirements. Consequently, many of the iVotronic's configuration options were not exercised in the CD13 race. Here we provide an overview of the iVotronic architectural and operational characteristics, as it was used in the Sarasota County CD13 race.

3.1 The iVotronic Election Process

The iVotronic voting process generally includes the following phases: (1) election generation and setup, (2) preparing Personal Electronic Ballots (PEBs) and removable non-volatile memory cards,

(3) initializing iVotronic terminals, (4) opening the polls, (5) voting, (6) closing the polls and accumulation, and (7) tabulation. We discuss each of these in the following subsections. While we will discuss some aspects of the iVotronic hardware in this section, we save many details about the hardware for a later section.

3.1.1 Election Generation. This early phase is largely outside the scope of the firmware code review. Our understanding is that the election staff creates the election definition files on the vendor's election management system (called Unity), generates a unique election identification code, defines the contests, and identifies the candidates in each contest. The staff exercises and tests these settings before settling on a final election configuration.

3.1.2 Preparing PEBs and CF Cards. The iVotronic stores and retrieves various data using two easily removable storage devices: the Personal Electronic Ballot (PEB) and a Compact Flash (CF) card. The PEB contains the election definition files produced by Unity for the precinct where the PEB will be used, as well as the election identification code. PEB initialization installs this information on the PEB's non-volatile storage. The CF contains audio files on machines configured for disabled voters (ADA) machines and information to identify the election. All CF cards are loaded with bit-for-bit identical information during the preparation stage. The election staff inventories, initializes, and tests these storage devices between elections, often just a month or so prior to Election Day.

3.1.3 Initializing iVotronic Terminals. Two election initialization operations relative to the iVotronic terminals are: (1) updating the firmware (when necessary) and (2) clearing the on-board memory. Firmware updates do not occur in every election cycle so firmware may persist between elections. The clear and test procedure erases information associated with past elections, initializes the persistent storage on the iVotronic terminals, and prepares the iVotronic terminal for use in the next election.

3.1.4 Opening the Polls. On Election Day, a poll worker opens the polls by inserting a PEB into each iVotronic terminal. This makes the iVotronic terminal ready for voting. A de facto standard practice is to use one PEB (called a "Master PEB") to open all terminals in a polling place. Each polling place has its own Master PEB. Master PEBs are ordinarily not used for anything other than opening and closing the polls; they are set aside, unused, for the rest of Election Day.

3.1.5 Voting. After each voter demonstrates her eligibility to vote and signs the sign-in roster, a poll worker accompanies her to an iVotronic terminal, inserts a PEB into the terminal, responds to an administrator screen (e.g., to select the proper precinct, for early voting or multi-precinct polling places), and then removes the PEB. Thus, the PEB serves the purpose of activating the machine to allow a single voter to cast a single ballot. The voter never handles the PEB. When the poll worker removes the PEB, a voter administration screen appears and the voter selects her desired options (e.g., the language in which to view the ballot). When the terminal displays the ballot, there are only two valid voter actions until the voter reaches the final summary page: (1) select or deselect one or more candidates on the page, or (2) page right or left (meaning to move forward or backward, respectively, through the ballot). Once the voter reaches the final summary page, she has three options: (1) select a race to re-vote, (2) page right or left, or (3) cast the ballot. The voter may cast her ballot any time after reaching the final summary page.

Some iVotronic terminals provide extra features designed to enhance their accessibility. These machines are known as ADA-capable terminals; the acronym refers to the Americans with Disabilities Act. Not all iVotronic terminals are ADA-capable. Non-ADA terminals have a standard ballot presentation style that utilizes color to highlight locations on the screen that can be activated by touching them, such as "page right" and "page left". ADA terminals can display color-enhanced ballots, but they also can display three non-color ballot formats: (1) high contrast with same font, (2) high contrast with a large font, and (3) an audio interface in which the contents of the ballot are spoken to the voter via an audio headset. ADA-capable terminals that are set to display color-enhanced ballots are usually not used for ADA voters because changing the terminal between the color-enhanced mode and the ADA mode requires a non-trivial administration action. Therefore, a voter who votes on an ADA-capable machine and does not invoke any of the ADA ballot formats will generally receive a black-and-white, high-contrast version of the ballot that they would have received on a non-ADA machine.

In Sarasota County, each polling place contained at least one ADA-capable machine as well as some number of non-ADA machines. ADA machines were not reserved solely for voters who needed the special accessibility features. Some non-ADA voters voted on ADA-capable terminals, and thus received the black-and-white, high-contrast regular-font ballot layout.

3.1.6 Poll Closing and Accumulation. At the end of the voting period, a poll worker reinserts the Master PEB into each terminal. We note that the iVotronic equipment does not itself impose any requirement to have a special Master PEB. Rather, the convention of using a Master PEB evolved because the iVotronic requires that the same PEB that opened a terminal be used to close that terminal. Designating a Master PEB has several procedural advantages over using multiple PEBs to open and close the polls; among other things, it reduces the potential for poll worker confusion about which PEB to use. Should the Master PEB be lost or unavailable when it is time to close the polls, there are procedures that allow an alternate PEB to close terminals.

It is also important to note that no cumulative vote count is kept during the voting period. Rather, iVotronic terminals accumulate Cast Vote Records (CVRs) in persistent, non-volatile memory. Each CVR records the set of candidates that a single voter voted for. CVRs are sometimes known colloquially as "ballot images", though it is important to point out that they are not stored as a graphical image; instead, a CVR simply contains a list of codes identifying the candidates associated with that CVR. The closing process generates the paper summary tape used in the canvass process from the CVRs that are stored in the terminal non-volatile memory. The summary tapes are signed by poll officials and become the official returns from that polling place.

A separate step in closing the polls is the accumulation of audit data, namely event log entries and ballot images (CVRs). The poll worker is given an option to transfer the contents of the three terminal flash memories to the removable Compact Flash (CF) card. Each of the flashes is copied using a low-level binary copy to a special format file in the CF. This option was exercised in the CD13 elections in Sarasota as part of the closing procedures and we understand the resulting files are available by public records request.

3.1.7 Tabulation. As with Election Generation, tabulation occurs on the Unity server, not on the iVotronic terminal.

3.2 iVotronic Hardware Architecture

As we noted earlier, the iVotronic architecture and construction details are proprietary and we agreed to protect the vendor's intellectual property where it is not specifically pertinent to this analysis. Thus, we give an overview without addressing details where they are not important to our findings. Figure 1 provides a visual overview.

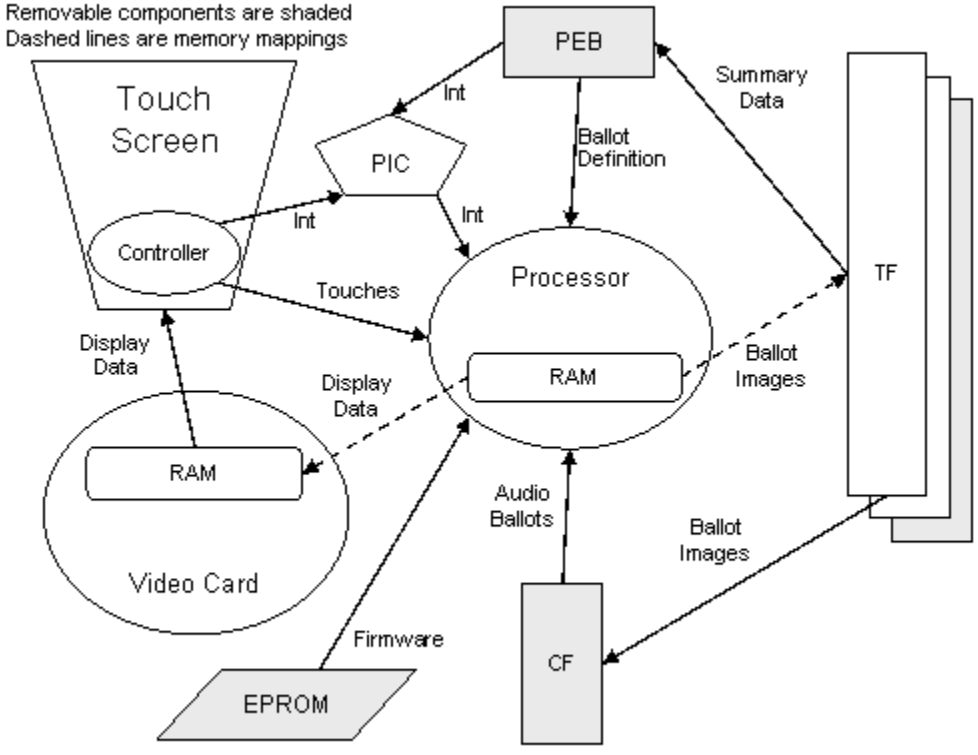
3.2.1 The iVotronic Terminal. The iVotronic terminal is the device that voters engage to review the contests and cast their ballot. As computers go, it is a simple device with the primary component and component packages that we discuss below.

3.2.2 Main Processor. The iVotronic processor is a widely used, general purpose processor. It is sufficiently mature that its properties are well understood and it has no distinguishing properties that impact this analysis. During each voter session, Random Access Memory (RAM) stores components of the contest and candidate records.

3.2.3 Touch Screen. The primary input/output interface is a touch screen, which is a graphic display panel with a pressure-sensitive surface. When pressure is placed on the touch screen, electrical resistance is reduced at the point of pressure. The screen is a commercial off-the shelf component.

3.2.4 Touch Screen Controller. The touch screen controller is a programmable microcontroller that determines the X and Y coordinates of the point of maximum pressure on the touch screen. The touch screen controller also performs other functions, such as providing information about the battery voltage level of the system and turning on and off the backlight. It communicates with the main processor via the synchronous serial I/O port. It interrupts the main processor when it has data

Figure 1. iVotronic Hardware Architecture



TF – Terminal Flash Memory, CF – Compact Flash Memory, PEB – Personal Electronic Ballot

on this port for the main processor to receive. The touch screen controller is a commercial off-the-shelf component.

3.2.5 Programmable Interrupt Controllers (PICs). The system has two programmable interrupt controllers, called PICs for short. The PICs intermediate interrupt requests for the main processor from other devices. The devices that request interrupts in the iVotronic are: (1) two asynchronous serial I/O ports, (2) a synchronous serial I/O port, (3) a timer circuit that generates an interrupt every millisecond, and (4) hardware exceptions. The PIC hardware is a commercial off-the-shelf component.

3.2.6 Real-Time Clock. The real-time clock device keeps an integer count of seconds. It is read by the main processor and used to compute the date and time of day to a resolution of one second. It also provides information such as the serial number and model of the iVotronic device, an indication of whether there is a PEB in the PEB slot, and whether the PEB is of the supervisor or voter type (PEB type is discussed in a later section). The real-time clock device provides this information to the main processor via a sequence of 12 characters that is repeated once per second, one bit at a time. The real-time clock device cannot interrupt the main processor. Software on the main processor must poll the real-time clock bit frequently enough not to miss any bits.

3.2.7 Serial Communications Ports. There are two asynchronous serial I/O ports and one synchronous serial I/O port. One of the two asynchronous serial I/O ports is dedicated to serving the RS 232 interface to the external communications (printer and modem) pack. The other is dedicated to infrared communications with the PEB. The synchronous serial I/O port is dedicated to communications with the touch screen controller. Each of these interfaces interrupts the main processor when input data is available.

3.2.8 External Communications Pack (Printer/Modem). A modem can be connected to the iVotronic by attaching a communications pack through an RS-232 interface to one of the two asynchronous serial communications ports of the iVotronic. The modem can be used for transmission of election results to a central location. The communications pack also provides a printer that can be used for printing summary tapes.

3.2.9 PEB and PEB Interface. A Personal Electronic Ballot (PEB) is a non-volatile memory device designed for use with the iVotronic. PEBs hold the ballot definition, are used to open the terminal and to initialize every voting session. A PEB is about the size of a pack of cards.

The PEB communicates with the iVotronic terminal through a short-range infrared interface. The iVotronic terminal contains a special slot that a PEB can be inserted into. The iVotronic contains a magnetic switch that senses the presence of a PEB, and the iVotronic is programmed to only communicate over the infrared interface when a PEB has been inserted. The infrared connection is completely physically shielded while the PEB is inserted into an iVotronic terminal. We reviewed software that drives the PEB's infrared device and the corresponding terminal software that interfaces with the PEB.

A Supervisor PEB is a PEB that is initialized to be utilized exclusively by a poll worker. Sarasota County used a voting process known as "Pollworker-Activated Mode." In this operation mode, a poll worker who possesses a "Supervisor PEB" enables a machine for each voter. Specifically, the poll worker:

1. engages a validated voter at the check-in table;
2. escorts them to an available voting machine;

3. inserts the PEB to enable the machine;
4. removes the PEB; and
5. leaves the voter to select and cast their votes.

The voting session cannot begin until the supervisor removes the PEB from the slot. The voter completes the session by pressing the vote button to cast their ballot. Should a voter leave an open session without casting their ballot, the poll worker can reinsert the PEB to cancel or cast their ballot and reinitialize the iVotronic for a subsequent voter.

The Supervisor PEB as issued to the poll worker is fully functional. Without any recharging or other re-initialization, poll workers can:

(1) open the polls;
(2) initiate new voting sessions;
(3) cancel ongoing problematic voting sessions;
(4) enter the service menu; and
(5) close the polls

There is an alternate election administration procedure that uses another type of PEB, the "Voter PEB". In that process, the clerk gives each voter a Voter PEB that enables their own terminal. This process is known as "Voter-Activated Mode." Since Sarasota County did not use Voter-Activated Mode and did not employ Voter PEBs, we do not detail their operation further.

A Master PEB is a single Supervisor PEB that the polling place elections staff selects to open and close the election on all machines in the polling place. Before the election begins, all Supervisor PEBs within a precinct contain the same data. In particular, the Master PEB is identical to all other PEBs in that polling place, except for the serial number and other PEB-specific identification information. Local officials generally mark the Master PEB with a colored band for proper identification. Normal election procedures require that the same PEB (i.e. the designated Master PEB) open and close each machine. There are fall-back procedures to use an alternative PEB to close the election, for example if the Master PEB is damaged.

3.2.10 Vote Button. The vote button is a physical switch that the voter uses to cast their ballot once they complete candidate selection. The vote button only becomes active once the voter has paged through the entire ballot to the last review page. A flashing light inside the vote button indicates its active status. On a non-ADA machine, it is the only physical button (as opposed to "buttons" displayed on the touch screen) that voters engage.

3.2.11 Paging/Response Buttons to Support ADA Voters (ADA machines only). The Help America Vote Act (HAVA) requires at least one ADA terminal in any polling place at which disabled persons will vote. Approximately every fifth iVotronic terminal is ADA equipped. These terminals differ slightly from non-ADA machines, most prominently in that ADA terminals have three physical buttons for interacting with the machine. Like the VOTE button, ADA buttons are mounted on the iVotronic terminal frame, not displayed on the touch screen itself. When the poll worker selects ADA audio voting, the touch screen is inactive and the voter presses the ADA buttons in response to the audio ballot. When non-audio voters use the machine, the ADA buttons are disabled.

3.3 Memory Architecture. The iVotronic memory system is engineered in a five-tier hierarchy.

3.3.1 PEB. As described above, the PEB contains non-volatile memory. At poll opening, the PEB contains the ballot definition, which is copied into the terminal flash (see below) when a voting

session begins. The PEB is designed to be easily and regularly inserted and removed from the voting terminal, as many as several hundred times per day, to initiate voter sessions and perform other functions. At the end of the election, one Master PEB in each polling place closes each terminal, receives and stores voting summary information, and may be inserted into a designated terminal to send results to the Supervisor's office via modem connection.

3.3.2 Non-volatile terminal flash provides persistent storage for ballot images. The iVotronic contains three internal flash chips that are used to store data in triplicate. If a voting terminal loses power, any ballot images recorded in this triply redundant store remain intact and available once power is restored, or through recovery procedures. The three flash chips are intended to contain identical information, and the iVotronic firmware regularly compares their contents to ensure 100% consistency. Two of these flash chips are fixed in the terminal and cannot be easily removed. The third can be easily removed by County elections technicians, for instance for auditing purposes, after opening the terminal case. Ballot images remain on these three flash chips and are available for audit until elections personnel conduct a clear and test operation (which erases terminal flash).

3.3.3 Compact Flash card. The Compact Flash (CF) card provides non-volatile storage. Like the PEB, the CF card is designed for easy removal. However, unlike the PEB, it is not intended to be removed from the terminal during the voting session. A sliding door on the terminal protects the CF card. In Sarasota County, tamper-resistant tape is used to seal this door to reduce the risk of removal without notice.

The CF card itself is similar to devices that consumers utilize in cameras and other portable devices that require high volume, non-volatile memory, e.g. SD cards. Before the election, elections staff insert the CF cards into the iVotronic terminals at County headquarters. The CF cards initially contain only audio files (for use with the audio interface provided by ADA machines) and information identifying this election uniquely. The contents of the CF card are not modified during the election. The poll closing procedure used in Sarasota County copies the ballot images and other audit information accumulated on the terminal flash to the CF card.

3.3.4 At the lowest level, the on-board Random Access Memory (RAM) provides volatile memory. RAM is not designed to be removed from the terminal and is not useful for routine audit purposes because its contents are volatile and vanish when the machine is powered down.

3.3.5 Erasable Programmable Read Only Memory (EPROM). The EPROM is a fixed chip that stores the iVotronic firmware (i.e., the executable code executed on the iVotronic's main processor). Firmware only needs to change when a software version update occurs. Elections staff typically load firmware to the EPROM through the service menu that copies the new firmware to the EPROM from a compact flash card that was prepared for this purpose.

