2007-CDOS-ESS-001-0403

ELECTION SYSTEMS & SOFTWARE

PROJECT OVERVIEW

COPY

2007-CDOS-ESS-001-0403
PROJECT OVERVIEW BINDER “A.3”

TABLE OF CONTENTS

2007-CDOS-ESS-001-0403
PROJECT OVERVIEW BINDER "A.3"

TABLE OF CONTENTS

1. INTRODUCTION
    a. Introduction statement
    b. Detailed test summary

2. COMPONENTS
    a. Components of the ESS voting system package

3. RECOMMENDATION
    a. Recommendation overview
    b. Voting system application recommendation
    c. Bar charts of residual failures

4. RESTRICTIONS
    a. Restrictions for use of the voting system

5. CONDITIONS
    a. Conditions for use of the voting system

6. COMMENTS

7. AUDIT REPORTS
    a. Testing Board response to Audit report
    b. Audit Report
    c. Associated correspondence – Located in Binder "A", Sec. 7

8. ADDITIONAL CORRESPONDENCE – Located in Binders "B - D"

2007-CDOS-ESS-001-0403
PROJECT OVERVIEW BINDER “A.3”

INTRODUCTION

STATE OF COLORADO
Department of State

1700 Broadway, Suite 270
Denver, CO 80290

Mike Coffman
Secretary of State

Holly Lowder
Director, Elections

Why an amendment to the A.2 binder/report?

During discussions with the Secretary of State, Legislators, the Vendor, and the County Clerks, there were many tests that were unable to be completed during the initial round of testing within the original deadline for the application of this system. The Secretary of State requested that the testing board conduct the additional testing by coordinating with the vendor to receive valid programming and the ballots for the devices to test. This updated documentation displays the updated successful conclusion of that testing.

In addition, the testing board conducted additional internal evaluations of the iVotronic PEB security issue which is reflected now in the conditions and comments section of this document to include a mitigation strategy for the PEB port given that disabling the device is unavailable as an option.

The remainder of the sections of the binder have been updated to reflect these changes which include the recommendation section, the restriction section and the conditions section.

Introduction

On April 3, 2007, Election Systems and Software (ESS) approached the Colorado Secretary of State's Office with an application to certify a voting system. The application was accepted by the Voting Systems Certification Program Testing Board (Testing Board). The system was assigned certification number: 2007-CDOS-ESS-001-0403.

The voting system is known by its federal certification name as "Unity 3.0.1.1" Federal certification is to the 2002VSS standards, and was obtained on August 31st, 2006 (NASED#: N-2-02-22-22-007).

The testing board proceeded to evaluate the ESS voting system during the time period of April 3rd – December 1st. All findings are documented within the binders A.2 – 31, as well as addendum binders : 13.1, 14.1, 16.1, 18.1, 19.1, 21.1, 22.1, 25.2, 27.1.

The Project Overview Binders (Binders "A.2 – D") provides an overview of the findings of the project, and the following additional information:

- Introduction
- System Components
- Recommendation to the Secretary of State
- Restrictions on the use of the voting system suggested by the Testing Board
- Conditions to the Recommendation suggested by the Testing Board
- Additional Comments by the Testing Board
- Independent Audit Reports
- Miscellaneous Correspondence of importance

During the process of certifying the system, the Testing Board adhered to the procedures outlined by the Voting System Program procedures document. The certification process was led by Jerome Lovato, with Tim Bishop and Michael Chadwell providing the primary cross evaluation. Additional cross check and documentation verification was conducted by Danny Casias with coordination by the Program Manager – John Gardner with assistance from Michael Chadwell as necessary.

The testing board evaluated the voting system in accordance with the requirements set forth in Secretary of State Rule 45, as well as applicable elements contained within the laws of the Help America Vote Act, Colorado Revised Statute, multiple sections of Title 1, and Secretary of State Rules as appropriate. All testing results and output which includes extensive video documentation of the evaluation process have been archived and well preserved in accordance with the Voting Systems Program procedures document.

Through the evaluation, the testing board identified a variety of deficiencies within the system which include functionality, security, auditability and documentation requirements. The following sections will address these deficiencies as either a restriction for use (preventing recommendation by the testing board), or a condition for use (allowing the system to be recommended provided conditional elements are adhered to). Restrictions are identified in a one-to-one value. One identified restriction = one failure on the Detail Test Summary. Conditional elements represent a one-to-many value. The execution of a single condition placed on the use of the system in many cases will address multiple failures as the testing board often experienced failures that exhibited a "daisy chain" effect. One high level failure would trigger many follow up test scenarios. Refer to the comments section of this binder for additional comments on this topic.

STATE OF COLORADO
Department of State

1700 Broadway, Suite 270
Denver, CO 80290

Mike Coffman
Secretary of State

Holly Lowder
Director, Elections

Detailed Test Summary - Ammended

The testing board executed the testing process for the ES&S Voting System in the manner prescribed by Rule 45, and the detailed procedures document provided on the Voting Systems Certification Program website (http://www.elections.colorado.gov/DDefault.aspx?tid=501).

The outcome of the process involved over 700 functional test evaluations, 3500 detailed line items for document review, and over 90 supplemental tests comprising the sections for application review, demonstration and work on completing the trusted build. The documentation comprised of this test work is evident in over 50 binders generated by the testing board, a multitude of boxes containing evidence generated from devices, ballots, reports, and other findings. In addition to this evidence, over 200 DVD records exist documenting the process of the testing board.

Below is the summary report of test status generated by the testing board regarding the ES&S Voting System evaluation:

ESS						
	# Requirements	# Passed	# Failed	Binder Status		% Passed
Phase I - Application	22	20	2	signed		90.91%
Phase II - Doc. Review	3524	3171	271	signed		89.98%
Phase III - Demo	58	58	0	signed		100.00%
Phase III - Trusted Build	25	12	13	signed		48.00%
Phase III - Functional Test (overall)	699	466	233		100.00%	66.67%
Security	139	88	51			63.31%
System Process	340	232	108			68.24%
Election (pre, ED and post)	220	146	74			66.36%
Independent Audit	1674	1674	Review of Test Board work			100.00%
Phase IV - Certification Doc.	n/a	n/a		n/a		
Phase V - Qualification Report	n/a	n/a		n/a		