3.4 iVotronic Software Architecture

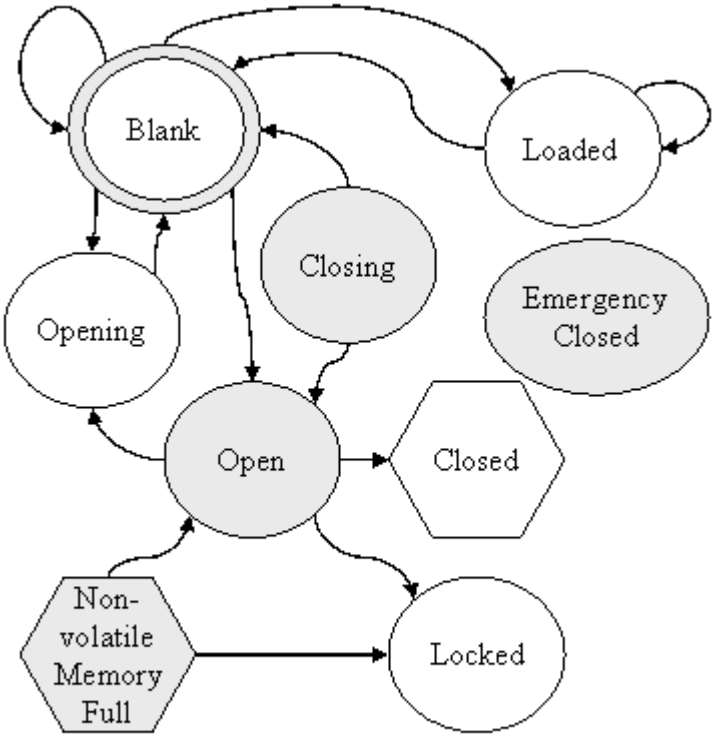
To protect intellectual property, we again avoid providing details where these details are not relevant to our findings. iVotronic firmware is organized into two module groups, one that handles hardware interaction and services; the other is best described as voting application code. The software allows nine processing modes. Figure 2 roughly summarizes these modes.

3.4.1 Low-Level and Machine Interface Code. This group of code provides a hardware abstraction layer designed specifically for the iVotronic. The iVotronic does not have an operating system as that term is commonly understood. The modules in this group perform necessary services that

operating systems typically provide, including management services and interfacing to the input/output devices. Most of this code is written in C, a high level language that is commonly used for operating system code; there are a few small assembly language modules.

3.4.2 Application Code. These modules include the code that runs the election, including vote selection, vote recording, and the graphical user interface. These modules also include code to generate the summary reports and transfer them to the PEB, as well as the code to transfer the ballot images and audit data from terminal flash to the CF card. The application code is written entirely in C.

Figure 2. iVotronic Software Modes



3.4.3 Relevant Code Properties

When we received the software, we did not expect to see high assurance source code. While the code meets the target 1990 voting system standard, there is a wide variation in naming and other readability characteristics. Global variables are integral components of virtually every function. While developers did not use "gotos", control flow is not standardized and is often unintuitive. The code base is aging and shows the effects of numerous modifications. The team was frustrated by the code's limited readability, and we suspected corresponding reliability issues.

Other aspects of the code structure present hurdles for readability and maintenance, so errors could easily be introduced during updates to the code made as part of the normal software life cycle. There is an excessive reliance on global variables compounded by a lack of a high–level design to model the software components and functions. This led to a repetitive coding style, in which functions sometimes repeat checks and initializations that were performed at earlier points. We identified several benign, harmless defects caused by this strategy.

A positive aspect of the iVotronic firmware is that it contains only a small amount of commercial off-the-shelf (COTS) code not written by the vendor, including a driver for the CF card and a standard C library provided by the compiler. We did not review the source code for any of this COTS code, but because COTS code was used so sparingly, this was not an impediment to the iVotronic firmware analysis.

Conversely, the iVotronic firmware source has several important properties that support reliability and maintenance. Of central importance, the vendor controls all critical code, as there is no commercial operating system. Thus, the iVotronic code need not provide general-purpose functionality; rather it focuses on special purpose electronic voting services that are narrowly tailored to this specific application.

Moreover, while the code is not highly readable, it avoids complex (and correspondingly dangerous) operations such as dynamic memory allocation and multi-threading. Though the

iVotronic code is not well modularized, it also does not suffer from well-known complexities associated with modern object-oriented programming, such as the fragile base class problem.

The basic structure of the application is simple. The voting code executes in a single-threaded, single-address space application, thereby avoiding many of the challenges associated with multi-threaded concurrent software. There is a single thread of control, corresponding to the main program. The processor is reset and the main program is reloaded and restarted with freshly initialized variables for each voter. There are hardware interrupt handlers that interact with I/O devices primarily to update global variables. Those global variables are read, and sometimes also updated, by the main program, thus there is a potential for timing-dependent errors.

4 Assumptions

During the course of any scientific analysis, investigators make many assumptions. Here, we list the most important subset of assumptions that we made. We used most of these assumptions to reduce the amount of code we had to review manually by limiting our examination of code to that which could have executed in the CD13 race. As our work progressed, we were able to independently corroborate these assumptions as noted below.

4.1 Election Configuration

While the iVotronic is used only for elections, voting system requirements can vary greatly from state to state, or even county to county. For example, some states leverage touch screen device capabilities to reduce natural candidate order bias by rotating the candidate order from voter to voter. Thus, even though the iVotronic code is special-purpose software targeted to a specific task (i.e. voting), there is always a significant amount of the code base that is not exercised in any given election. In many cases, configuration options determine which code paths can or cannot be executed. We examined the election configuration used in Sarasota County and used it to focus our efforts on relevant code and to allow us to understand the correct execution paths. In particular, we only examined code that could have been executed in Sarasota County in November, 2006, given the configuration options that were enabled in Sarasota County. Consequently, many of our assumptions refer to which configuration options were enabled.

We confirmed these assumptions in a variety of ways. For example: (1) we looked at screenshots of the Sarasota ballot; (2) we examined textual versions of the ballot definition files from the Sarasota election; (3) we loaded the Sarasota election definition onto an iVotronic and executed and observed a mock election using the same election definition files used in the November 2006 election; and (4) we obtained information about the November, 2006 election from the FLDoS and the Sarasota Supervisor of Elections staff.

4.1.1 No Candidate Rotation. As noted above, iVotronic firmware supports candidate rotation so that the candidate's ballot order is rotated from voter to voter. In Florida, the candidate order is static, so the Sarasota ballot that includes the CD13 race did not rotate candidates.

4.1.2 No Multi-page Races. Occasionally there are so many candidates in a race that it is not possible to effectively display them all on the same ballot page. The iVotronic firmware includes logic and features to handle multi-page contests. There were no normal-font, multi-page races on the Sarasota ballot. However, some races displayed in large-font mode required more than one page to display. The CD13 race displayed on a single page by itself in large font mode.

4.1.3 Multi-Column Display. The iVotronic firmware allows single and multi-column ballot pages. There were nine ballot styles used in Sarasota across 156 precincts. Of course, ballots differ

by precinct. The Sarasota ballot styles utilize between fifteen and twenty-one single column pages for initial candidate selection and three or four double column display pages for the ballot summary. Three or four review pages are two-column display. All re-vote pages are displayed in a single column.

4.1.4 Re-vote Pages. During the review process, when a voter selects a race to re-vote, the iVotronic software generates a display page containing only the selected race. This is a logical and appropriate process: the voter is presented with only the race that was selected for re-vote. However, we note that this behavior may create the illusion of a missing race on the original ballot: because the re-vote screen is so different from the main voting screen it may confuse voters into believing the revoted race did not appear on the original ballot. We address this as an undervote hypothesis in Section 6 below.

4.1.5 Text Ballots. The iVotronic manages ballots in either text or bit-mapped format. There is a significant amount of iVotronic code that deals with bit-mapped ballots. The Sarasota ballot was text-based, and there were no bit-mapped ballots used in Sarasota County.

4.1.6 No Multi-Language Ballots. iVotronic text-based ballots allow English or Spanish versions. Only English language ballots were activated in Sarasota.

4.1.7 No Straight Party Voting. Some states provide a simplified voting process for straight party voters. While the iVotronic firmware supports this voting feature, straight-party voting was not enabled in the Sarasota County ballot definition.

4.1.8 No Controlling Contests. When a voter's selection in one contest determines her eligibility to vote in a different contest, the former is called a "controlling contest". There were no controlling contests on the Sarasota ballot.

4.1.9 The Firmware Compilation Environment. We assume that the tools used to build the firmware from the source code:

1. Worked correctly;
2. Comply with the ANSI C programming language standard;
3. Do not have any bugs or unexpected behavior.

We assume that the firmware image provided to us was compiled correctly from the source code provided to us. We also assume that the firmware image provided to us was the firmware image that was actually executed by the iVotronic machines on Election Day. These assumptions imply that the executable software executed by the iVotronic systems during the election matched the source code we examined. As our study focused *only* on the source code, we did not attempt to reconstruct the executable firmware image. Both ES&S and FLDoS told us that the firmware compilation environment worked correctly.

4.2 Ballot Images Contain the Undervote. The undervote totals shown in the summary reports are identical to the ballot images that reside on non-volatile terminal flash memory. FLDoS confirmed that in their tests they extracted ballot images through the election management system, compared the count to the summary tape, and confirmed that the totals were identical. This indicates that if the undervote is due to a flaw or malicious act, that flaw or malicious act changed both the ballot image and the summary report. It also ensures that the undervote did not occur due to a tabulation error at poll closing or afterward by any means, either accidental or deliberate.

4.3 Hardware Configuration Assumptions. We assume that the external communications pack was not attached during the election. Also, we assume that the touch screen controller and PICs, did not fail in a malicious way; that is, they either functioned correctly or failed in a way that was detected and resulted in the machine being taken out of service.

5 Activities That Are Out of Scope for This Analysis (i.e. Things We Did Not Do)

5.1 We did not conduct a comprehensive election audit. The Statement of Work gave the task of this team as:

> *The sole purpose of this project is to conduct a scientifically rigorous static software analysis on the iVotronics version 8.0.1.2 firmware source code to determine and identify flaws, vulnerabilities or anomalies, if any, that may have potentially caused, contributed or otherwise created the higher than expected under-vote rate in the District 13 Race.*

The team's task was *not* to examine the iVotronic systems or the PEBs used in the election, or to perform forensic analysis on those systems to determine whether a problem in them caused the undervote. The team's task was to determine whether the source code used to create the firmware on those systems had flaws that would explain, or could have contributed to, the undervote. An analogy to the limited task of this group lies in the realm of automotive mechanics. If one car's computer has a problem, that car is examined. If many cars' computer systems have the same problem, a larger study is required to determine whether the programming is at fault. The individual cars are also examined to determine whether the individual computers were defective, or the programming on those individual computers was altered. In this analogy, the team is examining the programming. This is a part of the broader study into the computers failing, the FLDoS conducting the complete study of the cars' computer systems. Our investigation was just one part of a larger audit performed by the FLDoS.

Nevertheless, many hypotheses concerning the undervote can be ruled out through a combination of source code review and other evidence, such as the distribution of the undervote across the entire county, a similar undervote in Charlotte County (see Section 8.1 below), and the absence of undervote in other iVotronic jurisdictions.

5.2 We did not attempt to verify that the code is completely free of defects. There are fundamental limits on the ability of manual source code review to find defects in computer software. Manual code review is an imprecise process, guided by best practices and analyst intuition. It is impossible to check all code paths that might be executed in any nontrivial computer program. Also, in any nontrivial computer program, it is impossible to exhaustively enumerate and analyze the full state space that the code inhabits. Moreover, humans are fallible: just as the original software programmer can miss a defect in the code they write, so too can independent reviewers overlook subtle defects and bugs in the code.

We did not attempt to use formal methods. We did not attempt to apply software verification techniques such as Hoare logic, Dijkstra's weakest preconditions, or model checking to mathematically prove the absence of defects in the code. Rather, we used informal code inspection, guided by our engineering knowledge and experience.

Classically, software analysis usually involves a combination of static analysis (e.g., manual code review) and dynamic analysis (e.g., black-box testing, unit tests). This project was charged to perform static analysis of the code; dynamic analysis was not part of our charge. That said, the team were provided some access to equipment for testing and hands on evaluation, and we did

supplement our static code analysis with directed testing and experimentation with the iVotronic equipment. However, even the combination of static code analysis, black-box testing, and clear-box testing cannot reveal the presence or absence of all faults in non-trivial programs.

For these reasons, we make no claims that we found all bugs or defects in the code.

However, we did perform a systematic and structured analysis of the aspects of the code that we believed to be relevant to the CD13 undervote. The purpose of this study was not to find all potential defects, but rather to accomplish the limited objective of finding the specific class of defects that may have contributed to the CD13 undervote. The limited scope of our investigation helped us to focus our analysis and increases our confidence in the completeness of our findings. While another set of reviewers with access to the code might find bugs we missed, we do not believe they would find bugs or defects that caused or contributed to the CD13 undervote. Nonetheless, we accept that certainty is unlikely even with limited scope and correspondingly offer only our best professional opinions rather than absolute guarantees.

We also emphasize that, even though manual code analysis has limitations, it is nonetheless an effective and powerful way to analyze a system such as this. Code inspection is a state-of-the-art technique for evaluating the reliability, security, and accuracy of systems such as this, and it has important advantages over its competitors. For instance, code review can find many defects and problems that black-box testing (e.g., logic and accuracy tests, mock elections, and parallel testing) cannot. Code review is especially powerful when combined with other software testing and evaluation methods, such as those undertaken during the FLDoS audit. If there were a software flaw or bug that caused or contributed to the CD13 undervote, we believe that one of these methods would have been able to find it.

5.3 We did not conduct a Red-Team exercise. One popular computer system vulnerability assessment approach is to engage skilled security specialists to attack working systems in order to determine their security strength. Depending on the terms of the Red Team project, they may have extensive access to code for static and dynamic analysis, or they may simply observe the system to determine their attack simulation approach. When done right, Red Teams rely on skill sets acquired through years of red teaming and an understanding of how the systems are used in the field. Red Team assessments are often conducted under conditions that mirror how the system will be used in practice. In this case, a thorough Red Team assessment would have had to be performed under conditions that mirror an actual election, with consideration of all administrative and security mechanisms that are employed in practice. We did not conduct a Red Team assessment.

5.4 We did not examine election management system source code. As we note earlier, the Statement of Work confined our work to analysis of the source code for the iVotronic firmware. We did not exercise or examine the election management system software. We note, however, that no activities of the election management software after the election could have had any effect on the undervote, because the summary tapes produced at polling places at the close of voting also showed an identically high undervote rate, and there is no way that any failure or fault in the election management software after the election could have altered the summary tapes.

5.5 We did not duplicate FLDoS audits**.** The FLDoS audit plan and results to date are posted on the FLDoS web page. These tests include machine and equipment examination, parallel testing, and other analysis. Although some of the team's activities overlapped with these tests, we did not duplicate these efforts. For example, the FLDoS conducted two dynamic tests, termed parallel tests

[10]. These tests involved precisely reproducing election day behavior by having staff members acting as voters entering selections directed by scripts generated based on voting terminal audit records. They conducted two parallel tests, one with standby terminals that were not used in the CD13 election and the second test did utilize terminals used in the election. These tests did not reveal any anomalies. Thus, we did not conduct parallel tests and we did not disassemble any iVotronic or PEB hardware. Rather, we examined anomalous behavior only to identify possible hypotheses that might explain the CD13 undervote.

5.6 Software that we did not review. There are two categories of software components within the iVotronic terminal whose review is outside this project's scope and thus, was not available to the team. One category is firmware of I/O devices. There is a programmable microcontroller that manages communication between the main processor and the touch-screen. Comments in the main processor code identify the part number of the microcontroller. The interactions with the controller are well defined and are under the control of the main-processor firmware, which we reviewed.

The second category is third-party utility libraries. There is an I/O library provided by the manufacturer of the terminal, compact, and PEB flash memory modules and there are the C-language runtime libraries provided by the compiler vendor. These are reported to be generic libraries, proprietary to the respective third-party vendors, and are not considered part of the iVotronic firmware.

6 Findings

The first group of detailed findings that we present deals with reliability issues. In particular, our focus in this section is to identify potential non-malicious software faults that may have contributed to the CD13 undervote. We order these reliability findings based on the primary point at which they occur in the iVotronic election process. The later subsections detail asynchronous concerns and audit related issues.

Much of our work was focused on attempting to confirm or refute specific hypotheses that, if true, might explain the CD13 undervote. Consequently, many findings reflect our hypothesis-based approach, and we relate most of our findings to potential causes or contributors to the CD13 undervote. Once again, we do not claim to have exhaustively considered all possible undervote hypotheses. Rather, we examined those scenarios available to us and we spent considerable time and energy brainstorming and seeking alternative hypotheses. Our team spent many person-hours reviewing information about the undervote symptoms as well as reviewing the firmware source code. While potential hypotheses may remain unconsidered, we believe that we investigated those that are most likely and most dangerous.

We generally present our findings as follows. We describe the general hypothesis, and then refine it to a particular falsifiable hypothesis. We identify technology features that might make the hypothesis a reasonable one, and then give a technical analysis of the relevant parts of the system in light of the hypothesis. Then we discuss constraints that might inhibit the hypothesis holding and specific factors present in Sarasota that might have enabled the hypothetical flaw. We close with a remedy for the flaw postulated in the hypothesis.

We emphasize that we did not conduct independent investigations to verify information given to us by the FLDoS or Sarasota officials. Many of the enabling factors and constraints come from their information.

6.1 Findings About the Elections Setup

6.1.1 Problem in Name Information on One or More PEBs.

Overview. The iVotronic firmware utilizes a special character (@) as the first character in the candidate name field to indicate that the candidate is "non-votable". Were this symbol to inadvertently appear in a ballot definition, voters would be prevented from casting a vote for that candidate.

Hypothesis. Based on voter complaints indicating that they were not given an option to vote for a candidate, we considered the possibility that the non-votable character may have been erroneously prefixed on a candidate's name on some number of PEBs. Were the PEBs initialized with this flaw or if such a flaw was introduced during the voting process, the candidate would appear on the ballot, but since there was no controlling contest to enable the candidate, no voter with a voting session initialized with such a PEB would be able to vote for the candidate that was so marked.

Enabling Technology. PEBs are special purpose memory devices that hold ballot definition files to initialize voter sessions and store summary tapes when the election is closed. While malicious injection is possible, a more likely cause would be a faulty initialization process that mistakenly pre-pended the @ character to a candidate's name. In this case, approximately fifteen percent of PEBs would have had to contain this faulty ballot definition in order to cause the entire undervote.

Technical Analysis. We did not find any mechanisms in the firmware that can prevent this error (or attack). If the ballot definition marks a candidate as non-votable, the firmware recognizes them as non-votable and does not display a vote box, and the touch screen is not configured to detect a vote for that candidate. This status is pervasive through the entire voting process, including the summary and review process. Conversely, there is no code that modifies the candidate name field, so if this field contains this character initially, it will persist for the duration of the election on that machine.

Constraints. There are several contraindications regarding this hypothesis. The primary detractor is that if a candidate is labeled non-votable, voters would not have been able to correct the undervote through the review process. The vast majority of voter complaints confirmed that they were able to correct the undervote. Additionally, were this problem widespread, it is such a clear flaw that it would have generated many very specific reports that poll workers could easily have verified and would have noted in the precinct reports. We found very few voters that noted this problem. Finally, we requested a copy of the election definition files from the Sarasota County Supervisor of Elections. We were given a textual dump, output by Unity, of all of the ballot styles used in every precinct in Sarasota County. No candidate name was prefixed with an @ character in those files.