Requirements Status for Colorado Certification of ES&S Voting System										
Section Category	Binder #	Category	Sec	**Total # ofTests to complete**	Status:	DRE	PCOS	CCOS	EMS	**Remaining to complete**
Section "A" - Pre Testing	1	Application	aa	**22**	Pass	n/a	n/a	n/a	19	
					Pass Conditional	n/a	n/a	n/a	1	
					Suspend	n/a	n/a	n/a		
					Fail	n/a	n/a	n/a	2	
					Not applicable	n/a	n/a	n/a		**0**
	2-6	Documentation Review	ab	**3524**	Pass	414	348	293	281	
					Pass Conditional					
					Suspend					
					Fail	91	93	87	82	
					Not applicable	376	440	501	518	**0**
	7	Demonstration	ac	**54**	Pass	14	13	13	14	
					Pass Conditional					
					Suspend					
					Fail					
					Not applicable					**0**
	8	Trusted Build	ad	**20**	Pass	2	2	2	2	
					Pass Conditional					
					Suspend					
					Fail	2	2	2	2	
					Not applicable	1	1	1	1	**0**
	9-12	Source Code Review	ae	**0**	Not applicable	n/a	n/a	n/a	n/a	**0**

Requirements Status for Colorado Certification of ES&S Voting System										
Section Category	Binder #	Category	Sec	**Total # ofTests to complete**	Status:	DRE	PCOS	CCOS	EMS	**Remaining to complete**
Section "B" - Security Testing	13	System Access	ba	**36**	Pass	4	2		2	
					Pass Conditional	1				
					Suspend					
					Fail	1	4	5	13	
					Not applicable	1	1	1	1	**0**
	14	Operating System Security	bb	**20**	Pass				2	
					Pass Conditional					
					Suspend					
					Fail					
					Not applicable	9			9	**0**
	15	Database Security	bc	**24**	Not applicable	6	6	6	6	**0**
	16	Removable Media	bd	**13**	Pass	1	1		1	
					Pass Conditional			1		
					Suspend					
					Fail				1	
					Not applicable	2	2	2	2	**0**
	17	Networking and Telecommunications	be	**46**	Pass	1	2	2	3	
					Pass Conditional		1	2		
					Suspend					
					Fail	9	7	4	7	
					Not applicable	2	2	3	1	**0**

Requirements Status for Colorado Certification of ES&S Voting System										
Section Category	Binder #	Category	Sec	**Total # ofTests to complete**	Status:	DRE	PCOS	CCOS	EMS	**Remaining to complete**
Section "C" - System Testing	18	System	ca	**47**	Pass	7	2		7	
					Pass Conditional					
					Suspend					
					Fail	2	6		21	
					Not applicable				2	**0**
	18	System (central count)	cb	**11**	Pass	n/a	n/a	2	n/a	
					Pass Conditional	n/a	n/a		n/a	
					Suspend	n/a	n/a		n/a	
					Fail	n/a	n/a	9	n/a	
					Not applicable	n/a	n/a		n/a	**0**
	19-20	Ballot Process	cc	**153 ***	Pass	20	32	28	29	
					Pass Conditional	1				
					Suspend					
					Fail	3	11	6	1	
					Not applicable	1	3	12	6	**0**
	21	Performance	cd	**24**	Pass	1	1	1	5	
					Pass Conditional					
					Suspend					
					Fail	7	4	4	1	
					Not applicable					**0**
	21	DRE Processing	ce	**24**	Pass	17	n/a	n/a	n/a	
					Pass Conditional		n/a	n/a	n/a	
					Suspend		n/a	n/a	n/a	
					Fail	1	n/a	n/a	n/a	
					Not applicable	6	n/a	n/a	n/a	**0**
	22	Audits	cf	**29**	Pass	7	1	2	3	
					Pass Conditional					
					Suspend					
					Fail		6	5	5	
					Not applicable					**0**
	22	Reports	cg	**52**	Pass	6	8	6	8	
					Pass Conditional					
					Suspend					
					Fail	3	5	6	2	
					Not applicable	4	1	1	2	**0**

Requirements Status for Colorado Certification of ES&S Voting System										
Section Category	Binder #	Category	Sec	Total # ofTests to complete	Status:	DRE	PCOS	CCOS	EMS	Remaining to complete
Section "D" - Election Day Tests	23	Hardware Diagnostics Testing	da	8	Pass	1				
					Pass Conditional					
					Suspend					
					Fail	1	1	1		
					Not applicable		1	1	2	0
	23	Voting	db	65 *	Pass	13	14	12	2	
					Pass Conditional					
					Suspend					
					Fail		6	9	4	
					Not applicable	4			1	0
	24	Multi-Page Ballots	dc	6	Pass					
					Pass Conditional					
					Suspend					
					Fail		2	1	1	
					Not applicable	2				0
	24	Multiple Languages	dd	4	Pass					
					Pass Conditional					
					Suspend					
					Fail	1	1	1	1	
					Not applicable					0
	24	Provisional	de	25	Pass	3	1	2		
					Pass Conditional					
					Suspend					
					Fail	1	4	3	5	
					Not applicable	3	1	1	1	0
	25	V-VPAT	df	28	Pass	20	n/a	n/a	n/a	
					Pass Conditional	2	n/a	n/a	n/a	
					Suspend		n/a	n/a	n/a	
					Fail	5	n/a	n/a	n/a	
					Not applicable	1	n/a	n/a	n/a	0
	25	Accessibility	dg	41	Pass	27	n/a	n/a	n/a	
					Pass Conditional		n/a	n/a	n/a	
					Suspend		n/a	n/a	n/a	
					Fail	13	n/a	n/a	n/a	
					Not applicable	1	n/a	n/a	n/a	0
	26	Closing Polls	dh	30	Pass	10	14	n/a	n/a	
					Pass Conditional			n/a	n/a	
					Suspend			n/a	n/a	
					Fail	5	1	n/a	n/a	
					Not applicable			n/a	n/a	0

Requirements Status for Colorado Certification of ES&S Voting System										
Section Category	Binder #	Category	Sec	Total # ofTests to complete	Status:	DRE	PCOS	CCOS	EMS	Remaining to complete
Section "E" - Post Election	27	Post Election Audit	ea	4	Pass	1				
					Pass Conditional					
					Suspend					
					Fail		1	1	1	
					Not applicable					0
	27	Recount	eb	8	Pass		1	2		
					Pass Conditional					
					Suspend					
					Fail	1	1	1	1	
					Not applicable	1				0
	27	Recount (central count)	ec	1	Pass	n/a	n/a	1	n/a	
					Pass Conditional	n/a	n/a		n/a	
					Suspend	n/a	n/a		n/a	
					Fail	n/a	n/a		n/a	
					Not applicable	n/a	n/a		n/a	0

2007-CDOS-ESS-001-0403
PROJECT OVERVIEW BINDER “A.3”

COMPONENTS

STATE OF COLORADO
Department of State

1700 Broadway, Suite 270
Denver, CO 80290

Mike Coffman
Secretary of State

Holly Lowder
Director, Elections

Components

As submitted on April 3, 2007, the following components comprise the requested voting system package from ESS:

Component Name	System Function	Version Number
Unity	Software Application – includes only the following modules: Audit Manager Election Data Manager ESS Image Manager iVotronic Image Manager Optech Image Manager Hardware Programming Manager Election Reporting Manager	3.0.1.1: 7.3.0.0 7.4.4.0 7.4.2.0 2.0.1.0 4.0.0.0 5.2.4.0 7.1.2.1
M100	Precinct Optical Scanner	5.2.1.0
M650	Central Count Optical Scanner	2.1.0.0 (Green Light Only)
iVotronic ADA w/ 3-button	Direct Record Electronic Device	9.1.6.2
iVotronic non-ADA	Direct Record Electronic Device	9.1.6.2

The Unity 3.0.1.1 system originally included modules for Data Acquisition Manager, Ballot on Demand, and software and hardware components for the Automark system. These components were requested to be removed from the voting system by ESS representatives. The request to remove Data Acquisition Manager and Ballot on Demand can found in the correspondence section of Project Overview Binder "B" dated October 30, 2007, and the request to remove the Automark system can found in the correspondence section of Project Overview Binder "C" dated November 8, 2007.

Photographs and additional details on each component can be found under test # AA6-P1-605.

2007-CDOS-ESS-001-0403
PROJECT OVERVIEW BINDER “A.3”

RECOMMENDATION

STATE OF COLORADO
Department of State

1700 Broadway, Suite 270
Denver, CO 80290

Mike Coffman
Secretary of State

Holly Lowder
Director, Elections

Recommendation Overview

The approach of the testing board regarding a recommendation is absolute. Any one item outstanding in the **Restrictions** section of the binder (no "conditional use" option discoverable by the testing board) will trigger a "N" value on the **Recommendation** table.

Therefore, for quick understanding of the overall outstanding deficiencies with the system, and to provide a summary of reasons for the "Y" or "N" value in the **Recommendation** table.

The following table provides a high level summary statement of findings by the testing board. These items constitute a summary of the findings in the ***Restrictions*** section of the project overview binder.

Component (details in the components section)	Recommended to be Certified?	Reason
Software (Unity)	No	• Failure to provide required State documentation
Precinct Scanner (M100)	No	• Failure to provide required State documentation • Failure to prevent and detect normal operator changes within system.
Central Count Scanner (M650)	No	• Failure to provide required State documentation • Failure to prevent and detect normal operator changes within system.
DRE (ivotronic)	No	• Failure to provide required State Documentation • Failure to provide privacy and protection of votes. • Failure to provide voters opportunity to review ballot and make changes prior to voting. • Failure to prevent and detect normal operator changes within system. • Paper Record not accessible to blind voters. • Failure to meet state requirements for Accessibility

STATE OF COLORADO
Department of State

1700 Broadway, Suite 270
Denver, CO 80290

Mike Coffman
Secretary of State

Holly Lowder
Director, Elections

ES&S Recommendation for voting system application:

2007-CDOS-ESS-001-0403

Binary Assessment (with conditionals)

Component	Version	Accuracy	Security	Accessibility	Compliance	Testing Board Recommendation
UNITY	3.0.1.1	Y	Y	N/A	N[1]	**N**
M650	2.1.0.0	Y	Y	N	N[1]	**N**
M100	5.2.1.0	Y	Y	N	N[1]	**N**
iVOTRONIC	9.1.6.2	Y	N	N	N[1]	**N**

[1] Missing/insufficient state documentation pursuant to Colorado Secretary of State Rule 45.

Definitions:

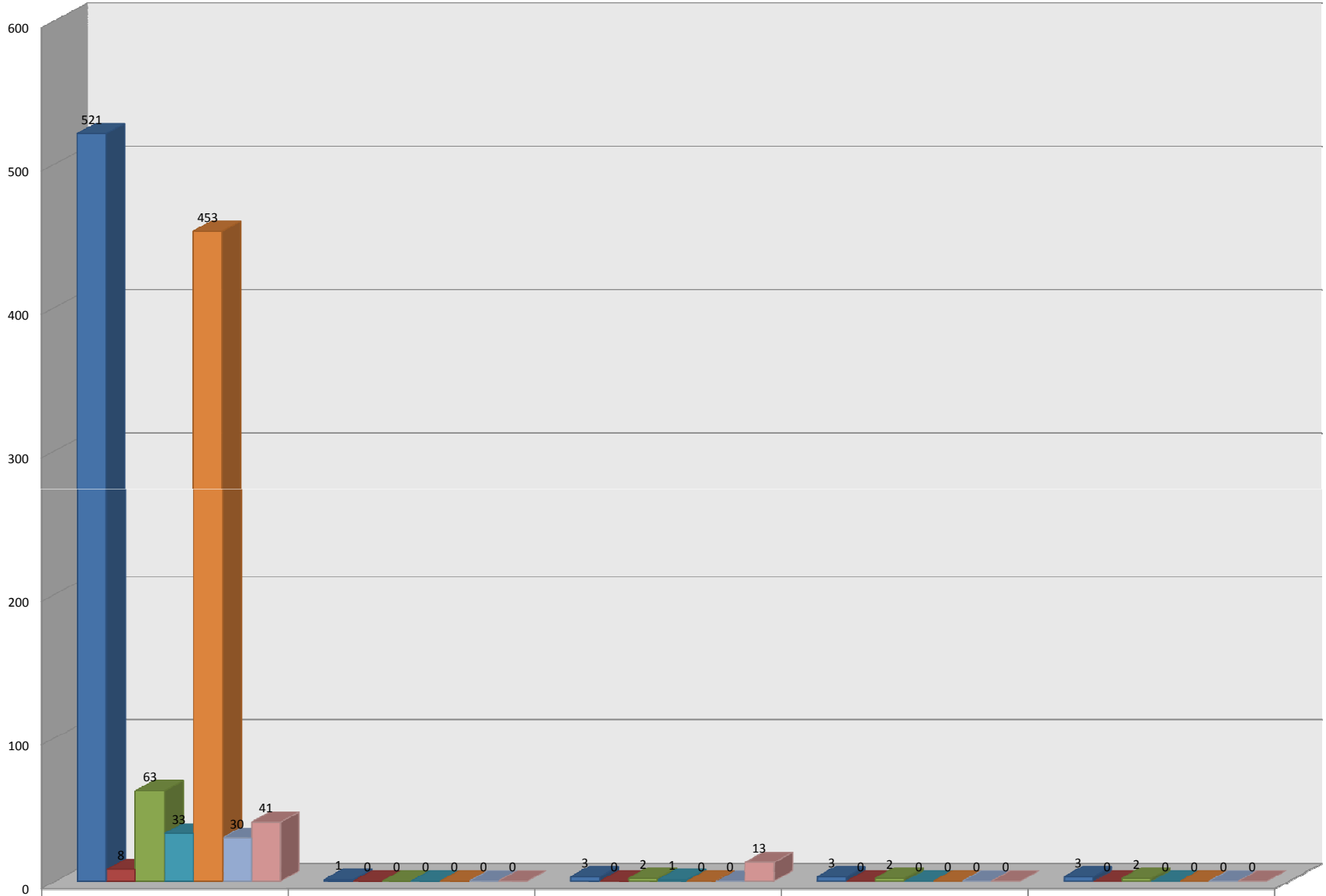
Accuracy – correctly reading, displaying, tabulating and reporting votes. (Functional, or Performance)