Enabling Factors Present in Sarasota. There are no enabling factors present. We were led to examine this hypothesis by the undervote symptoms.

Potential Remedy. Utilizing a special character in a name field is non-intuitive and error-prone. The contest record should include properly named and typed fields that reflect the contest's status on the ballot. It should be controlled by well-defined, clearly identified mechanisms.

6.2 Findings About Voting Sessions

6.2.1 Voting Phase Findings

6.2.1.1 Investigate Reports that the Display Was Slow to Respond to Touch.

Overview. We consider a scenario in which technical impacts from slow touch screen response may unintentionally prevent the voter's selection from registering during the vote selection process, but not during the review cycle.

Hypothesis. If a voter is able to interact with the touch screen in a sequence that causes the screen to display a candidate selection that does not match their most recent touch, it may cause the voter to misinterpret their selection for that race. Specifically, consider a situation where a voter touches a vote box twice in rapid succession. If the software delays updating the display in response to the second touch for some reason, after a very short period the voter may accept the first display response as conclusive, and due to the delay (if it exists) the voter might not notice the delayed update in response to the second touch. It is also possible that the second touch would cause the candidate to be deselected after having been selected.

Similarly, we consider a situation where a voter touches a vote box and waits patiently for her vote to display for a few moments before assuming her touch was not detected and touching the screen again. If the first touch is recorded and if the display is updated only after the second touch, the voter may accept the first display response as conclusive, while a delay (if it exists) could cause later display of the second recorded touch that the voter may not notice.

These scenarios are consistent with reports by some voters that they voted for a candidate, but noticed their vote was not registered when they reviewed their selections on the summary screen.

Enabling Technology. Low level hardware and software systems often utilize semaphores, polling routines, and other "wait and see" control procedures. We consider possible code flaws that may trigger these timing mechanisms in a way that exceeds normal limits, and cause corresponding synchronization problems.

Technical Analysis. Source code inspection reveals a predominantly sequential control process between touch detection, vote recording, and vote display. The only exceptions are a few interrupts that update global variables and return immediately. Our analysis indicates that the software cannot read a new touch until after the previous recorded selection displays. In particular, after detecting a touch to the screen, the software immediately updates the screen, then clears the buffer of touch events and waits for a quiescent state (i.e., where the voter is not touching the screen) before accepting the next touch event. At the hardware interface, the software cycle involves writing the image into a display buffer, and the delay in displaying this image to the voter can be measured in milliseconds. The touch screen controller displays this buffer approximately thirty times per second. Since the software extracts the information to generate the display value from the candidate's vote field, sequencing appears properly implemented. While it is conceivable that the touch detection mechanism may cause significant delay, such a delay could not result in a press, record, display synchronization problem in the scenario we describe.

Enabling Factors Present in Sarasota. The team reviewed numerous precinct log entries and noticed that several voters complained about slow touch screen response.

Constraints. (1) Machines where screen delay complaints originated did not uniformly reflect the high undervote. The first machine that we checked had only a 7% undervote rate, considerably less than the 18% undervote rate. (2) There is no logic that explains why such a fault, if it existed, would have affected only one race on the 15-21 page ballot.

Potential Remedy. Aging and dirty hardware components are out of this team's scope. However, from a systems perspective, when elections depend on machinery, Supervisors of Election must have an aggressive maintenance and replacement schedule in place for that machinery.

6.2.1.2 Consider Whether Event 18 ("Invalid Vote PEB") Caused or Contributed to the CD13 Undervote

Overview. Sarasota event logs reflected a significant number of Event 18 instances. These events are identified in the event log by the message "Invalid Vote PEB". We traced each Event 18

instance to its cause and verified that they had no impact on any voting function, thus no impact on the CD13 undervote.

Hypothesis. We considered the Event 18 occurrences anomalous and investigated whether they may indicate machine failure or relate to any unusual behavior, particularly behavior that may have contributed to the CD13 undervote.

Technical Analysis. As part of the audit record routinely provided by the iVotronic, the firmware logs events that describe activity that may be of interest after the election. Though this particular event is unusual, we tracked each event occurrence to its cause. Each was triggered by the machine losing (or never having) communication with the PEB during an operation that needs the PEB to be inserted. Such instances mis-assign the value 0 to the variable that tracks whether the PEB is a Voter PEB or a Supervisor PEB. The software that prints the event logs interprets a 0 (or any value other than the value for a Supervisor PEB) as meaning a Voter PEB.

We identified a number of entries in the event logs associated with PEB type 0, including "Invalid Vote PEB" (Event 18) as well as "Normal Ballot Cast" events. There are four different categories of unexpected event log entries associated with PEB type 0 in the Sarasota event log. The first category is caused by a benign software defect. The final three categories reflect valid responses to poll worker PEB handling anomalies.

1. A "Normal Ballot Cast" event associated with PEB type 0 on an ADA terminal with Spanish disabled, and the PEB serial number in the event log is 0.
2. An "Invalid Vote PEB" event is the first event of a day, and the PEB serial number in the event log is 0, and there was voting the previous day. This occurs only in early voting situations.
3. An "Invalid Vote PEB" event intermittently occurs, and is immediately followed by a subsequent vote cast, and the PEB serial number associated with the "Invalid Vote PEB" event is 0.
4. An "Invalid Vote PEB" event intermittently occurs without an immediately subsequent vote, and the PEB serial number associated with the "Invalid Vote PEB" event is 0.

The first of these symptoms results deterministically from a defect in the code that only triggers on ADA terminals when the Spanish ballot is not enabled. The defect is associated with a function call that attempts to query the PEB when no PEB is present. When no PEB is present, the iVotronic software routine that queries the PEB assigns a 0 to the global variable holding the PEB type, to indicate a failure when attempting to query the PEB. In this case, it also sets the global variable holding the PEB serial number to 0. After the poll worker removes the PEB, the iVotronic terminal proceeds to display an initial screen to the voter. In the process of composing that screen, a function is called to display the PEB voltage, and that function queries the PEB. Since this function is invoked after the PEB has been removed (i.e., when no PEB is inserted), it will have the side effect of setting the PEB type to 0 and the PEB serial number to 0. Later, when the voter finishes voting and casts their ballot, the iVotronic terminal writes an event log entry indicating a "Normal Ballot Cast" event, and it reads the current value of the PEB type and PEB serial number global variables without checking them for validity and stores them in the event log entry. Since those global variables hold the value 0, this results in a "Normal Ballot Cast" event with PEB kind 0 and PEB serial number 0. Later, when Unity software is used to convert the event log into a textual format, the PEB kind 0 will be interpreted as a Voter PEB, because Unity interprets any unknown PEB kind value as a Voter PEB. Other than this erroneous value (and the PEB's serial number also being set to 0), there is no negative impact and the only symptom is the message in the event log. The error condition occurs during language selection on ADA terminals where the Spanish language ballot is

not enabled. Thus, all ballots cast on ADA terminals in Sarasota reflect this event, and there are no other instances of this event in the event logs.

The team identified this symptom in early January and tracked its cause and impact within one day. We give an extensive, detailed analysis by routine and line number of this category of anomalous event log entries in Appendix F. This appendix is marked as proprietary and confidential, because it gives intimate detail regarding code flow and operation and it identifies the modules, functions, variables, and line numbers associated with this defect in the code. It is illustrative of the rigorous, detailed analysis that the team conducted throughout the review. However, due to its detail, it exposes significant proprietary information and is marked as proprietary and confidential.

Based on this analysis, we were able to determine that the first category of anomalous event log entries were benign. While they were indeed due to a defect in the code, we were able to exhaustively analyze that defect and determine that it did not cause any effect on voting other than causing incorrect log entries.

As it turns out, the other three categories of "Invalid Vote PEB" log entries had a different explanation. This complicated and extended our investigation, because the other three categories clouded the symptom patterns.

The second category of log entries occurs on machines utilized in early voting. The event uniformly occurred at the beginning of each day on every terminal where there was voting the previous day, thus is only applicable to early voting devices. Several investigators independently reconfirmed that all code sequences associated with the "Invalid Vote PEB" message (Event 18) had no possible impact on the election results. A query to the Sarasota elections staff confirmed our hypothesis that elections staff woke up terminals by dropping a PEB in the slot and quickly removing it. They did this as part of their opening process to confirm that the vote count on machines locked in secure storage was not tampered with overnight. This process caused the iVotronic machine to register a problem when attempting to query the PEB (because it had been removed before the iVotronic was able to fully read it), thus triggering an "Invalid Vote PEB" message. In other words, the software was operating as designed. Thus, this category, too, is benign.

There are a few event log instances where a new early voting day is not accompanied by Event 18. This pattern is consistent with the poll worker having left the awakening PEB in the machine until the open splash screen appeared. At that point, PEB removal does not trigger an "Invalid Vote PEB" message.

The final two categories of log entries presentations also reflect the proper software response to lost communication with the PEB. If the PEB was jiggled or spuriously removed, normally during voting session initialization, an "Invalid Vote PEB" message is generated. Typically when this happens, a poll worker need only remove and reinsert the PEB to begin a new normal voting session. The event logs consistently reflect that normal voting session completion events follow these Event 18 instances within a few minutes.

In the fourth category of log entries, the event intermittently occurs without an immediately subsequent vote. There are only a few of these log entries and they are all followed by another normal session within an hour or two. This symptom is consistent with the poll worker simply taking the voter to another terminal when they experience an "Invalid Vote PEB" (Event 18) message.

We note that one study identified a correlation between machines that contained a PEB with serial number 0 and a higher-than-average undervote rate in the CD13 race. That study hypothesized that the anomalous event log entries might reflect a software defect that could have contributed to the CD13 undervote. Our analysis refutes that hypothesis and fully explains the cause of the anomalous

log entries. It remains to be seen why there was a correlation between these log entries and the undervote rate. However, we note that every ADA terminal contained these anomalous log entries, and the overwhelming majority of anomalous log entries were associated with ADA terminals. Consequently, the study's results could be alternately described as revealing that ADA terminals were subject to a slightly higher-than-average CD13 undervote rate than non-ADA terminals. It is not clear why this might be so, though one possible explanation might lie with the slight differences in ballot presentation between ADA and non-ADA machines (e.g., non-ADA machines display parts of the ballot in color, while ADA machines display the ballot entirely in black and white when used by non-ADA voters). In summary, the evidence available to us is consistent with our conclusion that machine faults or errors did not cause or contribute to the CD13 undervote.

Potential Remedy. The vendor notified the team about the software defect in late January. They had independently identified this problem and they indicated to us that the flaw is corrected in subsequent iVotronic firmware versions. Again, the log entry does not affect the accuracy of the recording of votes, but the message does not accurately reflect the terminal's behavior. To resolve the early voting wake-up event, we note that the lock and unlock operations are designed for this function and may be a better option for poll workers.

6.2.1.3 Controlling Contests and Their Potential Effect on CD13

Overview. The iVotronic supports controlling contests, a special kind of contest where the selections made in one contest can affect eligibility to vote in another contest. If contest A controls contest B, the voter is only allowed to vote in contest B if she made a specific selection in contest A. As an example, a controlling contest situation may occur in a recall election. The mechanism is used to designate a particular contest—e.g., the choice to recall or not an incumbent—as controller. A subsequent contest—e.g., candidates for the office under recall vote—can be designated as controlled by the earlier contest. If the voter does not select the choice to recall the incumbent, then the voter is not allowed to make any selection in the controlled contest, and an undervote is recorded in the controlled contest.

Though there were no controlling contest relationships in Sarasota, if such a configuration were accidentally present, it could cause an undervote.

Hypothesis. We considered the possibility that the US Senate race was designed as a controlling contest and CD13 as the controlled contest. The iVotronic requires that the controlling contest appear earlier on the ballot than the controlled contest, so the US Senate race is the only possibility for the controller.

Enabling technology. Designating the Senate contest as controller for the CD13 contest could have prevented voters that made a particular selection in the Senate race from casting a vote in the CD13 race.

Technical Analysis. We verified through examination of the code that the iVotronic software enforces that the controller race must precede the controlled race. This restricts the possibility of a controller contest to the Senate race. As part of our investigation we verified that the ballot definition files for Sarasota County did not contain any controlling contests and the configuration of the Senate race was identical in all precincts.

In addition, if a particular voting machine were erroneously initialized with one or more ballot styles defining the Senate race as controller for the CD13 contest, this configuration would be highly visible to the voter: If the voter were to make a selection in a particular contest that is disallowed because of a prior selection in a controller contest, a message is displayed in the screen instructing the voter that unless the controller contest is re-visited and its selection changed, the voter cannot cast a choice in the current contest. We found no voter reports of having encountered

such behavior, which is unlikely since the undervote was extensive and widely distributed. Moreover, the parallel tests performed by the FLDoS did not reveal such behavior.

Finally, if this hypothesis were accurate, it would create a distinctive pattern in the ballot images, where some particular selection(s) in the US Senate race was always associated with an undervote in CD13 on the affected machine(s). To explain the CD13 undervote, most machines would have had to be affected in this way. No such pattern was observed in our examination of the ballot images. Our conclusion is that such configuration error could not have contributed to the observed effects in the CD13 contest.

Enabling factors present in Sarasota. The only factor of note that was present in Sarasota is that the CD13 contest was not the first contest in the ballot, which satisfies one requirement for a controlled contest. While in principle such configuration errors in selected ballot styles could lead to the observed undervote percentages, our review of the official ballot definitions, and more significantly, the lack of recorded voter complaints describing symptoms that match this error effectively rule it out as a factor in the CD13 undervote.

Mitigating Factors. Such a configuration error in the ballot definition can be easily ascertained with proper testing before the election. Again, it is imperative that testing be performed for each ballot definition.

6.2.1.4 Consider the Possibility of Definition of Straight Party Rules

Overview. The iVotronic supports a generic voting feature that can subject contests to straight party voting rules. If a particular contest sets a party preference, the iVotronic may prevent the voter from making selections for candidates of the selected party in all contests subject to straight party, by displaying the candidates of the selected party without a voting box. Instead, when the voter casts the ballot, the candidates of the selected party are recorded as "straight voted." There were no straight party rules in the official Sarasota ballot definitions. In addition, straight vote is neither a blank vote nor an undervote.

Hypothesis. Designating a selection in a straight party contest could subsequently prevent that voter from selecting a candidate of the same party in any contest subject to the straight party rule. If later the voter were to re-visit and de-select the straight party contest, this might result in undervotes in the contests subject to the straight party rules.

Technical Analysis. The presence of straight party rules is visible to the voter in various ways: (1) When interacting with a contest controlled by the straight party rules, the voting boxes of candidates of the same party would not be displayed if the voter is not allowed to make that selection; (2) if the voter were later to de-select the straight party contest, the summary pages would show the undervote in the contests without a selection. Note that the selection of a party subsequent to the choice of a candidate of the same party in another race may not clear the earlier choice for that candidate. In other words, the selection of a party in a straight party contest may not "clear" any voter choices in specific elections, and sets or removes the "default" choice in contests where the voter has not made any selection. Indeed, the erroneous configuration of straight party rules would more likely decrease, rather than increase, undervote rates. In addition, the fact that high undervote rates were observed among voters that displayed a tendency to "straight voting" is a counter-indication to this having been a factor in the undervote—if any of the other races had been configured to trigger straight party preference, that would result in the same party candidate being selected in the congressional race.

Enabling factors present in Sarasota. No enabling factors are present in Sarasota, except for the intrinsic capability of the iVotronic to be so configured.

Mitigating Factors. Such a configuration error in the ballot definition can be easily ascertained with proper testing before the election.

6.2.1.5 Investigate Reports That the CD13 Race Was Not Displayed

Overview. In reviewing polling place logs, we noticed several voter reports that the CD13 race did not appear on the ballot. However, when they noticed the "NO SELECTION MADE" message on the review screen, they returned to re-vote the race successfully. We consider reasons why the race might not have appeared on its proper ballot page.

Hypothesis. We analyzed the firmware to identify potential flaws that cause a race configured similarly to the CD13 race (two candidates, both major party candidates, top of the page, no write-in, appearing on the same page with a many-candidate race) to not be presented to every voter.

Technical Analysis. The team spent many hours with hands on testing and reviewing iVotronic operation, including analyzing the Sarasota election setup. There are three specific properties that are relevant to this question. The first two reflect relationships between the first two ballot pages and the third addresses the differences between the original CD13 vote page and its review page.

(1) While some localities may hold elections on average once per year, few voters vote more than once every two years. Many others vote every four years or less often. Thus, when voters begin the (unfamiliar) voting process, they may quickly grasp any detected patterns as they seek familiarity. On the Sarasota ballot, the first ballot page set a pattern of having two large (3 or 4 lines) headers, separated by a straight line, and followed by a large multi-candidate race. The second page format closely follows this pattern, with the exception that there is a two candidate race between the two headings. The similarity is clear when looking at the first two ballot pages side-by-side as seen in the first two screen shots in Appendix A. This effect is more pronounced when the second page is superimposed on the first.

(2) The vote touch spot for the Democrat and Republican candidates in the large races are similarily placed on the first two ballot pages. Again, the similarity is striking when the pages are superimposed. The vote box for Bill Nelson on the first page is set just above the vote box for Jim Davis on the second page; similarly for Harris and Crist. The problem is that if the voter is drawn to Crist or Davis on the second page because the ballots look the same and because that is where they voted on the first page, they may naturally have missed the CD13 contest. This page similarity is equally as noticeable on the iVotronic itself, where it was first identified.

(3) The third factor reflects the difference between the original CD13 screen and the re-vote screen that is presented when the user visits the review screen and then selects the CD13 race to change their vote in CD13. The original page containing the CD13 contest is full from top to bottom with seven header lines and candidates for two races. Conversely, the CD13 re-vote page displays only the CD13 race and shows only two candidates. The re-vote page for the CD13 race (third picture in Appendix A) is nearly blank, which is unlike any page on the original ballot. It is not surprising that people would insist that the CD13 page *that they saw when correcting an initial undervote in CD13* was not on the original ballot, because it actually was not. It is also easy to understand how voters who used the review screen to correct an initial undervote in CD13 would be convinced that they would not have missed the race had it been originally displayed.

Constraints. Some have suggested [7] that because another ballot page that contained the Hospital Board contest is similar to the CD13 page and the undervote pattern did not appear there, the ballot format could not have caused the CD13 undervote. We assume the researchers are referring to ballots styles other than the ballot style that was utilized in twenty Sarasota precincts that had the Hospital Board contest at the bottom of the ballot page. However, ballot pages #2 and #6 are different in several important ways with respect to the other ballot styles as well.