Security – vote data is protected and maintains integrity throughout system processing. (Audit, Security or Telecommunications)

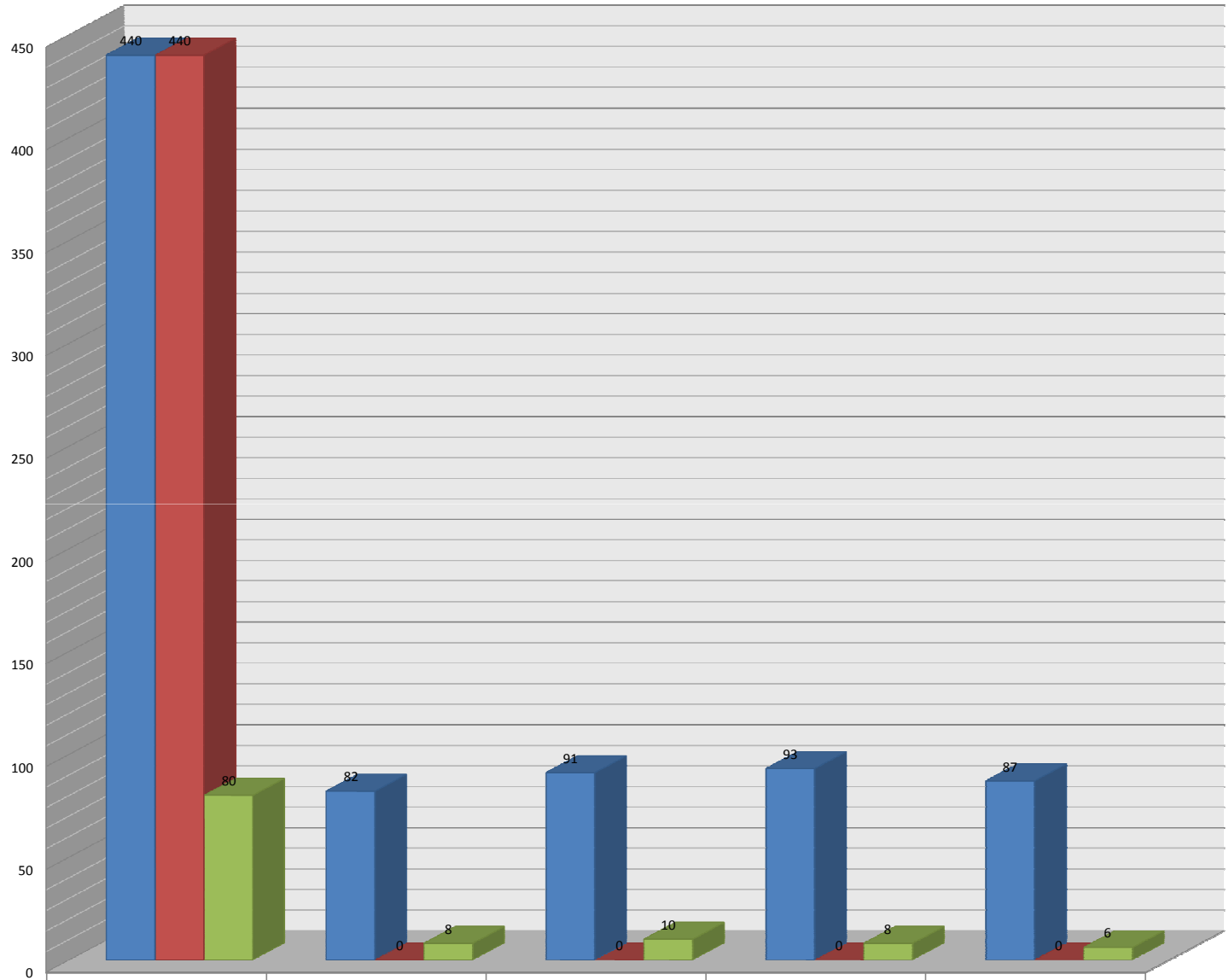
Accessibility – voter system have requisite usability and reliability. (Functional, Accessibility, or Physical Design)

Compliance – system conforms to federal and state requirements for certification and/or documentation. (Documentation)

ES&S Outstanding Functional Test Failures



	Total Tests	Total EMS Remaining Failures	Total iVotronic Remaining Failures	Total PCOS Remaingin Failures	Total CCOS Remaining Failures
Functional Requirements	521	1	3	3	3
Performance Levels	8	0	0	0	0
Physical design	63	0	2	2	2
Audit Capacity	33	0	1	0	0
Security	453	0	0	0	0
Telecommunications	30	0	0	0	0
Accessibility	41	0	13	0	0



	Total Tests**	Total EMS Remaining Failures	Total DRE iVotronic Remaining Failures	Total PCOS Remaingin Failures	Total CCOS Remaining Failures
Total Federal Tested incorrectly*	440	82	91	93	87
Total No Proof (documentation) of Test - Federal	440	0	0	0	0
Total No Proof (documentation) of Test - State	80	8	10	8	6

* Incorrectly tested means the ITA either reported that a required item was not tested, or a required item was tested incorreclty for the device type.

** Total tests has N/A items removed for chart scale.

Created by the Colorado Secretary of State's Office Voting systems Certification Program

2007-CDOS-ESS-001-0403
PROJECT OVERVIEW BINDER “A.3”

RESTRICTIONS

STATE OF COLORADO
Department of State

1700 Broadway, Suite 270
Denver, CO 80290

Mike Coffman
Secretary of State

Holly Lowder
Director, Elections

The Testing Board has identified the following items as deficient in the voting system, requiring restriction for use of the voting system components based on the review and testing of the voting system for compliance with state requirements:

Software Restrictions:

Unity 3.0.1.1	Rule	Text
1) Functional Requirements Testing board was unable to complete testing in this area due to incorrect programming of ballots provided by vendor with no resolution.	27.2	Multiple Page Ballots. In any election where a multiple page printed ballot is used, a voter must vote and return all pages of the ballot at the same time. Any voter who has returned at least one page of a multiple page printed ballot will be considered to have voted. Any additional page returned at a later time shall not be counted but shall be appropriately marked, set aside, and preserved as other election materials in accordance with section 1-7-802, C.R.S.
2) Documentation Requirements Insufficient state documentation.	45.5.2.4.1	In addition to other documentation requirements in this rule, the voting system provider shall provide the following documents: (e) A list of minimum services needed for successful, secure and hardened operation of all components of voting system.
	45.5.2.5.2	The voting systems shall include detailed documentation as to the level, location, and programming of audit trail information throughout the system. The audit information shall apply to: (a) Operating Systems (workstation, server, and/or DRE); (d) Election Result Consolidation and Reporting.
	45.5.2.6.1	(d) The voting system shall meet the following requirements for operating system security: (iv) The voting system provider shall provide documentation containing a list of minimum services and executables that are required to run the voting system application;

Precinct Count Scanner Restrictions:

M100 - 5.2.1.0	Rule	Text
1) Functional Requirements Testing board was unable to complete testing in this area due to incorrect programming of ballots provided by vendor with no resolution.	27.2	Multiple Page Ballots. In any election where a multiple page printed ballot is used, a voter must vote and return all pages of the ballot at the same time. Any voter who has returned at least one page of a multiple page printed ballot will be considered to have voted. Any additional page returned at a later time shall not be counted but shall be appropriately marked, set aside, and preserved as other election materials in accordance with section 1-7-802, C.R.S.
	45.6.2.3.7	All ballots provided shall be blank with no marks on them. The following combinations of ballots are required: (d) One separate deck of ballots consisting of 200 ballots of any single precinct from the Coordinated election shall be provided that contains a two page ballot (races on four faces).
Ballot handling errors: misfeeds (<= 1 per 5000) and corrective action reporting. Device tested outside of acceptable criteria.	45.6.2.3.15 45.6.2.2.1	Test all ballot reading functions – are they accurate and reliable as described in the requirement - how do scanner(s) responds to smudges, folds, etc; response to valid and invalid or absence of marks. The system shall stop and inform operator of ballot handling errors such as misfeeds, damaged ballot, and multiple feeds. Also, give corrective measures to remove the ballot, sort is as unreadable (out stack) and gives a way to restart or recount the uncounted ballots. (Mis-feeds =< 1 per 5,000).
3) Physical and Design Characteristics	45.5.2.3.14	The voting system shall contain a control subsystem that consists of the physical devices and software that accomplish and validate the following operations: (a) Voting system Preparation - The control subsystem shall encompass the hardware and software required to prepare remote location voting devices and memory devices for election use. Remote site preparation includes all operations necessary to install ballot displays, software, and memory devices in each voting device. The control subsystem shall be designed in such a manner as to facilitate the automated validation of ballot and software installation and to detect errors arising from their incorrect selection or improper installation.
	45.5.2.3.15	The voting system shall have a high level of integration between the ballot layout subsystem and the vote tabulation subsystem. This integration shall permit and facilitate the automatic transfer of all ballot setup information from the automated ballot layout module to the single ballot tabulation system that will be used in a fully integrated manner for DRE, optical scan, and any other voting devices included in the voting system.
4) Documentation Requirements Insufficient state documentation.	45.5.2.2.3	The voting system provider shall publish and specify processing standards for each component of the voting system as part of the documentation required for certification.

M100 - 5.2.1.0	Rule	Text
Documentation Requirements continued	45.5.2.6.2	The voting system provider shall provide documentation detailing voting system security in the areas listed below. The system shall contain documented configurations, properties and procedures to prevent, detect and log changes to system capabilities for: (i) Preventing access to vote data, including individual votes and vote totals, to unauthorized individuals.

Central Count Scanner Restrictions:

M650 - 2.1.0.0	Rule	Text
1) Functional Requirements Testing board was unable to complete testing in this area due to incorrect programming of ballots provided by vendor with no resolution.	27.2	Multiple Page Ballots. In any election where a multiple page printed ballot is used, a voter must vote and return all pages of the ballot at the same time. Any voter who has returned at least one page of a multiple page printed ballot will be considered to have voted. Any additional page returned at a later time shall not be counted but shall be appropriately marked, set aside, and preserved as other election materials in accordance with section 1-7-802, C.R.S.
	45.6.2.3.7	All ballots provided shall be blank with no marks on them. The following combinations of ballots are required: (d) One separate deck of ballots consisting of 200 ballots of any single precinct from the Coordinated election shall be provided that contains a two page ballot (races on four faces).
Ballot handling errors: misfeeds (<= 1 per 5000) and corrective action reporting. Device tested outside of acceptable criteria.	45.6.2.3.15 45.6.2.2.1	Test all ballot reading functions – are they accurate and reliable as described in the requirement - how do scanner(s) responds to smudges, folds, etc; response to valid and invalid or absence of marks. The system shall stop and inform operator of ballot handling errors such as misfeeds, damaged ballot, and multiple feeds. Also, give corrective measures to remove the ballot, sort is as unreadable (out stack) and gives a way to restart or recount the uncounted ballots. (Mis-feeds =< 1 per 5,000).
2) Physical and Design Characteristics	45.5.2.3.14	The voting system shall contain a control subsystem that consists of the physical devices and software that accomplish and validate the following operations: (a) Voting system Preparation - The control subsystem shall encompass the hardware and software required to prepare remote location voting devices and memory devices for election use. Remote site preparation includes all operations necessary to install ballot displays, software, and memory devices in each voting device. The control subsystem shall be designed in such a manner as to facilitate the automated validation of ballot and software installation and to detect errors arising from their incorrect selection or improper installation.

M650 - 2.1.0.0	Rule	Text
Physical and Design Characteristics continued	45.5.2.3.15	The voting system shall have a high level of integration between the ballot layout subsystem and the vote tabulation subsystem. This integration shall permit and facilitate the automatic transfer of all ballot setup information from the automated ballot layout module to the single ballot tabulation system that will be used in a fully integrated manner for DRE, optical scan, and any other voting devices included in the voting system.
3) Documentation Requirements Insufficient state documentation.	45.5.2.2.3	The voting system provider shall publish and specify processing standards for each component of the voting system as part of the documentation required for certification.
	45.5.2.3.18	The approach to design shall be unrestricted, and it may incorporate any form or variant of technology that is capable of meeting the requirements of this rule, and other attributes specified herein. The frequency of voting system malfunctions and maintenance requirements shall be reduced to the lowest level consistent with cost constraints. Applicants are required to meet or exceed MIL-HDBK-454; "Standard General As Amended 10/2/07 Page 119 Requirements for Electronic Equipment" that is hereby adopted and incorporated by reference, as a guide in the selection and application of materials and parts only as is relevant to this section.