(1) Page #2 has two (2) three-line headers while page #6 has three (3) one-line headers.

(2) Page #2 has two contests, while page #6 has three equal-sized contests.

(3) Page #2 has one contest with six candidates; page #6 contains only two-candidate contests.

(4) There are nine possible selections on page #2, while there are only six selections on page 6.

(5) The Hospital Board contest appears significantly later in the ballot, after voters have had a chance to acclimate themselves to the ballot format. In contrast, the CD13 contest appears on page #2 immediately after a page containing a single contest, which may have primed voters to expect a single contest per page. That expectation may have become weakened by the time voters reached the Hospital Board contest on page #6.

Another study [9] shows that precincts with older voters, who may be more susceptible to such ballot distractions, experienced a higher undervote rate. Some suggest that older voters may be more susceptible to such ballot complexities than younger voters.

6.2.1.6 Reports that The Voter Choice and the Displayed Values Do Not Agree.

Overview: We consider the possibility that the update of data structures recording voter selection may not be reflected through updates in the information displayed on the screen.

Hypothesis: The recording of voter selection is a multi-step process, starting with the detection of a touch that matches a particular contest and candidate/choice. The choice is recorded to RAM and a subsequent call is made to the display functions. A situation with improper serialization of operations or improper synchronization might lead to erroneous information being displayed to the voter.

Enabling technology. The iVotronic machine allows interrupt-driven processing, as we describe in detail in Section 6.3 below. This allows the execution of machine instructions from an interrupt handler between two statements that appear as consecutive in the application code. If these instructions could update or change variables that change control flow between the time when the structures are updated and the display is called, this could lead to a lack of faithfulness in the information represented to the voter.

Technical analysis: The voting sequence was reviewed from the voting session start to ballot casting. All the updates to the data structures recording voter intent were traced through the code.

The user interaction sequence could be summarized as follows:

(1) A touch is detected and matched to a screen position corresponding to a valid choice.

(2) If this choice corresponds to a contest selection choice (other possible choices are, for instance, to change the ballot page), then a set of checks is performed to decide if the choice is valid. While a de-selection choice is always valid (unselect a candidate or YES/NO choice for a proposition), the same is not the case for a positive selection. For instance, if the contest allows for multiple candidates to be selected (not the case with CD13) and the voter has already made enough selections, the attempted selection is ignored. Other selections that are disallowed have been discussed in findings 6.2.1.3 and 6.2.1.4

(3) If all the checks are satisfied, the selection state for the candidate/choice is changed. A function to refresh the current page is then invoked.

(4) The refresh function scans all contests in the current page for one with a candidate or choice whose selection state has changed by comparing the current selection state to the previous selection state. This contest is then re-displayed, by writing its current state representation to the screen memory buffers. The refresh function also updates the previous selection state for the candidate/choice to the current state.

We note that the application code must make explicit calls for the touch screen controller to update the variables indicating where the last touch occurred. Moreover, once in step (1) the application code detects the location of the touch and resolves which contest candidate/selection it matches, it no longer polls for touch events, ignoring all user input until after the refresh call is performed (or the change is rejected, if the selection is disallowed through a failed check as described above). Therefore, it is not possible for the user to interrupt the normal execution sequence by (for example) selecting to go to a new ballot page before the code has completely processed its prior input. This includes touch screen input and input from buttons, such as the vote button.

The only parts of the code that modify a candidate's current selection state are those that perform the checks in steps (2)-(3). The only part of the code that modifies the previous selection state of a candidate is code that displays a modified contest in step (4). Therefore, if a touch is detected and matched and a change of selection correspondingly triggered for the contest, the code to refresh the contest representation in the screen will be called.

The refresh display code writes directly and synchronously to display buffers. The screen driver displays the changes the next time the screen is refreshed (asynchronously), which happens at a relatively high rate. Meanwhile, the code will have returned to wait for a new user selection through the touch screen or the vote cast button.

We note that the touch screen controller is relatively slow, in particular much slower than the display refresh rate. This makes it highly unlikely that a voter could make a (de-)selection in a race, and quickly browse to another page before the display is updated to reflect the changes. In fact, quickly pressing the screen at different screen positions will not register a touch since the touch screen controller filters out rapid or random touches. While this may be frustrating to the user experience, a discarded user input has no impact on a mismatch between display and recorded voter intention.

Finally, even if the user were not to notice a screen update before moving to another page, due to distraction, haste, etc., the current selection state for candidates/choices would be displayed correctly in the summary pages as well as if the user were to re-visit the page, either by browsing back or through the re-vote process. This is because when a page is displayed anew all the contests are displayed using the current selection states for the candidates.

Our analysis was aided by the fact that the iVotronic code is single-threaded. The only source of concurrency is via interrupt handlers.

Enabling factors present in Sarasota: The relative slowness of the touch screen controller may have contributed to some voter dissatisfaction and comments that it was difficult to make selections in some races. This could have been exacerbated in those contests that appear in particular screen positions, since in our testing of the touch screen we noticed that screen responsiveness could vary as a function of finger angle to screen, for instance. The fact that the CD13 race was the first on the top of a page may have been a factor in an increased number of complaints by voters that it was difficult to record choices in that race.

Mitigating factors: Ideally, all touch screens should be re-calibrated and tested prior to an election to ensure performance quality parameters.

6.2.2 Findings Regarding Recording of Votes

6.2.2.1 Prospective Software Faults During Transfer From Volatile to Non-Volatile Memory.

Overview. The iVotronic voting process requires the votes registered on ballots in volatile memory (RAM) be transferred to non-volatile memory (the internal triply-redundant terminal flash) when

the voter completes their selections and casts their ballot by pressing the vote button. Such data transfers are a natural place to examine in response to the reported anomalous data patterns.

Hypothesis. We analyzed the iVotronic firmware to identify potential flaws that could cause multiple vote images to be modified by a single voter session. If possible then this would reduce the necessary occurrence of an error from 10,000+ (the number of undervotes) to several hundred (the number of voting machines), since a single error in each machine could affect all votes on that machine.

Technical Analysis. The team spent combined approximately twenty hours reviewing all the code capable of writing to the non-volatile (terminal flash) chips. There are three terminal flash chips, each of which contains a copy of the ballot images (e.g., each voter's selections on the ballot). The process of writing to a terminal flash is more complex than writing data to RAM since each write operation to terminal flash must be preceded by a write-enable command. The write enable requires that a specific command word be written to the start of the flash memory sector; without the write-enable command, any subsequent write operation would be nullified. This makes it much more difficult for an errant memory write operation to corrupt the ballot images stored in terminal flash. Code that enables writes to the sectors containing ballot images exists in a single location in the firmware code. This location performs a write of a single, complete ballot image (copied from RAM). The only other operation performed on these sectors of the terminal flash memories is complete sector erasing.

A close inspection of the code reveals that actions from a single voter cannot modify the ballot images previously written to terminal flash. Furthermore, the code that writes a new ballot image to terminal flash is careful to avoid overwriting previously written ballot images: it identifies where current ballot images reside and writes the new ballot image to an unused portion of the sector. There is some randomization in the selection of which sector a ballot image will be written (to provide anonymity to the voter by avoiding ordering ballot images in the same manner as the votes were cast), but proper checks are made to avoid overwriting prior ballot images or writing partial ballot images due to memory storage full condition. Once a sector is chosen, ballot images are stored sequentially in that sector until that sector is full. If the selected sector is full, an alternate sector will be chosen.

The code reveals that it is possible that a ballot image would not be written if all sectors were full when the write is attempted. This event would generate an emergency shutdown condition, all prior ballot images would be retained in persistent memory, and the event log is appropriately updated to reflect the emergency shutdown condition. This event is unlikely to occur in practice due to the capacity of the terminal flash to hold many ballot images.

Mitigating Factors Present in Sarasota. Most of the voting machines contained dozens or hundreds of ballot images at poll closing. This is far fewer than the storage capability of the terminal flash memories. No voting machines contained enough ballot images to fill the terminal flash memory necessitating an emergency shutdown.

6.2.3 Findings About Terminal Closing Processes.

6.2.3.1 Examine Potential Flaws When Transferring Results from Terminal Flash.

Overview. We analyzed the firmware to identify potential flaws in accumulating and extracting results from the iVotronic. Results may be extracted either by writing the terminal flash images (event log and vote images) to an inserted compact flash card or by uploading them through the serial port/modem.

Enabling Technology. Two separate routines are used for reporting terminal flash results to a compact flash card or to the serial/modem port.

(1) In the case that the results are written to a compact flash, one routine is called. This function checks to see that enough space is available on the compact flash card to hold either one terminal flash image (2 megabytes) or all three terminal flash images (6 megabytes). The choice of whether to copy one flash image or all three is made by the user and selected via an administration screen. If enough room is available then the contents of either one or three of the terminal flash images is written to the compact flash card. This operation writes 31 sectors of 128 blocks of 512 bytes per block. The last sector on each terminal flash chip is not written, but this sector contains only utility information not relevant to any audit. The actual data transfer is performed by a COTS library function po_write(), which provides a low-level interface to the compact flash.

(2) Reporting flash images over the serial link is performed by a different routine.

These two methods of reporting correctly reflect the contents of the terminal flash memory relevant to accessing vote image and event data. They also serve as a semi-independent check of the functionality of the other.

6.2.3.2 Do Audit Log Functions Record Events Properly?

Overview. We consider whether the audit log functions can fail to record events that they are called to record.

Hypothesis. An event occurs that should be listed in the audit log, such as closing the polls and reopening them, but is not shown when the log is examined. This situation would not cause or explain any undervotes itself, but it would explain how an auditable action that would cause the undervotes might not be recorded.

Enabling Technology. We consider how the audit log subsystem works, focusing on the logging functions. On the iVotronic, the audit log is also known as the event log.

Technical Analysis. The audit logs are stored in terminal flash during the voting. Three copies of the log are kept, and they are written sequentially (that is, the first copy is written, then the second, then the third). Audit log records are stored in an area called the “event queue”, which is stored in specific flash sectors. The bytes in terminal flash memory are bitwise initialized to zero. In what follows, think of the event queue as an array of event records stored in the elements of the event queue.

An event record consists of a numeric event code, the time the event is recorded, the serial number of the PEB involved, and the type of PEB (Supervisor or Voter). When the event log subsystem is initialized, it sets a variable to the beginning of the unused section of the event queue. It assumes events are written sequentially, so if there are events in the queue, it skips over them until it finds the first unoccupied element. This initializes the audit log subsystem.

The routine that records events takes a parameter indicating the event to be logged. The record is constructed and added to the event queue and is written to the event log in terminal flash. If the event queue is full, an “emergency close” routine is called. That routine immediately calls the routine that records events, causing an infinite recursion. The calling stack would grow until a hardware fault occurs. This could overwrite much of memory. It would undoubtedly cause the terminal to crash, freeze, lock up, or cease operating properly.

Mitigating Factors Present in Sarasota. While there were several reports of terminals locking, audit logs did not show improper terminal closing or events that would indicate that this situation occurred. Moreover, event log storage space is sufficiently large that only an extraordinary voter volume could cause memory to fill.

Potential Remedy. The vendor could fix the interaction between the routine that records events and the "emergency close" routine to handle the case of the audit log being full. Also, election officials could check that all terminals are properly closed on Election Day, and that closing is properly reflected in the audit logs.

6.3 Findings Related to Asynchronous Processes

Overview. While the iVotronic has only one main thread of control, it does include hardware interrupt handlers, which read and/or update global variables. When a variable is read or updated concurrently by the main thread and interrupt handlers, there is a risk of timing-dependent errors, usually called "race conditions". Race conditions are difficult to detect in testing, because the combination of event timings that results in erroneous behavior may be very rare, and may depend on random events and minor variations in hardware tolerances that cannot be directly controlled or reproduced. Therefore, one cannot rule out, a priori, the possibility of timing-dependent errors involving asynchronously updated variables in the iVotronic.

Hypothesis. A timing-dependent error involving interaction between the main program and hardware interrupt handlers might cause erroneous behavior on some machines during the election, that would not show up on other machines or during pre- and post-election testing.

Enabling Technology. We considered the interactions between interrupt handlers and the main thread, through global variables, looking for potential race conditions.

Technical Analysis. We searched the code for indications of multi-threading. While there are comments in a few places that mention "thread" and "multi-tasking", we were unable to find any indication of multiple threads in the executable code. As far as we can tell, the iVotronic software runs directly on the main processor hardware, with no operating system kernel, and the main program is the only thread of control other than the asynchronous hardware interrupt handlers. Apparently, the main program is reloaded and called each time a new voter session is started.

We reviewed all the sources of hardware interrupts, and all of the hardware interrupt handlers, to understand their interactions with the rest of the iVotronic software. We enumerated all of the global variables read or modified by hardware interrupt handlers or by functions called (directly or indirectly) from hardware interrupt handlers. We then examined the places in the main program and the subprograms called by the main program where references are made to those variables.

We first verified that the asynchronously updated variables are all of a size that permits them to be read and written atomically by the main program; that is, it is not possible for a hardware interrupt handler to execute between the reading/writing of one byte of the variable and the reading/writing of the rest of the variable. All of the asynchronously updated variables passed this check.

We then attempted to verify that all such variables were declared as "volatile", so that the compiler would not perform unsafe optimizations (e.g., suppression of apparently-redundant load and store operations) on them. Most of the asynchronously updated global variables were not declared to be volatile, but we do not believe this mattered with the particular compiler used on the iVotronic software. That is, with there being so many cases, if the compiler performed optimizations of the kind that would be unsafe on these variables: (a) the results would probably have been detected in testing; (b) the probability of failure would have been uniform over time, affecting all races with equal probability; (c) the failures would be exhibited in ways other than just undervotes.

We next classified the uses of the asynchronously updated global variables, according to usage. Most of the uses conformed to one of the following generally-safe paradigms:

(1) Count-down timers. A software count-down timer is a global variable that is decremented periodically, in these cases by the hardware timer interrupt handler, until it reaches zero. The main

thread uses a count-down timer to delay for a given length of time, by setting the timer to a positive count (usually a count of milliseconds) and then looping until the timer value has reached zero. This usage pattern is generally free from dangerous race conditions, so long as the variable is of a size that can be read atomically by the main thread, and only the main thread sets the value of the timer. If an interrupt handler may also reset the timer, it is possible for the main thread to delay for a longer or shorter time than expected. The iVotronic software contains several variables of this type, though we believe the code could be simplified by consolidating some of these timers.

(2) Counters. A software counter is a global variable that is incremented periodically, in these cases by an interrupt handler. The main thread uses a counter similarly to a count-down timer, by initializing the counter to zero, and then looping until the value passes some limit. This usage pattern is generally free from race conditions, so long as the variable is of a size that can be read atomically by the main thread, the code that increments the counter stops before the variable can overflow, and either the main thread uses ">=" or ">" (rather than "=") to check the timer, or the incrementing code stops at some moderately small value. The iVotronic software contains several variables of this type. They appeared to be used correctly. However, the code could be simplified and made more readable by consolidating some of the counters, and by adopting a more uniform policy of using just the count-down or just the count-up paradigm, rather than the present apparently arbitrary alternation.

(3) Read-only state variables. A read-only state variable is updated by the interrupt handler and read by the main thread. This usage pattern can be safe if the size of the variable allows for it to be read atomically by the main thread, and the logic of the main thread takes into account the volatility of the value of the variable. The iVotronic software contains many variables of this type, including those that keep track of the device model and serial number (presumably not changing, once set), whether a PEB is currently inserted and the type of PEB inserted (changing, but not ordinarily changing during the casting of a single ballot), and the X and Y coordinates of the last event on the touch screen (changing rapidly). While we did not find any specific errors in the usage of such variables, we did find that the need to continually poll for changes in these variables made the logic of the main thread difficult to follow.

We also verified that the interrupt handlers were either coded in a re-entrant-safe fashion or took steps (e.g., disabling interrupts) to ensure that they would not be called re-entrantly. We also examined all code that disabled interrupts for a lengthy period of time; no problems were detected.

A characteristic of this software architecture, in which the main program polls for changes made to global state variables by interrupt handlers, is that there may be variable delays in the responses to external events, depending on what the main program is doing when the event occurs and how soon after the event it checks the corresponding global state variable. In the worst case, an event may fail to be detected entirely, if the main thread does not check the corresponding global state variable before it is again modified by a subsequent event.

For example, when a voter touches a location on the touch screen, an interrupt handler records the fact that the screen has been touched in a global state variable, and also records the X and Y coordinates of the touch. If the main program does not check these variables before the voter touches another location, the first touch will be ignored. This is consistent with observed behavior of the iVotronic, i.e., if a person touches two locations on the touch screen in rapid succession the system will ignore the first touch. There is no problem in this case, since the voter can see whether each touch was registered by whether a corresponding "X" that appears on the screen. In fact, this behavior may be desirable, since the last touch point would normally be the one the voter intends.

Finally, we examined whether the various libraries we did not read (see Section 5.6) may have interrupt handlers or sections where they disable interrupts. We considered this possibility highly

unlikely for three reasons (1) We did not see any functions performed by library calls that would logically require an interrupt handler. (2) The library code is a generic, off-the-shelf product intended for embedded applications. It would put too large a burden on customers to design systems that use interrupts if the users did not have visibility of all interrupt handlers. To avoid IRQ number conflicts, the application would need to be involved in setting up the mapping from IRQ number to handler entry point. (3) In the section of code that sets up interrupt handling vectors, we did not find any references to function entry points not present in the code we reviewed.

6.4 Findings Related to Election Audits

6.4.1 Suggestions to Improve Audits

Overview. During this review we realized that certain enhancements to the iVotronic audit logs could have made the code review and audit easier and/or more complete. Based on our experiences in this work, we believe that there are opportunities to augment voting systems in ways that would significantly enhance our ability to perform meaningful election audits after the fact. We present these observations here.

Paper Trail. A paper trail would have served to confirm that votes were not altered after they had been recorded. In this case, the code review to check that ballot images were not altered after they had been recorded was fairly easy, if we assume the absence of malicious activity. If some voters verified that the paper trail was an accurate record of their intent before casting their ballot, then the number and contents of spoiled paper records would provide additional evidence to an audit regarding how many voters reached the review process without selecting a candidate in CD13 and how many successfully voted in CD13 thereafter. However, there is no reason to believe that a paper trail would have prevented the anomalous undervote. If many voters did not check the review screen, it seems likely that, all else being equal, many voters would also fail to check the paper trail. All in all, a paper trail might have provided some additional information to an audit, but likely would not have prevented the high undervote rate and likely would not have eliminated the controversy.

Voter Action Log[1]. If the audit logs had been expanded to record all user interactions with the system, this would have permitted a more detailed analysis of the cause of the CD13 undervote. This expansion appears to be feasible, since the storage capabilities of modern voting machines far exceed the requirements for logging screen touches and screen contents.

Such a system would have two major advantages and one minor advantage not found in existing touch screen audit mechanisms. The first is that issues of voter confusion of the ballot structure or machine interactions can be studied at the conclusion of the election. This can provide valuable feedback for improving the specifications for ballot designs and for the operation of future voting systems.

The second is that a full log of all user interaction might reduce our reliance on code review and enhance confidence in the results of any audit. The most complex code in the iVotronic is the user interface where selections were made and displayed. The undervote question involves whether the voter selection was an accurate reflection of what the voter saw on the display. The complexity of the user interface code made it difficult to answer this question with confidence. A log of all user interaction would provide a way to sidestep this difficult code review problem. Auditors could inspect the log to examine voter actions for evidence to infer display accuracy rather than studying

[1] This idea originated in conversations between David Wagner and Steven Bellovin unrelated to this report.

the code to try to predict whether the code will always display the appropriate information under all possible foreseeable circumstances.

The third advantage, admittedly minor, is that a full event log containing all user interaction enables a semi-independent way of tallying the votes assuming the ballot used is known and the voting terminal displayed the ballot properly. An independent system can read the event log and, using only that information, can count the votes for each candidate in each race. While this does not help to verify that the votes were recorded correctly in the first place, it might provide a way to check that the tabulator summed up the votes correctly. The system or systems calculating votes from voter logs can be developed by a different company than the electronic voting machines, or by multiple different developers.

One problem is that there are significant unsolved vote secrecy problems with this mechanism. A full log of all user interactions creates a covert channel through which a voter could transmit evidence of how she cast her vote. This enables the voter to sell her vote or to be coerced, and, of course, such an interactions log would have to be stored in a way that does not compromise the privacy of the voter.

7 Security-Related Findings

As one component of the code review, we analyzed the security of the iVotronic firmware to determine whether fraud or computer intrusion could have caused or contributed to the CD13 undervote. This section details our findings about the security properties of the system that might be relevant to the CD13 undervote.

We discovered several software vulnerabilities in the firmware. We are convinced that none of them were exploited in Sarasota in a way that would have caused or contributed to the CD13 undervote. We present these threats as pertinent to this report under the Statement of Work because we cannot absolutely rule them out as a possible contribution to the undervote.

Our security findings relate to external data in four areas: PEBs, Compact Flash cards, modem operations, and password handling. External communications are natural targets that intruders might try to attack. While our analysis of modem operation did not reveal any software vulnerabilities, we discovered software security vulnerabilities in the other three areas. It is our assessment that none of them were exploited in Sarasota. We give our reasoning in more detail below.

There is a natural inclination to try to represent the following attacks with a probabilistic model that identifies a list of preconditions that would be necessary before an attack is possible and then treat each of these preconditions as independent events. If each such event is unlikely and the events are independent, then the probabilities multiply, yielding an attack likelihood that is statistically insignificant or even indistinguishable from zero. For instance, if we identify 10 such events, each occurring with probability 1/2, then the total probability of a successful attack is less than one in a thousand.

While this argument is intuitively appealing, it is also inherently flawed. There are two problems. First, these events are not independent. Often, if we assume a sufficiently motivated and skilled adversary, many or even all of the preconditions may pose no problem for such an attack.

Second, there is no way to scientifically or systematically assign probabilities to the events. This is true for many reasons, but we give two here. First, there are no current or historical records upon which to found an estimate of these probabilities. We can prognosticate about the likelihood that someone can, for example, steal a voting terminal from a controlled space, but prognosticating is the best we can do and different prognosticators may predict dramatically different values with no scientific way to reconcile the difference.

Finally, attacks are not random. Attacks are deliberate human acts, not acts of nature. This makes the presence or absence of attacks hard to predict and limits the usefulness of probabilistic models based on random behavior. Further, we are not aware of any Byzantine models that capture the particular features of this situation.

In Table 1, we identify conditions that would have to occur for an attack to be successful. However, as argued above, the number of conditions found in Table 1 cannot be used as a measure of the ease or difficulty of attack; instead, a more nuanced analysis is necessary.

Conditions to Exploit a PEB Virus
Must be a malicious, sophisticated intruder
They must acquire:
- one or more voting terminal(s)
- one or more PEB(s)
The virus must:
- be effectively injected
- propagate
- execute its designed attack
- delete any trace
Accomplish all of this undetected

Table 1. Virus Conditions

Table 1 is a simplification, and it disregards factors that may influence the difficulty of exploiting these vulnerabilities. For instance, source code for the iVotronic firmware would certainly facilitate development of the attacks described below, though it is probably not a necessary precondition. In any given circumstance, other items or knowledge may be necessary or helpful to execute an exploit.

7.1 The Virus Threat

We identified several buffer overflow vulnerabilities that in a worst case scenario may allow an attacker to take control of a voting machine by corrupting data on a PEB. These create the possibility of a virus that propagates by exploiting the buffer overflow vulnerability. Viruses pose a serious threat to computer system integrity. Procedural and physical security defenses may reduce or mitigate virus risk but cannot guarantee attack prevention. Unfortunately, the testing procedures that are standard practice in the elections community are unlikely to discover these vulnerabilities or the presence of a virus. The vulnerabilities might be found through careful analysis of the voting machine's source code (as we have done). While it may also be difficult for a prospective attacker to discover these weaknesses, their presence opens a door for attack.

If these vulnerabilities were exploited, it would be possible to hide their existence. A cleverly constructed virus can cover its tracks so that infected machines could not be detected by ordinary means and an appropriately programmed virus could self-destruct and erase all its tracks.

It is possible that an outsider could trigger an attack and that once one machine is infected, the virus would spread from machine to machine through removable storage media without further attacker involvement. We give a detailed description of potential virus exploits in Appendix B.

7.2 Vulnerability Verification

Buffer overflow vulnerabilities are well understood, both practically and in the literature. There is no doubt that the bugs that allow buffer overflow attacks to occur are present in the iVotronic firmware. However, we did not implement exploits for any of these vulnerabilities. Moreover, in what we believe to be unprecedented cooperation, the vendor (Election Systems & Software (ES&S)) offered to provide us with iVotronic equipment and technical analysis so we could implement these exploits for demonstration purposes. We declined their invitation for two reasons. First, we are confident that we could implement a rudimentary attack in a reasonably short period of time, but we believe such a simple exploit is not revealing. Several laboratory attacks, e.g. [11, 12] provide convincing evidence that academically identified buffer overflows in voting systems can be

exploited in laboratory environments. Thus, there is little scientific benefit in constructing another elementary attack. Alternatively, we considered attempting to construct a more sophisticated attack, with all the features that a real attacker might implement. In the end, the team decided that such an effort was not necessary for our analysis. Nonetheless, we appreciate the vendor's willingness to provide the resources we would have needed.

7.3 Buffer Overflow Overview

A buffer overflow is a computer attack that results from copying more data than the destination area can hold, which results in writing over other data. Any buffer overflow bug constitutes a potentially dangerous defect. In the absence of malicious intent, it can produce unpredictable program behavior, but when the data being copied is carefully constructed, it can allow an attacker to transfer program control to her own malicious code. Once this happens, the attacker controls the machine.

Buffer overflows result from trust that the software places, inappropriately, in data from an external source. In the iVotronic firmware, the software implicitly trusts that the election definition file in the terminal's flash memory was generated by a legitimate entity. This assumption is not universally justified.

Not all buffer overflow defects are exploitable. Input filters, operational procedures, and even good fortune may establish an environment where a buffer overflow cannot be exploited to take control of the machine.

7.4 Propagation Mechanisms

Viruses that can infect only a single machine or a few machines are rarely dangerous. It is well known that viruses can propagate through removable storage media. The two prospective removable media on the iVotronic each have software security vulnerabilities.

7.4.1 Compact Flash (CF) Cards

The CF card stores an election ID file and a set of audio (.WAV) files that support various election functions. The code that reads one of these files from the CF card exhibits a classic buffer overflow vulnerability. It reads a variable-length string from the CF card and stores it into a fixed-size array in memory without size or other validity checks. If a malicious party embeds the data on the CF card, an overly long string can overwrite the return address on the stack and cause execution to jump to malicious code that was loaded into memory from the CF card.

To assess the risk associated with the CF vulnerability, we contacted Florida election officials, Sarasota County election officials, and vendor employees to understand how CF cards are handled during election administration. Here is our understanding of the processes in use in Sarasota County. (Any errors in this description are our responsibility. We thank election officials for their assistance in understanding election processes.)

Before the election, the election is set up on the election administration server at a central county location. A single CF card is written with the files needed for the election, from this server. This card serves as a master copy. The data on the master CF card is then duplicated, using CF duplication equipment, onto hundreds of CF cards, one for each iVotronic machine. Only four highly trusted permanent employees of the Sarasota County elections department have access to the elections administration server and to the master CF card before duplication. Before the election, a duplicated CF card is inserted into each iVotronic machine and the case is sealed with a tamper-evident seal by county election workers. Consequently, all CF cards contain exactly the same data before the start of the election, since they were duplicated from the same master copy.

The iVotronic machines are transported to the polling place with the tamper-evident seal intact. In ordinary operation, poll workers never need to disturb the tamper-evident seal. After the election is over and the machines are returned to the county warehouse, two county election workers verify that the seal is intact on every machine before removing the CF card. Consequently, the tamper-evident seal protects the CF card from the time the CF card is inserted by county workers until the time when the CF card is removed by county workers. Any attempt to gain access to the CF card before then will presumably be detectable because it would involve breaking the tamper-evident seal. After the election, the CF cards are inserted into a CF reader attached to a laptop or server at the county warehouse, the contents of the CF cards are uploaded to the election administration server and archived, and the CF cards are stored for reuse in a subsequent election.

This process seems excellent from a security point of view. Each CF card is associated with a single iVotronic machine. CF cards are not shared between machines, so there is no likelihood that they would form a route for infection to spread. The contents of CF cards are erased between elections, so even if a CF card were to come to contain malicious data during the course of one election, that malicious data would be overwritten before the CF card is inserted into another iVotronic machine for the next election.

For these reasons, assuming the above procedures were followed, we believe that the CF cards posed a very low risk of spreading viruses in Sarasota County. Given that the above procedures were followed, we do not see any way that an outsider could have injected a virus and caused it to spread among Sarasota machines using CF cards. It is to Sarasota County's credit that their procedures regarding the chain of custody and security protections for CF cards are able to defend against even unanticipated security threats such as this one.

7.4.2 The Insider Threat

The greatest security threat to any computer system is the insider threat. This certainly applies to voting systems.

We illustrate the hypothetical insider impact by discussing the master CF card. The master CF card is a critical item. If that card contains malicious data when it is duplicated, then the malicious data will be duplicated onto all CF cards, which might then cause the infection of all iVotronic machines. If the election administration server is compromised, the CF card could be loaded with malicious software at its source. Alternatively, if someone were able to swap the legitimate master CF card for an illegitimate CF card that had been prepared in advance, they could arrange that the illegitimate CF card contained malicious data. Thus, the procedures for handling of the master CF card before it is duplicated are critical. CF cards are small devices, about the size of a postage stamp. This would make it easy to conceal a replacement card.

One significant mitigating factor in this case is that under Sarasota County procedures, only four highly trusted individuals are authorized to access the election administration server and the CF cards. This reduces the risk because it limits the number of people with an opportunity to exploit this vulnerability.

It was outside the scope of this report to perform a comprehensive review of the physical and operational security of the Sarasota County elections department.

7.4.3 The Potential for a PEB Virus

In a second removable media vulnerability, the PEB is also a potential virus propagation vehicle. Once a device (terminal) is infected by a PEB, that terminal may infect other PEBs inserted into it. Thus, if PEBs move between devices within a precinct, the virus could spread from machine to machine and from PEB to PEB. If terminals move from precinct to precinct, the virus can propagate

throughout the county over time. Though the vulnerability that we discovered depends upon what operations are invoked on the machine (see Appendix C), it is still possible for a PEB virus to propagate if those operations are triggered with sufficient frequency. For example, triggering the voting operation with a malicious PEB will not propagate a virus, but opening the polls or printing reports on an iVotronic may pass on a virus.

This is a critical point in analyzing a potential virus exploit in the CD13 race, because an infected PEB cannot propagate the virus unless a terminal with that PEB inserted executes an unsafe operation. We detail this mitigating factor in paragraph 7.5.2.

Thus, an attacker may need to target a Master PEB in order to improve propagation likelihood. The Master PEB is important because it is used to open every terminal in a polling place, and the process of opening the polls is an unsafe operation. Thus, an infected Master PEB might infect every terminal in a polling location, though the virus could only spread to other precincts during a subsequent election.

A sophisticated virus attack might also attempt to infect a supervisor terminal. Supervisor terminals are a central point of risk, since an infected supervisor terminal can infect many PEBs prepared for a given election. We emphasize the need to carefully guard access to supervisor terminals and limit the operations that are performed on PEBs that are inserted into them.

The PEB vulnerability arises from an architectural flaw in the iVotronic source code design. During our source code analysis, we found many PEB-related security bugs that could be used by a virus. These bugs were similar in nature and are instances of the same architectural flaw. Significant additional discussion about PEB viruses appears in Appendix B.

7.5 Mitigating Factors

7.5.1 Supervisor Terminals in Sarasota

We noted above that supervisor terminals make excellent virus attack targets because they can have a much wider impact than voting terminals. The procedures in Sarasota mitigate the risk to some extent. For example, in Sarasota, supervisor terminals are stored in the secure Data Acquisition Reporting Center (DARC room) within the Supervisor of Elections office. Sarasota maintains fifteen supervisor terminals and uses only a subset of them for each election. For example, in the November 2006 election, Sarasota used six supervisor terminals. While infecting one supervisor terminal would be damaging, this policy would likely localize the impact.

7.5.2 Propagation Limited by PEB Operations

As we mentioned above, terminals can only be infected by corrupted PEBs if certain operations are executed while those PEBs are inserted.

It is important to note that the hypothesized PEB virus cannot be passed during the most frequent, and in many cases exclusive PEB operation: voter initialization. That is, the voter initialization operation is *safe*.

Also, an attacker might find it difficult to build a PEB that exploits all unsafe operation without noticeably interfering with the safe operations. If this were the case, it could complicate construction of a virus or slow its spread. We have no specific reason to believe this to be the case, but because we have not implemented a working exploit for the reasons stated above, it is hard to know what difficulties a virus writer might face.

7.5.3 PEB Handling Procedures

Removable media virus propagation properties are well understood and are easily estimated when we can assume random assignment of machines and media and machine to polling places. In our initial analysis, we generated a simulation that assumed such random behavior. Using that data, we estimated that a PEB virus would take four to six elections to propagate throughout the county. Closer evaluation identified PEB and terminal handling procedures that mitigate this threat. Specifically, PEBs do not move between precincts between primary and general elections. Thus, even if all PEBs in a particular precinct became infected during the primary, they would not be distributed among other precincts so the virus could not propagate further via PEB distribution between the primary and general elections.

Of course this does not prevent propagation since infected terminals can also spread a virus. In Sarasota, terminal distribution is less uniform, but definitely non-random. Specifically, terminals are stored on pallets in the county warehouse between elections. If after a primary each terminal from an infected precinct were assigned to a different precinct for the general election, the number of terminals in that precinct is the upper bound on the possible number of propagated precincts. More realistically, we were told that Sarasota stores terminals in the warehouse in such a way that they naturally retain a temporal clustering. While they may not be reassigned to the same precinct in the next election, they are likely to be removed from the warehouse and assigned to polling places in the next election in an order that is correlated to the order in which terminals were collected in the last election, causing a clustering effect. This kind of clustering would slow the propagation rate.

7.5.4 PEB Inventory Control

We were informed that, in Sarasota, PEBs are bar-coded and carefully inventory controlled. During non-election periods, they are stored behind three-tier locks, within the supervisor's office, inside the security controlled DARC room, and locked in cages. Of course an attacker may obtain a PEB from somewhere other than Sarasota County, but it is noteworthy that Sarasota strongly protects their PEBs between elections.

7.5.5 No "Shrink-wrap Effect"

One factor facilitating the spread of viruses on the modern Internet is "the shrink wrap effect", where many users use the same software and where attack mechanisms are well known and are even published on the Web. Because shrink-wrapped software is in widespread use, there are many potential targets and there are many people able to acquire the information and skills necessary to create such a virus. The iVotronic architecture is not subject to the shrink wrap effect. It is special purpose hardware and software whose architecture and implementation details are protected from wide distribution. Only sophisticated attackers with specific goals could exploit these vulnerabilities and they could only confidently perpetuate the spread of such a virus with extensive preparation and perhaps a bit of luck.

7.5.6 Virus Developer's Tradeoffs

While virus writers may exercise an immense variety of attacks and deception techniques, these techniques are subject to tradeoffs. For instance, if an attacker chooses to propagate a virus from machine to machine, she introduces the possibility that the virus could be detected by someone who knew how to look for it. FLDoS conducted such an integrity check during the ongoing audit process. In their test of six iVotronics terminals used during the election, FLDoS extracted the removable iVotronic firmware EPROM chips, placed them in a commercial EEPROM reader, and saved the firmware image into a bit-image file. They compared these extracted images to an image

from the software's secure build process conducted by the federally approved independent testing authority that certified the iVotronic and they were identical. This provides strong evidence that no virus was resident in these iVotronics after the election and therefore strongly suggests that no virus was present on Sarasota terminals after the election.

Conversely, if virus writers elect to cover their tracks and destroy all evidence after they accomplish their task, they limit their impact by limiting their ability to perpetuate themselves.

7.5.7 Controlled Hardware and Software

The PEB is a special purpose device that is not available off the shelf. While limited availability does not provide strong systematic security, it does eliminate the shrink wrap vulnerability.

Additionally, a prospective attacker would almost certainly need to acquire one or two voting terminals for use in preparing the virus, likely through theft or fraud. The attacker would need to prepare the attack well in advance, easily taking weeks or months of technical work to create such a virus. These factors significantly reduce the potential attacker population.

7.5.8 A Sophisticated Intruder or an Insider

Some activities we describe are most easily accomplished by trusted insiders. However, insiders accept risk of suspicion as well. Additionally the number of insiders with the access and opportunity to mount this kind of attack is limited and their identities and responsibilities are well-known.

A virus-based attack by an outsider would certainly require considerable technical sophistication and preparation. Such an attack certainly could not be mounted by the average person on the street, by the average computer user, or probably even by the average software developer. The attacker would need to be skilled in computer programming and in the exploitation of computer security vulnerabilities, with broad and deep understanding of computer software and reverse engineering.