DRE Restrictions:

iVotronic ADA (3-Button) and Non-ADA 9.1.6.4	Rule	Text
1) Functional Requirements	45.5.2.9.12	The V-VPAT printer shall print at a font size no less than ten (10) points for ease of readability. Any protective covering intended to be transparent shall be in such condition that it can be made transparent by ordinary cleaning of its exposed surface.
	45.5.2.9.20	The V-VPAT shall allow a voter to spoil his or her paper record no more than two (2) times. Upon spoiling, the voter shall be able to modify and verify selections on the DRE without having to reselect all of his or her choices.
	45.5.2.9.21	Before the voter causes a third and final record to be printed, the voter shall be presented with a warning notice that the selections made on screen shall be final and the voter shall see and verify a printout of his or her vote, but shall not be given additional opportunities to change their vote.

iVotronic ADA (3-Button) and Non-ADA 9.1.6.4	Rule	Text
2) Physical and Design Characteristics	45.5.2.3.14	The voting system shall contain a control subsystem that consists of the physical devices and software that accomplish and validate the following operations: (a) Voting system Preparation - The control subsystem shall encompass the hardware and software required to prepare remote location voting devices and memory devices for election use. Remote site preparation includes all operations necessary to install ballot displays, software, and memory devices in each voting device. The control subsystem shall be designed in such a manner as to facilitate the automated validation of ballot and software installation and to detect errors arising from their incorrect selection or improper installation.
	45.5.2.3.15	The voting system shall have a high level of integration between the ballot layout subsystem and the vote tabulation subsystem. This integration shall permit and facilitate the automatic transfer of all ballot setup information from the automated ballot layout module to the single ballot tabulation system that will be used in a fully integrated manner for DRE, optical scan, and any other voting devices included in the voting system.
Vote data is visible during paper changing events on V-VPAT.	45.5.2.3.21	The voting system shall provide capabilities to enforce confidentiality of voters' ballot choices. (a) All optical scan devices, associated ballot boxes and V-VPAT storage devices shall provide physical locks and procedures to prevent disclosure of voters' confidential ballot choices during and after the vote casting operation.
3) Documentation Requirements Insufficient state documentation.	45.5.2.2.3	The voting system provider shall publish and specify processing standards for each component of the voting system as part of the documentation required for certification.
	45.5.2.3.13	All DRE voting devices shall use touch screen technology or other technology providing visual ballot display and election. The voting system provider shall provide documentation concerning the use of touch screen or other display and selection technology, including but not limited to: (b) Technical documentation describing the nature and sensitivity of any other technology used to display and select offices, candidates, or issues; (c) Any mean time between failure (MTBF) data collected on the vote recording devices; and

iVotronic ADA (3-Button) and Non-ADA 9.1.6.4	Rule	Text
Documentation Requirements continued	45.5.2.3.18	The approach to design shall be unrestricted, and it may incorporate any form or variant of technology that is capable of meeting the requirements of this rule, and other attributes specified herein. The frequency of voting system malfunctions and maintenance requirements shall be reduced to the lowest level consistent with cost constraints. Applicants are required to meet or exceed MIL-HDBK-454; "Standard General As Amended 10/2/07 Page 119 Requirements for Electronic Equipment" that is hereby adopted and incorporated by reference, as a guide in the selection and application of materials and parts only as is relevant to this section.
	45.5.2.3.22	The voting system and all associated components shall have an estimated useful life of at least eight (8) years. Voting system provider shall provide documentation of the basis for the estimate.
4) Audit Capacity	37.1.4	The voting systems described in the foregoing paragraphs shall produce a record with an audit capacity for such system. (b) The voting system shall provide the voter with an opportunity to change the ballot or correct any error before the permanent paper record is produced.
5) Accessibility	1-5-704 35.1.5 35.1.7	(1) Notwithstanding any other provision of this article, each voting system certified by the secretary of state for use in local, state, and federal elections shall have the capability to accept accessible voter interface devices in the voting system configuration to allow the voting system to meet the following minimum standards: (d) Devices providing audio and visual access shall be able to work both separately and simultaneously. (f) Any voting system that requires any visual perception shall allow the font size as it appears to the voter to be set from a minimum of fourteen points to a maximum of twenty-four points before the voting system is delivered to the polling place. A san-serif font of 18 points will allow the most universal access. (m) Voting booths shall have voting controls at a minimum height of thirty-six inches above the finished floor with a minimum knee clearance of twenty-seven inches high, thirty inches wide, and nineteen inches deep, or the accessible voter interface devices shall be designed so as to allow their use on top of a table to meet such requirements. Tabletop installations shall ensure adequate privacy.
Due to inability to process multiple languages on this system.	34.5	If a political subdivision acquires a new voting system, the system must be accessible to persons with physical, cultural/educational, mental/cognitive disabilities and provide the voter in a manner that provides the same opportunity for access and participation (including privacy and independence) as for other voters.

iVotronic ADA (3-Button) and Non-ADA 9.1.6.4	Rule	Text
	35.1.15	If a forward approach by a person in a wheelchair to a voting system is necessary, the maximum high-forward reach allowed shall be 48 inches (1220 mm) and the minimum low-forward reach shall be 15 inches (380 mm). If the high-forward reach is over an obstruction, reach and clearances shall be as shown in the Figure 1., or otherwise in accordance with the Americans with Disabilities Act Accessibility Guidelines for Buildings and Facilities ("ADAAG"), as written at the time the system is certified for use in the state of Colorado;
	35.1.17	The highest operable part of controls, dispensers, receptacles, and other operable equipment shall be placed within at least one of the reach ranges outlined in paragraphs (15) and (16) of this subsection.
	37.1.4	The voting systems described in the foregoing paragraphs shall produce a record with an audit capacity for such system. (d) The paper record shall be accessible for individuals with disabilities including non-visual accessibility for the blind and visually impaired, in a manner that provides the same opportunity for access and participation (including privacy and independence) as for other voters.
Accessibility continued	45.5.2.8.1	Specific minimum accessibility requirements include those specified in section §1-5-704 C.R.S., Secretary of State Rule 34, Rule 35 and the following: (b) Audio ballots shall meet the following standards: (ii) The audio system shall allow voters to control within reasonable limits, the rate of speech. (f) Adjustability of color settings, screen contrasts and/or screen angles/tilt may be made by either the poll worker or voter if the system uses a display screen. A minimum of two color settings, two contrast settings and two angles shall be available for all display screens.
	45.5.2.8.2	Documentation of the accessibility of the voting system shall include the following items at a minimum: (c) Technology used by the voting system that prevents headset/headphone interference with hearing aids; (g) Various methods of voting to ensure access by persons with multiple disabilities; (i) Method for adjusting color settings, screen contrasts, and screen angles/tilt if the system uses a display screen.
	45.5.2.9.10	The V-VPAT device shall be designed to allow every voter to review, and accept or reject his/her paper record in as private and independent manner as possible for both disabled and nondisabled voters.