7.5.9 Margin for Error

One challenge facing any would-be attacker is the low margin for error in mounting this kind of attack, and software developers well know that perfect software, including attacking software, does not exist. If the virus contains a bug or programming error that causes it to behave in a way different from how its creator intended, that bug might have effects that could disable the attack, cause it to be detected by election officials, or expose the attacker's identity and methods to forensic analysis.

Just as all application code has defects, attacker code is also subject to defects. Moreover, it would be difficult for an attacker to test virus operation rigorously in the lab before injecting it into the wild, so an attacker would have to be concerned about the possibility of bugs in her code. There is no clear way for an attacker to influence or control the virus after it has been introduced into the system, so if she wants to remain undetected, the attacker must plan to succeed on the first try. Even with the most careful precautions, complex first try attacks are not guaranteed to succeed.

7.5.10 Temporal Proximity

Another significant mitigating factor is that, because of the delay in the spread of the virus, unless the attacker has special insider access, the attacker would need to prepare the attack in advance and inject the virus well before the election that has been targeted for attack. For instance, if an outsider wanted to manipulate the November 7, 2006 general election, the attacker would have had to fully prepare and program the virus well in advance: at a minimum, because of the complexity of the attack, we believe that the virus would have to have been introduced before or during the August 2006 primary election and probably earlier, thus could not be candidate-specific. The virus would be a "fire-and-forget" weapon: the attacker probably could not change its programming or targeting

after it was introduced. This means that the attack would need to be highly premeditated and well planned. An attacker could not mount this kind of attack on the spur of the moment or on a whim.

7.5.11 Decentralized Election Administration

An additional mitigating factor is that because each county ordinarily administers its own elections and counties do not share equipment, a virus would not spread outside the boundaries of a single county. An attacker who wanted to influence the election in multiple counties would have to inject the virus in each targeted county, and introducing the virus requires some kind of physical presence. This cannot be performed by someone living in some other country on the other side of the world or even someone in a neighboring county because these devices are not connected (for example, by a network). The attacker would have no way of knowing whether their attack would successfully change the outcome of the election.

7.6 Assessing the Factors

Taking into account all of the factors examined above, we judge there are strong reasons to believe no such virus was present during the November, 2006 election. To explain the observed undervote rate in Sarasota, Charlotte, and Lee Counties [8] (also see Section 8.1) all being caused by a virus, we would have to assume that the attacker separately attacked each of these three counties, at a corresponding increase in risk of getting caught. Also, as the discussion above highlights, these attacks would require substantial technical sophistication and extensive advance preparation. If an attacker had the capability to mount such an attack, the attacker could have exploited this capability in a far less noticeable way (e.g., by silently switching votes from one candidate to another instead of creating a high, attention-grabbing undervote rate). It is not clear what would motivate an attacker to use these capabilities in this way. Furthermore, there are other plausible explanations for the CD13 undervote that do not require such unlikely assumptions.

Finally, we found absolutely no evidence of any attack in Sarasota County that caused or contributed to the CD13 undervote, although we acknowledge that a highly sophisticated, perfectly executed attack might leave no evidence.

7.7 Modem Communications

We also investigated whether a virus might be able to spread by modem. After the polls are closed at the end of Election Day, an iVotronic may be connected to an extra “communications pack” device. The communications pack contains a modem that can transmit the election results to the county’s central server’s Data Acquisition Manager (DAM) over the phone. After examining the iVotronic source code, we could not see any way that a virus could spread from the Unity server to an iVotronic machine. Very little data is transmitted from the Unity server to the iVotronic machine, and that data is handled by the iVotronic code in a safe way. We did not see any buffer overruns or other security vulnerabilities in the code that handles data received from the Unity server. Consequently, we believe there is no way to infect an iVotronic machine over the modem.

Moreover, Sarasota collection procedures do not involve connecting iVotronic machines to modems. Rather, PEBs are transported to four regional sites where they are entered into a laptop computer through a PEB reader and the results are reported via modem connection to the election central. The modem connection is manually synchronized via a separate phone connection.

7.8 Fixing the Virus Vulnerabilities

The misplaced trust in PEB data gives a prospective attacker several optional exploits that existed in the code during the CD13 election. It was beyond the scope of this review to identify an exhaustive list of all places in the code that may be vulnerable, since much of the firmware was not executed in Sarasota. All vulnerabilities must be eliminated or mitigated before the software could be considered secure.

Fixing these vulnerabilities is likely to be non-trivial because it requires fixing a flaw in the architecture and architectural flaws tend to be more difficult to fix once they are implemented. The software needs to avoid trusting inputs from untrusted sources. This would require introducing input validation and defensive programming through much of the code.

7.9 Procedural Defenses to Remediate These Vulnerabilities

Until the iVotronic firmware is modified to fix these vulnerabilities, there are a number of procedural defenses that election officials could use to defend against the virus threat.

1. Each terminal and each PEB should be assigned to a single precinct. This assignment should never be changed or rotated among precincts and should remain fixed for the lifetime of the equipment.

2. Master PEBs should be strictly controlled using procedures similar to those applied to paper ballots. They should be constantly under lock and key during the voting day, with sign-out and sign-in procedures to maintain the chain of custody at all times.

3. Polling place procedures should minimize PEB cross-pollination: i.e., minimize the number of terminals that any particular PEB is ever inserted into and minimize the number of PEBs that are ever inserted into any given terminal. For instance, officials might set an upper bound of 5, specifying that no PEB be used with more than 5 terminals and no terminal be used with more than 5 PEBs. Optimally, a poll-worker with a PEB would be assigned a set of terminals, no other PEBs would be used on those terminals, and that PEB would never be inserted in any other terminal.

4. Supervisor terminals should be rigorously controlled. No unsafe operation should ever be performed on any supervisor terminal, if it possibly can be avoided. (See Appendix C for a list of safe and unsafe operations.)

5. Numbered tamper-evident seals should be used to deter tampering with the CF card slot. Logs should be kept of all seals applied and/or removed, and two-person controls should be applied when election workers handle CF cards.

Many of these procedures are in place in Sarasota County and their practices inspired some of our suggestions.

7.10 Passwords

A general security review is beyond the scope of our task. However, we detected significant password weaknesses that may allow an intruder to inject a virus into a terminal if they are given unsupervised access. We could not construct any scenario where password exploit could have caused the undervote symptoms without injecting a virus into the system. See Appendix D for further discussion of the password issues.

7.11 Security Summary

Our security analysis revealed several software defects that could allow an attacker to introduce a virus into the voting system that spreads through removable storage devices. We cannot absolutely

rule out the possibility that an attack was mounted during the November, 2006 general election. It is in principle possible to mount an attack that would leave no trace after the election is over and it is impossible (by definition) to detect such an attack. However, we found no evidence of an attack and there are strong reasons to believe that these vulnerabilities were not exploited in a way that caused or contributed to the CD13 undervote.

8 Analysis of Hypotheses

Team members and others have proposed numerous hypotheses that might explain the observed undervote. This section of the report deals explicitly with these hypotheses.

8.1 Assumptions.

We make the following assumptions based on information furnished by the Secretary of State concerning tests and activities not performed by our team. We did not independently verify them. We give textual names to each assumption for ease of reference.

SOURCE MATCH. The software used on the subject DREs was generated from the same source code that was examined by the team.

CVR CORRESPONDENCE. No discrepancy was observed among the following: (1) the summary tape generated on Election Day at the close of polls on individual machines; (2) the individual cast vote records ("CVRs," or "ballot images") recorded by the machines; and (3) the totals that were accumulated and reported by Unity.

OBJECT MATCH. The software present on the machine's internal EPROM after the election was the software originally certified.

TEST CONFIRMATION. No behavior was observed during the Secretary of State's testing that would have caused any valid selection in CD-13 to be altered or recorded as an undervote [10].

CHARLOTTE and LEE UNDERVOTE. Charlotte County observed an undervote of approximately 26% in the statewide race for Attorney General, and Lee County had a similar though slightly lower undervote rate in the Attorney General race. In Charlotte and Lee Counties, the layout of the Attorney General race was similar to the layout of the CD13 race in Sarasota County [8]. Other contests without the multiple-contest-per-page format did not have a high undervote rate.

If we hypothesize that the CD13 undervotes were caused by deliberate fraud, not by human factors considerations, then our hypothetical scenario has to include an explanation for why the Attorney General race in Charlotte and Lee Counties had such a high undervote rate. In other words, such a hypothesis has to assume that the attack was not limited to just Sarasota County, but also affected Charlotte and Lee Counties.

FLORIDA UNIQUENESS. We are unaware of any other jurisdictions in the United States that used the same iVotronic version that reported undervote percentages of the Sarasota and Charlotte Counties' magnitude.

8.2 Relevant findings from our source code review

The following definitions reflect observations made by the team based on source code review. We investigated these issues in a systematic and structured fashion and found no evidence to contradict any of these properties.

COMPLETE BALLOT. We observed no evidence during our code review of any defects in the code that would cause anything less than the complete ballot to be presented to the voter.

PROPER DISPLAY. We observed no evidence during our code review of any defects in the code that would cause the display screens presented to the voter to inaccurately reflect the ballot from the PEB or the selections made by the voter.

ACCURATE VOTE DATA. We observed no evidence during our code review of any defects in the code that would cause the ballot images recorded to terminal flash when the voter casts her ballot to incorrectly reflect the selections made by the voter.

FULL RECORDING. The preceding three properties necessarily imply that when voters pressed the vote button, the CD-13 race was present on the ballot and, if the voter did not make any selection in the CD-13 race,, the screen for that race showed no vote for either candidate, and the review screen displayed the message "NO SELECTION MADE" in the CD-13 race. We note that our observations are consistent with the explanation that ballot design combined with the absence of a prominent undervote warning led to the high undervote.

NO MALWARE. We saw no sign of any malicious logic deliberately introduced into the code to rig the election by falsely recording undervotes.

NO SERIAL RACE EFFECT. No evidence was found that a selection (or lack of a selection) in any race affected any other race or question on the ballot in any way. That is, selecting (or failing to select) a candidate in race X did not affect the presence or absence of race Y on the ballot presented to the voter or the presentation of candidates in race Y, and did not affect the proper recording of the voter's selections in race Y or the appearance of the review screen in race Y. See Sections 4.1.7, 4.1.8, and 6.2.1.3 for further analysis.

NO SERIAL VOTER EFFECT. No evidence was found of any serial effect between voters. That is, as far as we can tell, the behavior of the machine for voter n+1 was not affected by any act performed or not performed by the previous voters 1 through n, assuming that voter n completed the act of casting a ballot.

NO TIME-SENSITIVE CODE. There is no indication of any time-sensitive code that would cause the machine to behave differently on Election Day than at any other time. We examined all of the source code that reads the clock and all of the code that uses any value based on a clock reading (directly or indirectly), and it was all innocuous. The amount of code in this category was limited enough that we were able to exhaustively analyze all of it, and we are confident that this code could not have contributed to an undervote.

NO VOTE PEB EFFECT. No condition giving rise to an "Invalid vote PEB" log event (of which 308 were recorded during the election) would cause the CD-13 race not to be displayed to the voter, cause a selection to be altered, or cause a valid selection to be recorded as an undervote. See Section 6.2.1.2 for further analysis.

NO SUPER PEB EFFECT. No condition giving rise to an "Invalid super PEB" log event (of which 48 were recorded during the election) would cause the CD-13 race not to be displayed to the voter, cause a selection to be altered, or cause a valid selection to be recorded as an undervote.

NO NETWORK EFFECT. No "networking" effects were observed, namely any condition occurring on machine B networked to machine A that would cause any ballot alteration or undervote on machine A.

NO PEB CLUSTER EFFECT. No "PEB cluster" effects were observed. That is, the fact that machines A and B in the same polling place were activated with the same PEB had no effect on any race on either machine or any other machine on which such PEB was used. In Sarasota, the PEB is removed from machine A before the voter votes, so no "state" caused by the voter can be transferred to machine B.

8.3 Malicious Software Hypothesis

If the software used on the subject DREs was not generated from the same source code that was examined by the team, then the team's observations from examining that source code would be irrelevant. If the software on the machines was different, it must have been altered before or during the election, the altered version used during the election, and the altered software must have been subsequently replaced by the original certified version since this is the version that is now resident in the machines.

If the undervote was caused by malicious logic deliberately introduced into the source code, we did not find any evidence of such malicious logic in the source code examined by the team.

8.4 Hypotheses Summary

8.4.1 Machines dropped selections made in the CD13 race, creating an undervote.

Contraindications: (see Section 6.2.1.6)

- TEST CONFIRMATION
- FLORIDA UNIQUENESS. If the claimed behavior were present in the certified iVotronic software, one would expect that it would have been observed in other jurisdictions using the same software.
- FULL RECORDING

8.4.2 Votes were validly cast in the CD-13 race but were erroneously reported as undervotes.

Contraindications: (see Sections 6.2.2.1 and 6.2.3.1)

- TEST CONFIRMATION
- FLORIDA UNIQUENESS
- FULL RECORDING

8.4.3 No selection made in the CD13 race, but the review screen showed a vote, creating an undervote.

Contraindications: (see Section 6.2.1.6)

- TEST CONFIRMATION
- FLORIDA UNIQUENESS
- FULL RECORDING

8.4.4 Machine did not display the CD-13 race to some percentage of voters

Contraindications: (see Section 2.5.4 and 6.2.1.5).

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- FLORIDA UNIQUENESS
- FULL RECORDING (especially PROPER DISPLAY)

8.4.5 The particular ballot style used in Sarasota County caused the machine to behave abnormally.

Contraindications:

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- FULL RECORDING

8.4.6 Some dynamic error not easily visible in the source code, e.g. buffer overflow or data left from previous voters caused the anomalous undervote.

Contraindications: (see Section 7)

- TEST CONFIRMATION. The error did not occur in testing, but would have had to occur with great frequency during voting.
- CHARLOTTE and LEE UNDERVOTE. Why did the problem occur in Sarasota, Charlotte, and Lee Counties, but nowhere else?
- FLORIDA UNIQUENESS

8.4.7 The touch screens were miscalibrated to prevent voting in the District 13 race.

Contraindications:

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- A very large number of machines would have exhibited the problem, and could not have been recalibrated before post-election testing. Thus, the problem would have been observed in testing.
- The undervote would have been much higher.
- Other races on other screens would have been affected but were not.

8.4.8 The touchscreens were miscalibrated so that the hotspot and corresponding candidate box were misaligned.

Contraindications:

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- A large number of machines would have exhibited the problem and could not have been recalibrated before post-election testing. Thus, the problem would have been observed in testing.
- Other races on other screens would have been affected but were not.

8.4.9 The touchscreen smoothing filter caused the undervote.

A smoothing filter is a mathematical procedure for damping transient touch screen effects such as the voter altering the position of her finger or changing the pressure exerted by the finger on the screen. The allegation has been floated on Internet newsgroups that the iVotronic touch screen filter could have caused the undervote. No explanation has been offered how the effect would confine itself to a single race on a single screen. The touch screen filter does not act differently on different screens.

Contraindications:

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- FULL RECORDING
- FLORIDA UNIQUENESS
- Other races would have been affected but were not.

8.4.10 A “controlling contest” specification linked CD-13 to a vote in a different race, thus affecting the voter’s selection in CD-13.

Contraindications: (see Section 6.2.1.3)

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- NO SERIAL RACE EFFECT
- The Sarasota County ballot styles did not contain any controlling contests (Section 4.1.8).
- No voter complaints about controlling contest messages (Section 6.2.1.3).

8.4.11 A "straight party" specification linked CD-13 to a vote in a different race, thus affecting the voter's selection in CD-13.

Contraindications: (see Section 6.2.1.4)

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- NO SERIAL RACE EFFECT
- The Sarasota County ballot styles did not enable straight-party voting (Section 4.1.7).

8.4.12 A "special event," such as a write-in or ADA voter, triggered an anomaly for this or subsequent voters resulting in the CD-13 undervote.

Contraindications:

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- FULL RECORDING
- FLORIDA UNIQUENESS
- Too few special events occurred to account for the undervote.
- Non-ADA machines also showed high undervote rates.

8.4.13 The actions of a voter in a race other than CD-13 affected the CD-13 race.

Contraindications:

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- FULL RECORDING
- NO SERIAL RACE EFFECT
- FLORIDA UNIQUENESS

8.4.14 Returning to a contest from the review page caused the CD-13 undervote.

Contraindications:

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- FULL RECORDING
- NO SERIAL RACE EFFECT
- FLORIDA UNIQUENESS
- Why would only CD-13 be affected?

8.4.15 A special language voter caused the CD-13 undervote.

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- FULL RECORDING
- FLORIDA UNIQUENESS
- Too few special language voters to account for the undervote.

8.4.16 A mishandled interrupt changed the state of the machine and caused the CD-13 undervote.

Contraindications: (see Section 6.3)

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- FULL RECORDING
- FLORIDA UNIQUENESS
- A mishandled interrupt bug would have had to have affected the majority of machines to cause the observed CD-13 undervote rate, which means it must have occurred with fairly high frequency on the election; but the fact that no problem was observed during testing means that it would could only have occurred with low frequency during testing.

8.4.17 There was an error writing from RAM to the terminal memories causing valid votes in CD-13 to be recorded as undervotes.

Contraindications: (see Section 6.2.2.1)

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- FULL RECORDING (especially ACCURATE VOTE DATA)
- CHARLOTTE and LEE UNDERVOTE
- FLORIDA UNIQUENESS

8.4.18 Having multiple contests on the same ballot page caused changes depending on the order in which the contests were voted.

Contraindications:

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- FULL RECORDING
- NO SERIAL RACE EFFECT
- FLORIDA UNIQUENESS
- Why would only CD-13 be affected? In Sarasota County, there were other ballot pages containing multiple races, but there are no signs that those other races were similarly affected.

8.4.19 Variables holding information about voters were initialized to incorrect values or not initialized at all.

Contraindications:

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- FULL RECORDING
- FLORIDA UNIQUENESS
- Why would only CD-13 be affected? In Sarasota County, there were other ballot pages containing multiple races, but there are no signs that those other races were similarly affected.

8.4.20 An extra ballot style without CD13 was present on the supervisor terminal and large numbers of voters received a defective ballot.

Contraindications:

- TEST CONFIRMATION
- CVR CORRESPONDENCE. The CVRs show that the race was present on all ballots displayed to the voters.
- FULL RECORDING
- Large numbers of voters would have reported this problem

8.4.21 The machine software made an error in determining where to write a ballot image, thereby overwriting parts of images previously written and deleting votes in CD-13.