2007-CDOS-ESS-001-0403
PROJECT OVERVIEW BINDER “A.3”

CONDITIONS

STATE OF COLORADO
Department of State

1700 Broadway, Suite 270
Denver, CO 80290

Mike Coffman
Secretary of State

Holly Lowder
Director, Elections

Conditions for Use

The Testing Board recommends the Secretary of State adopt the following conditions for use of the voting system. These conditions are required to be in place ***should*** the Secretary approve for certification any or all of the items indicated in the **COMPONENTS** section. Being that many conditions address the security, auditability and availability of the system component, the testing board would firmly reject any option which removes, replaces or diminishes the conditional requirement and still allow the system to be used and recommended for certification. Any "Y" value in the recommendation table would change to a "N" value with any change to the conditions.

Global Conditions (applies to all components):

1) Modem and other telecommunication devices may not be used on any subsystem component - system provider was unable to meet or provide prerequisite FIPS 140/180 certifications.

2) Provisional ballots must be processed separately from non-conditional ballots - system subcomponents are unable to functionally differentiate and correctly process to Colorado specific requirements.

3) Coordination of Escrow Setup - Upon Certification, voting system manufacturer must coordinate the Escrow of the TRUSTED BUILD software with SOS escrow, or third party escrow service as required by Rule 11 prior to use in Colorado.

4) Abstract Report generation - abstracts used for State reporting must come from Unity Software, or other external solution, rather than from the specific device.

5) Trusted Build Verification
   a) The system components do not allow for proper verification of trusted build software. Any breach of custody and/or other security incidents will require the rebuild of the component with the state trusted build software. This requirement applies to all voting devices, firmware and software components of the system.
   b) Counties shall ensure that hardware, software and firmware purchased for use of the system matches the specifications of VSTL/EAC and/or State Certified and trusted versions, not to the version presented in the vendor documentation.

6) Counties using the voting system shall testify through their security plan submission that the voting system is used only on a closed network.

7) Due to known system failures, the vendor did not submit any information to the testing board for testing alternative language requirements. Use of this voting system will be limited to counties that are not required to provide alternative languages to voters under Secretary of State Rule 45.5.2.3.4.

8) The voting system does not have the means to process multi-paged ballots. Counties must take this into consideration when programming and designing ballot layouts to accommodate the requirements in C.R.S. 1-5-704 within the limitations of the system.

Software Conditions (Unity 3.0.1.1):

1) System/database/network security hardening.

a) Because the voting system operates in a non-restricted system configuration containing open file system access to locate, copy, open and overwrite without detection, election vote content database files outside of election management system application by third-party tools, counties will be required to modify their physical environmental conditions. If the system is approved for certification, counties shall submit plans for approval to the Secretary of State's office on overcoming these conditions through county environmental and/or procedural changes where possible.

b) In addition to physical environmental changes, counties shall create a second (or backup) copy of the Unity database that is created immediately after the point of memory card downloads. The backup copy shall be stored on closed CD Media and documented as matching the master database. This process shall be observed by two election staff members. Chain of custody documents shall be generated for the media, and the media shall be sealed with at least two tamper evident seals and stored in a sealed or lockable transfer case that is stored in a limited access area. AFTER the close of polls, the designated election official shall load the sealed copy of the database onto the server and proceed with uploading memory cards after documenting the loading of the backup master database onto the system. After loading the sealed database copy, the county shall re-secure the database copy with seals (updating necessary logs) in the limited access location.

c) Additionally, to overcome deficiencies in security and auditing of the system, the county will be required to perform increased Election Night and Post Election Audits for this system. All post-election audit data shall process a hand count of paper ballots which shall match the totals report from the specific device, as well as the totals for the Unity/ERM database. Counties shall prepare for this event with one of two methods:

Option #1 – Prepare for the upload of memory cartridges/components as normal. Print necessary zero report. Upon uploading each individual memory card, print a summary report showing the change in totals from the upload of the memory card. Label the report to match the name/number of the memory card uploaded. Continue to upload memory cards and print totals reports to match. When auditing a specific device, use the difference between the report totals for the memory card selected for the audit and the totals from the immediately preceding memory card report to calculate vote totals generated by the Unity/ERM software.

When memory cards are delivered to the county for upload, the machine generated report shall be delivered for inspection as well. On election night, when the summary report indicated above is created, the difference totals (delta report) are immediately compared to the totals from the report generated by the device at the polling place. If the reports match, the public is ensured that the totals from the polling place match the totals from the county server. If the totals are different, the county is to report the situation (on election night) to the Secretary of State for audit, security and remedy procedures.

During the post election audit process, the totals of the paper record for the specific device are to be hand counted and verified against the electronic record for the device. The canvass board shall report the verification of three totals to match – the paper record of the device, the totals of the

electronic vote on the device, and the totals in the Unity/ERM server; OR

Option #2 – Prepare for the upload of memory cartridges by creating one master default database (containing all memory cards/cartridges). Create individual databases to contain values (upload data) for each separate memory card (or in some instances by batch of ballots – see condition #4b under Central Count devices. Upload memory card/cartridges into master database, and into the specific database created for that memory card (two separate uploads). This process must happen on Election Night and with observation by at least two people. Election summary reports shall be printed from each individual database and manually added together. The totals from the individual databases must match the master database before proceeding. Upon verification that the master and individual databases match, the county can then use the individual reports to conduct a hand count of the paper ballot (or paper record) generated by the device to show that the ERM totals match. The verification of the separate upload databases verify that the database totals match the field totals on each memory card device, as was designed after the point of Logic and Accuracy testing took place.

2) Ballot-On-Demand restriction.
No provision for ballot reconciliation. This will require counties to have an extra supply of preprinted ballots on hand.

3) Audit Trail Information:

a) Counties will be required to produce certain reports identified in C.R.S. 1-7-509 using an external process which will include at a minimum exporting result from the Unity/ERM software for processing by other methods.

b) Operators of the system shall also be required to maintain logs indicating use of the report printing functions of the software, and detailed information to changes of the system including hardware changes which shall include: insert removable media, remove removable media, modify system hardware drivers, modify system physical hardware, and any other system property changes made by either judges or other trusted staff. Logs shall be maintained physically in a file outside or separate from the database, which is NOT accessible for review and/or modification by user/operator accounts on the system, but that is readily accessible to election officials or other interested party.