Contraindications: (see Section 6.2.2.1)

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- FULL RECORDING (especially ACCURATE VOTE DATA)
- FLORIDA UNIQUENESS
- We studied the source code that is responsible for recording ballot images. That code was simple, clean, and well-structured. The amount of code in this category was limited enough that we were able to exhaustively review all of it. We are confident this code has no error that would cause previously recorded ballot images to be overwritten. See Section 6.2.2.1 for further analysis.

8.4.22 The actions of one voter affected the ability of the next or a subsequent voter to have a CD-13 vote recorded.

Contraindications:

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- FULL RECORDING
- NO SERIAL RACE EFFECT
- NO SERIAL VOTER EFFECT
- FLORIDA UNIQUENESS

8.4.23 Time-sensitive code was present on the machines to affect CD-13, but only during actual voting and was untestable before and after the elections (see Section 7).

Contraindications:

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- FULL RECORDING
- NO TIME-SENSITIVE CODE
- FLORIDA UNIQUENESS

8.4.24 An error caused electronic vote totals generated from ballot images to be written incorrectly to the closing PEB.

Contraindications: (see Section 6.2.3.1)

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- FULL RECORDING
- FLORIDA UNIQUENESS

8.4.25 Insertion of an invalid PEB (either vote or supervisor PEB) into the machine caused CD-13

to be affected.

Contraindications: (see Section 6.2.1.2)

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- FULL RECORDING
- FLORIDA UNIQUENESS
- NO VOTE PEB EFFECT
- NO SUPER PEB EFFECT

8.4.26 The networking of multiple DREs together in the same polling caused CD-13 to be affected.

Contraindications:

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- FULL RECORDING
- NO NETWORK EFFECT
- FLORIDA UNIQUENESS

8.4.27 The use of the same PEB or set of PEBs among machines in the same polling place caused CD-13 to be affected.

Contraindications:

- TEST CONFIRMATION
- CVR CORRESPONDENCE
- FULL RECORDING
- NO NETWORK EFFECT
- NO PEB CLUSTER EFFECT
- FLORIDA UNIQUENESS

8.4.28 The machines were tampered with after the election to erase the CD-13 votes on a high percentage of ballots.

Contraindications:

- CVR CORRESPONDENCE. The machine memories after the election (and presently) agree with the tallies produced and printed out at the precincts on election night. Therefore, the intrusion would have to have been made county-wide during the election and there is no evidence of such a widespread attack.

8.4.29 Firmware in the machines was tampered with to drop votes from the District 13 race and then erase itself before or at the close of polls, so no subsequent testing would reveal the intrusion.

Contraindications:

- CHARLOTTE and LEE UNDERVOTE. Any such attack should have been duplicated in Charlotte and Lee Counties, too.

8.4.30 Malware not present or visible in the source code was inserted into the machines in advance of the election to cause the CD-13 undervote.

Contraindications:

- TEST CONFIRMATION. Any such malware would have had to have erased itself before the testing.
- CHARLOTTE and LEE UNDERVOTE. Any such attack would have had to have been duplicated in Charlotte and Lee Counties, too.

9 Conclusions

There is no topic that compares to electronic voting with regard to diversity of security demands, inherent complexity, and the intensity of emotions it elicits in society today. Voting systems demand the highest integrity standards. Everyone wants them to be perfect, but every method of software verification and validation has limitations that leave the possibility of undetected faults. Software code review has been proven to be one of the most effective methods of recognizing and identifying faults, but no software review can claim to provide absolute assurance that software is entirely fault free.

This report presents the background, organization, process, findings, and opinions of our firmware code review. We conclude with the following summarizing statements.

9.1 We are confident that no iVotronic firmware bug contributed to the CD13 undervote.

9.2 Independent audits benefit from cooperation from vendors, election officials, and developers.

9.3 Our analysis suggests several important points regarding electronic voting software.

9.3.1 Electronic voting code review demands technical specialists and is resource intensive.

9.3.2 Strong standards and standards enforcement are essential to effective audit.

9.3.3 Statistical analysis can contribute to election auditing, but it cannot replace code review. Statistical analysis and code reviews, used in combination, can be more effective than either method on its own.

9.4 Electronic voting software needs to be secure. While properly implemented procedures can mitigate many threats, neither election procedures, code reviews, paper trails, rigorous testing, advocacy group oversight, nor any other mitigating factor can systematically ensure voting system integrity where faulty electronic voting system software is employed. Secure software, written to exacting and enforced standards, and carefully constructed election system procedures are necessary to provide electronic voting system integrity.

10 Acknowledgments

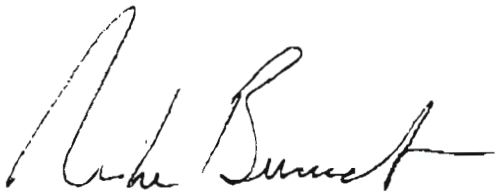
Due to this project beginning and continuing through the holiday season and the new year, we owe a debt of gratitude to several FSU staff and students for their support efforts *above and beyond the call of duty*. Leo Kermes, Jon Nilson, Louis Brooks, Tina Suen, Kenny Zahn, Randy Langley, Yu Wang, Rick Bessey, Greg Thompson, and Edwina Hall responded to our calls well beyond anything we could have expected and we thank them for their efforts.

As part of our work we used Fortify Source Code Analysis (SCA), made by Fortify Software, in order to assist with the code review process. Fortify Software donated the tool to us free of charge for use on this project and we thank them for their contribution. We note that two members of the team (Bishop and Wagner) are on Fortify Software's Technical Advisory Board.

11 Team Endorsement

Ted Baker

Matt Bishop



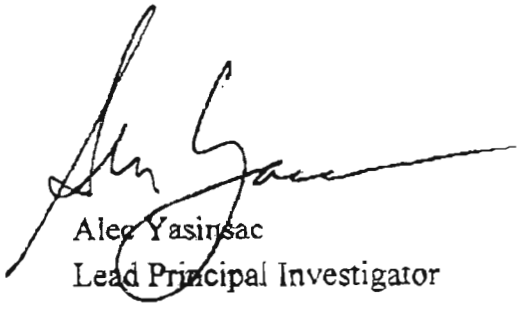
Michael Burmester
Co-Principal Investigator

Breno de Medeiros
Co-Principal Investigator

Michael Shamos

Gary Tyson

David Wagner



Alec Yasinsac
Lead Principal Investigator

12 REFERENCES

1 "Software Review and Security Analysis for ES&S iVotronic Voting Machine Firmware.", Florida State University Statement of Work, December 15, 2006, available on the web at http://election.dos.state.fl.us/pdf/FSUstatementWork.pdf

2 Dan Wallach, "Testing Undervote Hypotheses in Sarasota: Declaration of Technical Expert Dan Wallach", December 3, 2006

3 K. Coffey, M. Herron, et al., "Notice of Contest Regarding the Election for Representative in the One Hundred Tenth Congress From Florida's Thirteenth Congressional District", December 20, 2007, http://www.heraldtribune.com/assets/pdf/SH83691220.PDF

4 Victor Hull, "Buchanan, Jennings win in Dist. 13", HeraldTribune.com, September 6, 2006, http://www.heraldtribune.com/apps/pbcs.dll/article?AID=/20060906/NEWS/609060717

5 Matthew Doig, Maurice Tamman, "Analysis points to bad ballot design", HeraldTribune.com, December 5, 2006, http://www.heraldtribune.com/apps/pbcs.dll/article?AID=/20061205/NEWS/612050604/-1/xml

6 M. Mindy Moretti, "Poor ballot design again appears to plague a Florida election result", http://electionline.org/Newsletters/tabid/87/ctl/Detail/mid/643/xmid/229/xmfid/3/Default.aspx, December 7, 2006

7 Walter R. Mebane, Jr. and David L. Dill, "Factors Associated with the Excessive CD-13 Undervote in the 2006 General Election in Sarasota County, Florida", Internet Report, 01/18/07

8 Laurin Frisina, Michael C. Herron, James Honaker, Jeffrey B. Lewis, "Ballot Formats, Touchscreens, and Undervotes:A Study of the 2006 Midterm Elections in Florida", December 3, 2006, http://www.dartmouth.edu/~herron/cd13.pdf

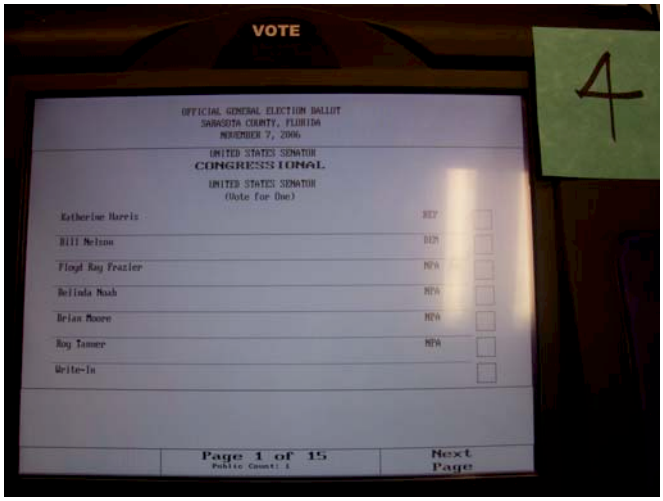
9 Maurice Tamman, "'Older' precincts added to problem", Sarasota Herald Tribune, Jan. 2, 2007

10 "Parallel Test Summary Report for Sarasota County, FL, November 7, 2006," December 18, 2006, Prepared by: Bureau of Voting Systems Certification, http://election.dos.state.fl.us/pdf/parallelTestSumReprt12-18-06.pdf

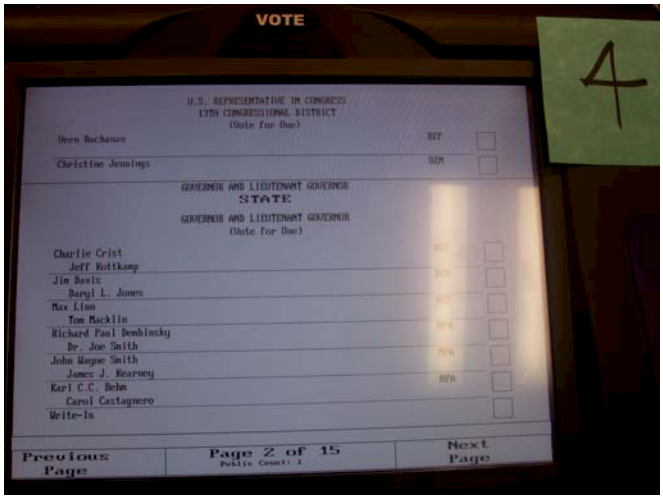
11 Harri Hursti, "Critical Security Issues with Diebold TSx", May 11, 2006. http://www.blackboxvoting.org/BBVtsxstudy.pdf

12 Ariel J. Feldman, J. Alex Halderman, and Edward W. Felten, "Security Analysis of the Diebold AccuVote-TS Voting Machine", Center for Information Technology Policy and Dept. of Computer Science, Princeton University, September 13, 2006

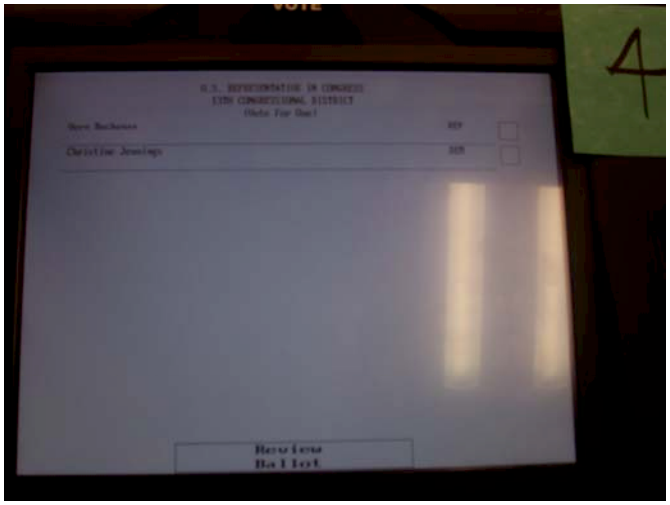
Appendix A CD13 Screenshots



1st Ballot Page, US Senate Race Only



2nd Page: US Congress CD13 & Florida Governor



CD13 Re-vote Page

Appendix B Technical Analysis of the PEB Virus Threat

1 Creating an Attack Scenario

We are not aware of any plausible scenario under which an outsider could introduce a virus in the days before a general election and cause it to spread rapidly enough to infect many or most of the machines before the end of election day. Consequently, an attacker without special inside access would have to introduce the virus months in advance if they wanted to influence some particular race.

1.1 Introducing the Virus

An attacker might be able to inject the virus into a single machine by breaking into a polling place where the machines are stored unattended before election day. Or, the attacker could volunteer as a poll worker and inject the virus during a quiet lull on election day. Injecting the virus into a single machine could take only seconds, if the attacker is highly sophisticated and prepared in advance, and would not necessarily require any kind of suspicious-looking activity.

The virus spread rate depends upon many variables, on how the virus is programmed, the details of the operational processes used by county election workers, how the machines are used, and other details that one would not expect to have any effect on system security. We cannot confidently estimate how rapidly or slowly such a virus would spread without additional detail about Sarasota election management procedures. The full range of possibilities is analyzed later. The virus might propagate from election to election, taking half a dozen or so elections before the majority of machines are infected. In this scenario, a virus introduced at one point by an outsider might not have the capacity to cause large-scale influence until years after it was introduced.

Alternatively, a virus introduced by an outsider in one election might spread to all of the machines before the next election. For instance, in this scenario, a virus introduced by an outsider during the primary election might propagate rapidly enough to infect all of the machines used in the subsequent general election.

We call an iVotronic machine infected if the virus is resident in that machine's firmware. A PEB contains non-volatile storage, which is used to store the election definition file and other data. An infected machine can overwrite the election definition file with maliciously chosen data. If that happens, we say that the PEB has been infected. Due to a flaw in the iVotronic code, when the iVotronic reads the election definition file from a corrupted PEB, the iVotronic machine may become infected. If so, the virus could take up residence in the iVotronic firmware, replace the running code of the machine, and remain resident there.

The specific vulnerability is that the iVotronic software copies a variable-length nul-terminated (C-style) string from the ballot definition file into a fixed-size stack-allocated buffer. If the string in the ballot definition is too long, this will overflow the bounds of the fixed-size buffer and overwrite other parts of memory. An attacker could use well-known techniques to exploit this bug, inject malicious code into the address space of the iVotronic machine, and cause the processor to begin executing that malicious code. At this point, the attacker has complete control over the iVotronic: the iVotronic is infected.

We found numerous instances of this type of bug. Misplaced trust in the election definition file can be found throughout the iVotronic software. We found a number of buffer overruns of this type. The software also contains array out-of-bounds errors, integer overflow vulnerabilities, and other security holes. They all arise due to the fundamental architectural flaw of misplaced trust.

A security expert might call this a failure of input validation. Standard advice in computer security is to "validate" all inputs, i.e., to check that their values fall within expected ranges and satisfy the relationships one expects, without making any assumptions that these conditions will necessarily hold until they have been explicitly checked. The architectural flaw is that the needed input validation is missing from the iVotronic software.

Finding and exploiting this vulnerability would require technical sophistication and dedication. We found these vulnerabilities by inspecting the source code. With more effort, an attacker may find these vulnerabilities without access to source code. The biggest barrier is that a would-be attacker would need access to an iVotronic machine for experimentation. Given this, a technically competent attacker may be able to, with sufficient time and motivation, discover these vulnerabilities. An attacker with the patience to reverse-engineer and disassemble the firmware would probably discover these flaws, but simpler methods would probably also suffice to reveal the vulnerability. For instance, applying a fuzzing tool to an existing election definition file would likely reveal the existence of stack-based buffer overruns, and at that point standard exploit methods might suffice.

1.2 Developing the Virus

Once the details of the vulnerability are known to the attacker, developing an attack seems likely to be straightforward if tedious. Ultimately, our best guess is that discovering this attack would be a matter of technical competence, tedium, and hard work, and it would require considerable motivation, but it would not require genius-level skills. A highly motivated and skilled lone individual could probably do everything needed to exploit the vulnerability. Consequently, the threat cannot be ignored.

Once the attacker has control of a machine, they would still need to develop a virus that automatically spreads from machine to machine. This virus could work by writing the exploit code onto every PEB that is inserted into an infected machine. Developing a working virus would require further work, but is likely within reach of a technically skilled programmer.

2 A Hypothetical Scenario: A Day in the Life of a Virus

To pull the pieces together, we illustrate one example scenario of how a virus might work by identifying what might happen in a step-by-step fashion:

1. The attacker obtains a voting machine for testing and a PEB. Of course these are controlled items and possessing them places the attacker at risk of discovery and prosecution. Stealing these items would be risky and illegal. Nonetheless, if the attacker can obtain these items, she could use these to develop malicious data and malicious code that, if placed on a PEB, can exploit the vulnerabilities and replace the running code of machines that use the PEB.
2. The attacker volunteers to serve as a poll worker. In many jurisdictions, the need for poll workers is so great that it is easy to become a poll worker simply by volunteering far enough in advance. In the worst case, the attacker might be installed as the chief poll worker in a polling place.
3. The attacker prepares an infected PEB. For instance, if PEBs are provided to chief poll workers before election day, then the attacker might take the master PEB and "infect" it by writing the malicious data and code that he prepared earlier onto the PEB. At this point, the PEB is "infected." In a worst case scenario, if the attacker is able to use this PEB to open the iVotronic machines, then all of the machines in that polling place are infected. The attacker's job is now done; the virus will spread without any further help from her.

4. At this point, the virus has been introduced into circulation within this one polling place. If only one machine is infected, it can infect any PEB that is inserted into it by writing the same malicious data and code onto that PEB.
5. By the end of the day, many of the PEBs in that polling place may have become infected. Also, at least one machine in that polling place is infected, and possibly all of them.
6. At the end of the election, the machines and PEBs are returned to election headquarters and are later returned to storage.
7. During normal procedures before the next election, the PEBs are cleared, which erases the viral content on them and returns them to an uninfected state. The terminals are also cleared, but a sophisticated attacker can write the virus to prevent the clear operation from removing or detecting the virus.
8. The machines are reassigned to polling places and distributed before the next election. Some of those machines may still be infected, so some polling places in the new election will receive infected machines.
9. When opening the polls, if one of the machines in the polling place is infected, it can infect the PEB used to open it and all other machines subsequently opened with that PEB will become infected. For each polling place that started out with one infected machine, we can expect the virus to infect about half of the machines by the end of the day, on average.
10. At the end of the day, we now have a larger population of infected machines. These machines are returned to the elections warehouse and then reused in another election. Because the number of infected machines only increases and never decreases, the infected population will grow over time. Under worst case conditions, we can expect exponential growth in the number of infected machines.
11. After some number of elections, most of the machines in the population are infected.
12. Up until now, the virus might have done nothing other than spread. At some point, the virus's payload might be activated (e.g., if it is triggered to activate after a certain date). At that point, all infected machines are controlled by the attacker, and will behave as directed by the attacker program. For example, the virus may flip votes for a selected party in selected contests or may change votes into undervotes. At some pre-determined date, or after it has accomplished its goal, if its creator has programmed it to do so, the virus might self-destruct, erase all indications of its presence, and return all iVotronic machines and PEBs to their factory state.

This is one example scenario. Many variations are possible, each with their own strengths and weaknesses. There are many ways to introduce the virus initially. Also, as we shall see below, there are other ways that viruses might spread.

2.1 Hypothetical Propagation Speed

One obvious question about such a scenario is: How fast can such a virus spread?

2.1.1 Bottom-up Propagation Speed and Limits

Since propagation is only accomplished via shared media, media sharing restrictions can control propagation from the bottom up (i.e. PEB to terminal to PEB). As we noted earlier viruses are highly unlikely to spread across county lines because counties generally do not share media and equipment. If no media are routinely shared, the virus could only propagate via policy violation, human error, or illegal activity. Moreover, voting is a safe operation, so inserting a PEB into a terminal to initiate a voting session cannot infect the terminal.

Thus, the media sharing policies within counties control the propagation potential. We constructed a simulation to evaluate the propagation speed in a hypothetical environment where machines and PEBs are randomly distributed. This simulation suggested that the number of infected machines grows exponentially with the number of elections and, under these assumptions, the virus would spread to infect most machines within about five elections. However, as discussed in Section 7.5.3, these randomness assumptions do not reflect the practices in place in Sarasota County, so we reference this result only as a baseline, pessimistic scenario. Even in this case, comprehensive propagation would take three election cycles on average so an attacker who wanted to infect most of the iVotronic machines by injecting a virus into a single machine would need to prepare the virus and introduce it into circulation several election cycles in advance.

This also shows how implementing a process that consistently places terminals and PEBs in the same precinct in every election can prevent virus propagation. Again, this leverages the decentralized nature of election management to enhance security protections.

2.1.2 Supervisor Terminal

If a Supervisor terminal were infected, the process of preparing PEBs for the next election could cause every prepared PEB to become infected. The subsequent poll-opening process at each polling place with these infected PEBs would cause all of the voting terminals to become infected. This means that the Supervisor terminal is a single, central point of vulnerability.

There are three primary threats to the supervisor terminal. The first is an insider attack, which is straightforward. The second is compromise via illegal activity, essentially where an intruder breaks into the office holding the Supervisor terminal.

The third threat relates to how PEBs are handled after they are returned from the polling place. If an attacker can infect one or more PEBs after the election and return them to circulation, they may be able to infect the Supervisor terminal during preparations for the next election.

The detailed process used to handle such PEBs in Sarasota County would have a major impact on how quickly a virus would spread. For instance, if Sarasota County workers ordinarily insert every PEB into the Supervisor terminal and invoke the Clear Supervisor PEB Vote Totals operation (from the Election Central Applications menu) after the election is over, then the risk of a virus is pronounced: an attacker would just have to introduce a single infected PEB to infect the Supervisor terminal. As another example, if Sarasota County workers ordinarily perform the Qualify PEB(s) operation (from the Supervisor terminal's Service menu) on every PEB before performing any other operation on that PEB, then the risk would be significantly reduced, since this operation clears the contents of the PEB before it has a chance to infect the Supervisor terminal.

3 Ineffective Defenses Against the Virus Threat

We examined many security features of the system to see if they would be able to ward off viruses. Our analysis is as follows:

Proprietary file formats are not an effective defense against viruses. The election definition file, as stored on the PEB, is in proprietary format. This format includes several version fields, magic constants, and other values that must be correct, or else the iVotronic machine will reject the election definition file as invalid. However, this will not prevent the spread of a virus. First, it would not be difficult for a sophisticated attacker with access to an iVotronic machine to reverse-engineer these constraints. Second, these constants and version fields are the same for every iVotronic machine across the country, so they cannot be treated as cryptographically secure secrets. Third, the part of the file where the virus would be inserted does not contain any of these magic constants or

version fields. Therefore, all an attacker would need to do is to take an existing election definition file and overwrite only the portion needed to hold the virus.

- Checksums cannot prevent viruses. The iVotronic election definition file format contains an unkeyed 8-bit checksum (the sum of the bytes modulo 256). This checksum is a reasonable way to detect random errors (e.g., hardware bit flips), but it is not an effective defense against malicious activity. This kind of unkeyed checksum does not prevent malicious tampering with the contents of the election definition file while in transit, because an attacker can arrange for his change to leave the checksum field valid, or can modify the election definition file and then overwrite the checksum with a correct checksum value. The PEB also uses a CRC16 checksum to check for random errors in stored data, but this will not detect or deter malicious attacks for the same reason.
- The Election Qualification Code (EQC) does not prevent viruses. The EQC is a 32-bit election-specific secret code that must be present on a PEB; otherwise, the PEB will be rejected by the iVotronic machine. (See Appendix D for more details.) Unfortunately, this does not prevent the spread of viruses. The EQC is the same for all iVotronic machines and all PEBs in a county, for any one election. As long as the virus takes care to leave the EQC field in the PEB undisturbed, the EQC will not limit virus propagation. Also, the EQC will not prevent virus introduction. The EQC is stored in the clear, not cryptographically protected on a PEB, so a malicious poll worker who gains unsupervised access to a PEB before the end of an election could overwrite the data on the PEB, leaving the EQC undisturbed, and re-introduce it into circulation before the end of the election.

We conclude that though these mechanisms may deter or complicate an attack, they would not pose an effective defense against viruses. This is not surprising, as security is not their designed purpose. It does not indicate a flaw in those mechanisms; it is well known that mechanisms intended to improve reliability and detect random errors generally are not sufficient to prevent malicious attack. We emphasize that we do not allege that the checksums or file formats or EQC mechanisms are flawed in any way, merely that they do not serve as an effective barrier to viruses.

Appendix C Safe and Unsafe Operations

We note that the mere act of inserting an infected PEB into an iVotronic will not infect the machine. Infection can spread only if one invokes vulnerable operations while a PEB is inserted. We analyzed nearly all available operations and reflect the results in the tables below. If performing an operation while an infected PEB is inserted can cause an iVotronic machine infection, then we call that operation *unsafe*. If performing that operation cannot infect the iVotronic even in the presence of an infected PEB, we call that operation *safe*. In some cases we were not able to identify from the code whether the operation is safe or not; in that case, we labeled it as *unknown*. We assume that the iVotronic machine is initially uninfected and ask only whether invoking that operation can cause the iVotronic to become newly infected.

We note that the results for Voter terminals (ordinary iVotronic machines, typically used for voting) differ for Supervisor terminals. Also, in some cases the results vary according to the machine mode. We also distinguish between operations that are ordinarily performed by poll workers under normal operation (e.g., opening or closing the polls), operations that can only be invoked via the Service menu (which is only accessible using a special password), and operations that can only be invoked via the Elections Central Administration menu (which requires yet another special password). The latter two menus are normally only used by county election workers or technicians; they are not normally accessible to poll workers or voters. Our analysis for the latter two categories are presented in separate tables.

Lastly, in some cases the results depend upon whether a Supervisor or Voter PEB is inserted into the machine. Because Sarasota County uses Pollworker-activated Mode, which only uses Supervisor PEBs, we did not analyze any of the code that was associated with Voter PEBs.

The “Qualify PEB(s)” operation (accessible via the Service menu) deserves special comment. This operation clears PEB contents and erases any data previously stored on it. Therefore, not only is this operation safe to perform on an infected PEB, it also cleans infected PEBs.

There is a subtlety associated with “Qualify PEB(s).” Suppose that we have a PEB whose firmware (software that operates infrared communications) has been replaced by the attacker. The “Qualify PEB(s)” operation sends commands to the PEB instructing the PEB to erase all of its data and leaves it up to the PEB to do so. If the PEB’s firmware has been replaced by malicious code, then the PEB might ignore these instructions to erase itself. In short, if the attacker has had the chance to physically tamper with the PEB, then we cannot rely upon “Qualify PEB(s)” to erase and disinfect the PEB. On the other hand, for PEBs that have not been under the physical control of the attacker and that contain only malicious data—not malicious firmware—“Qualify PEB(s)” will indeed erase all malicious data present. We did not analyze whether there was any way for a malicious iVotronic machine to attack or corrupt the firmware or code on the PEB, as this was outside the scope of our analysis.

We note that while these virus vulnerabilities are dangerous, the number of unsafe operations is a bit misleading relative to the actual threat that they pose. Many of these operations are rarely performed so are unlikely to infect a large number of PEBs. Moreover, while there are many unsafe operations, each may require a distinct exploit and it may not be possible to exploit more than one operation with a single PEB. It may also be true that preparing a PEB for exploit may corrupt it for normal operation, thus exposing it to detection or surreptitious removal from service.

1 Ordinary operation, Voter terminal, Supervisor PEB

This describes the ordinary functions of a Voter terminal when used in Poll worker-activated mode. . (Voter-activated mode was not analyzed.)

Operation (mode)	**Safe/unsafe to perform with an untrusted PEB inserted.**
Opening the polls (BLANK)	*unsafe*
Voting (OPEN)	safe
Closing the polls	*unknown*
Printing reports, modem vote results on a-closed terminal (CLOSED, EMERGENCYCLOSED)	*unsafe*
everything else	*unknown* (not analyzed)

2 Ordinary operation, Supervisor terminal, Supervisor PEB

This describes the ordinary functions of a Supervisor terminal when used with Supervisor PEBs. (Operation with Voter PEBs, i.e., Voter-activated mode, was not analyzed.)

Operation (mode)	**Safe/unsafe to perform with an untrusted PEB inserted.**
Prepare Voter PEB	not analyzed (only used for Voter-activated mode)
Opening the terminal for voting (BLANK, LOADED)	*unsafe*
Closing the terminal (OPEN)	*unsafe* if performed before the designated time for closing the polls
Printing a late zero tape (OPEN)	*unsafe*
Printing reports, modem vote results on a-closed terminal (CLOSED, EMERGENCYCLOSED)	*unsafe*
Unlocking a locked terminal (LOCKED)	safe
everything else	not analyzed

3 Service menu

The following comments apply to both Voter and Supervisor terminals, except where noted.

Operation	Safe/unsafe to perform on an untrusted PEB?
Clear And Test Terminal	safe
Set Time and Date	safe
Qualify PEB(s)	safe
Upload PEB to Compact Flash	*unknown*
Upload 3 Flash Memories to Compact Flash	safe
Test Printer	safe
Test Modem	*unsafe* in every mode
Upload Firmware	safe
Load System Files (Text Ballots)	safe
Enable Audio ballot on Unit	safe
Set Volume	safe
Force Coded Ballot Entry	safe
VOTE Button Configuration	safe
Enable Receipt Printing	safe
Select Progress Bar	safe
Logic And Accuracy Test	*unsafe*, if L&A testing is enabled (i.e., mode isn't OPEN or CLOSING and public count is zero), depending upon which option the user subsequently selects; see Logic and Accuracy Test menu below for details and full analysis
Enable Zoom Selection Screen	safe

3.1 Elections Central Applications menu, for Voter terminals

Operation	Safe/unsafe to perform on an untrusted PEB?
Upload Terminal Audit Data Serial	*Unknown*
Upload Terminal Audit Data to CompactFlash	*Unknown*
Print Report to Screen	*unsafe* if polls have not yet been opened (i.e., BLANK or OPENING mode); safe otherwise
Print Report To The Printer	*unsafe* if polls have not yet been opened (i.e., BLANK or OPENING mode); safe otherwise
Print Event Log	not analyzed
Print Vote Summary With Write-Ins	safe
Print Vote Summary Minus Write-Ins	safe

3.2 Logic and Accuracy Tests menu

This menu provides several options for L&A testing. The following comments apply to both Voter and Supervisor terminals, except where noted otherwise.

Operation	**Safe/unsafe to perform on an untrusted PEB?**
Vote For One Test	*unsafe* in every mode
Multi Vote Test	*unsafe* in every mode
Vote Selected Ballot Test	*unsafe* in every mode
Print L And A Vote Totals to Screen	*unsafe* if polls have not yet been opened (i.e., BLANK or OPENING mode), for Voter terminals; *unsafe* in every mode, for Supervisor terminals; otherwise, *unknown*
Print L And A Vote Totals to Printer	*unsafe* if polls have not yet been opened (i.e., BLANK or OPENING mode), for Voter terminals; *unsafe* in every mode, for Supervisor terminals; otherwise, *unknown*
Transfer Results To PEB	*unsafe* if polls have not yet been opened (i.e., BLANK or OPENING mode), for Voter terminals; *unsafe* in every mode, for Supervisor terminals; otherwise, *unknown*
Clear And Test Terminal	safe

3.3 Elections Central Applications menu, for Supervisor terminals

Operation	**Safe on an untrusted PEB?**
Prepare PEB for Polling Location	Safe
Test Vote	Safe
Clear Supervisor PEB Vote Totals	*unsafe*
Prepare PEB for Serial Audit	Safe
Prepare PEB for CompactFlash Audit	Safe
Prepare PEB for Clear And Test	safe
Upload PEB Vote Results	*unsafe*
Print Report To Screen	*unsafe*
Print Report To The Printer	*unsafe*
Start Election Qualification Trail	safe
Color Option Numbers	safe
Print Event Log	safe
Print Vote Summary With Write-Ins	*unsafe*
Print Vote Summary Minus Write-Ins	*unsafe*

Appendix D Passwords

We analyzed the access control mechanisms in the iVotronic software to determine whether they ensure that only authorized users are able to invoke sensitive functions on the machines. The iVotronic uses password protection to control access to sensitive functions. Therefore, we analyzed all uses of passwords in the iVotronic.

We found several passwords, used for different purposes:

- The Service Menu password is used to control access to the Service Menu, which provides functions that would ordinarily only be needed in the county warehouse. The Service Menu is not normally used by poll workers.
- The ECA password controls access to the Elections Central Administration menu. This menu provides additional functionality over and beyond the Service menu. The ECA menu is only accessible from the Service menu; therefore, reaching the ECA menu requires knowledge of both the Service password and the ECA password.
- The Clear and Test password is used to control access to the clear and test operation. The clear and test operation erases all votes stored on the iVotronic machine and prepares it for use in the next election. Because this operation can irreversibly delete votes, this is a sensitive function that must be protected from unauthorized individuals.
- The Election Qualification password is used to prepare a machine for a new election.
- The Upload Firmware password is used to control the ability to upgrade the executable software resident on the iVotronic's internal flash memory. This is an extremely sensitive operation, because it allows replacing the iVotronic's software. If this were invoked by a malicious individual, they could use it to install malicious software on the iVotronic machine or to infect it with a virus. This operation is available as a menu option in the Service menu. Therefore, invoking this operation requires knowledge of both the Service password and the Upload Firmware password.
- The Override password is used to control certain exceptional conditions that should not normally arise. For instance, if the user tries to close the polls on an iVotronic machine before the official time when the election is due to end, the machine requires the user to enter an override password before proceeding.
- The modem password is used by the iVotronic machine to transmit results back to the Unity Data Acquisition Manager (DAM) system at the county headquarters. When the iVotronic machine connects to the Unity server over the telephone, it first sends the modem password over the phone. While we do not have access to the Unity server source code to check how the Unity server uses this password, it would be logical to presume that the Unity server checks that the proper password has been sent before allowing the connection to continue. The modem password does not need to be known by any human.

Typically, the override password would be the only password divulged to poll workers; the other passwords would not be revealed to poll workers, and would be told only to county election workers.

Next, we analyze password security strength to determine if they can be guessed by an ill-intentioned individual. The modem password can be set at the Unity server when the election is configured. It is included in the election definition file. It is listed in the clear in the election definition file found on every PEB and, eventually, on every iVotronic machine. It is the same for

all iVotronic machines within a county. If it is not set, there is a default value hard-coded into the source code; this default is the same for all iVotronic machines across the nation. It is up to election officials to choose this password in a way that ensures it is unguessable, to change this password frequently (e.g., after every election), and to control who knows the password. Those are operational questions that are beyond the scope of a source code review.

Like the modem password, the override password can also be set at the Unity server when the election is configured. It too is included in the clear in the election definition file found on every PEB, and it is the same for all iVotronic machines within a county. It is selected and managed by election officials, so the management of this password is beyond the scope of a source code review.

Each of the other passwords mentioned above is fixed and hard-coded into the source code. They are the same for all iVotronic machines in the country, and likely to be known to every election official who manages elections on an iVotronic machine. They can never be changed, without changing the firmware on the iVotronic machine. This represents poor practice.

The Service Menu password, Clear and Test password, ECA password, and Upload Firmware password are three-letter case-insensitive passwords. Each one is chosen to be mnemonic and easy to remember. The problem is they are also likely to be fairly easy to guess. They follow a memorable pattern. Someone who knows one of these passwords can probably guess what the other ones are without too much difficulty. These passwords provide very little security.

The Election Qualification password is a five-letter case-insensitive password that is chosen to be easily memorable. It does not follow the same pattern as the other passwords.

The weakness of the Upload Firmware and Service passwords are of primary concern, because someone who knows those two passwords can replace the software on the iVotronic with malicious software that switches votes from one candidate to another, that turns valid votes into undervotes or deletes them entirely, that infects the machine with a virus, or that otherwise compromises the integrity of the election. These functions should be better protected.

Our judgment is that the password mechanisms on the iVotronic are poorly conceived and poorly implemented. The consequence is that the passwords by themselves do not do a good job of preventing unauthorized individuals from accessing critical system functions.

Finally, these passwords can all be bypassed using a special type of PEB, called a Factory Test PEB. When a PEB is inserted, the iVotronic machine queries the PEB to ask it what kind of PEB it is, and the PEB returns a single byte indicating what type of PEB it is. A Factory Test PEB identifies itself by returning a special single-byte value. This special value is hard-coded into the iVotronic code. Anyone who knows the special single-byte value, has access to a PEB and is able to program the PEB could construct a PEB that identifies itself as a Factory Test PEB. When a Factory Test PEB is present, all password checks are bypassed: in places where the user would normally need to enter a password, the password check is bypassed, the machine functions as though the correct password had been entered, and a log entry is appended to the event log as though the user entered the correct password. This undocumented backdoor poses a risk of unauthorized access to critical system functions, because it provides a way that a malicious individual could bypass the password checks by tampering with a PEB.