4) Performance Deficiencies.
Due to failures in performance, counties shall allow extra time for downloads and uploads of memory card devices. This may impact programming, testing and use of the system on election night. Counties shall ensure trusted staff is properly trained on this issue and accommodating the allowable time required for programming memory devices.

5) Provisional Ballots.
The software is not capable of processing provisional ballots internally to accept federal and state only questions. A procedure outside of the voting system will be required. Additionally, the abstracts and reports created by the software do not meet the requirements of rule 41.6.3(g) and users of the system will be required to generate an abstract outside of the voting system.

6) Election Database Creation and testing.
a) The system was unable to be fully tested with all testing board requirements for ballot layouts as required. Therefore, additional testing will be required by counties for both electronic and paper ballots to ensure all voting positions are working as designed prior to each election. This shall include

ordering a complete set of at least 5 ballots of each style that contain the prescribed design for that election. County officials shall mark each possible position for each race on the ballots. All ballots shall be tested internally prior to the public logic and accuracy test. The goal of the pretest is to ensure that all available positions are counting when marked correctly.

b) Counties are to ensure that ballots are designed and created according to state requirements. The system does not prevent a "backflow" of data changes, nor do system logs accurately represent changes made within the system, and the effect of the changes. Counties using the system shall be required to maintain a written log/audit of changes made to any component of the system after the point when ballots are ordered and/or when any memory cards are created/burned – whichever is earlier.

Precinct Count Scanner Conditions (M100):

1) Intrusion seals for protection of Trusted Build firmware.

Device has no provision of Trusted Build verification once installed. Counties will be required to maintain constant seals on voting device memory slot, back panel, and other entry points as indicated by the Secretary of State.

2) External Battery backup (UPS) devices required.

Insufficient internal power reserves to sustain minimum 3 hour continuous operation. Counties shall purchase and use an external power supply that meets or exceeds the vendors' recommendation for the component.

3) Device security accessibility.

a) Device level administrative functions requiring access involving the use of keys, memory cards, and passwords must be restricted to single person entry with detailed logs.

b) County use of voting system will require use of Unity Software to modify the "administrator" password on the voting devices.

4) Ballot/Race conditions simulation.

Additional County testing shall be required to accommodate ballots with conditions from each election. This shall include ordering a complete set of at least 5 ballots of each style that contain the prescribed design for that election. County officials shall mark each possible position for each race on the ballots. All ballots shall be tested internally prior to the public logic and accuracy test. The goal of the pretest is to ensure that all available positions are counting when marked correctly.

5) Audit trail information:

a) Operators of the system shall also be required to maintain logs indicating use of the administrator functions of the device by either judges or other trusted staff.

b) Counties will be required to produce certain reports identified in C.R.S. 1-7-509 using an external process which will include at a minimum exporting result from the Unity software for processing by other methods.

c) Judges shall be required to include device serial number on all reports regarding use of the device. Additionally, the county shall include the device serial number on applicable reports from the device.

d) Counties will be required to perform additional post election audit functions for the device to accommodate for security deficiencies. In an effort to increase confidence in the recording of votes

by the device, the post-election audit shall include the verification of the hand count of paper ballots to match the totals generated from the Unity/ERM software as indicated in Software condition #1c.

6) Voting Secrecy.
Insufficient privacy of ballot was detected using secrecy sleeve. Election administrators must ensure that the system secrecy sleeve (from ESS) is used for ballots with only one column. For ballots with more than one column, the counties shall create a secrecy sleeve to accommodate the deficiency and submit design form to Secretary of State for approval.

Central Count Scanner Conditions (M650):

1) Intrusion seals for protection of Trusted Build firmware.
Device has no provision of Trusted Build verification once installed. Counties will be required to maintain constant seals on voting device memory slot, back panel, and other entry points as indicated by the Secretary of State.

2) External Battery backup (UPS) devices required.
Insufficient internal power reserves to sustain minimum 3 hour continuous operation. Counties shall purchase and use an external power supply that meets or exceeds the vendors' recommendation for the component.

3) Audit trail information:
a) Judges shall be required to include device serial number on all reports regarding use of the device. Additionally, the county shall include the device serial number on applicable reports from the device.
b) Counties will be required to produce certain reports identified in C.R.S. 1-7-509 using an external process which will include at a minimum exporting result from the Unity software for processing by other methods.
c) Batches must be saved to zip disk. Save must take place after each batch.
b) Counties will be required to perform additional post election audit functions for the device to accommodate for security deficiencies. In an effort to increase confidence in the recording of votes by the device, the post-election audit shall include a hand count of at least the following amounts of ballots:

Considering the closest race in the election, if the difference between the top two candidates for the race is:

10% or greater, then hand count 60 ballots for every 10,000 cast;
9.00% - 9.99%, then hand count 65 ballots for every 10,000 cast;
8.00% - 8.99%, then hand count 70 ballots for every 10,000 cast;
7.00% - 7.99%, then hand count 80 ballots for every 10,000 cast;
6.00% - 6.99%, then hand count 95 ballots for every 10,000 cast;
5.00% - 5.99%, then hand count 115 ballots for every 10,000 cast;
4.00% - 4.99%, then hand count 140 ballots for every 10,000 cast;
3.00% - 3.99%, then hand count 185 ballots for every 10,000 cast;
2.00% - 2.99%, then hand count 275 ballots for every 10,000 cast;
1.00% - 1.99%, then hand count 550 ballots for every 10,000 cast;

0.01% - 0.99%, then hand count 1200 ballots for every 10,000 cast.

The verification of the hand count of paper ballots shall match the totals generated from the Unity/ERM software as indicated in Software condition #1c. Counties shall load only the master database from the secured storage location for processing the post election audit ballots as indicated in Software Condition #1b. Counties shall prepare database and batches of ballots prior to scanning into system (for election results) to accurately generate reports in batch sizes as necessary for the audit. If the county or system is not capable of accommodating the requirement of batch size after the outcome of the election is revealed, the highest percentage of ballots shall be used for the audit process.

4) Ballot/Race conditions simulation.
Additional County testing shall be required to accommodate ballots with conditions listed. This shall include ordering a complete set of at least 5 ballots of each style that contain the prescribed design for that election. County officials shall mark each possible position for each race on the ballots. All ballots shall be tested internally prior to the public logic and accuracy test. The goal of the pretest is to ensure that all available positions are counting when marked correctly.

5) Device Security Accessibility.
Device level administrative functions requiring access involving the use of keys, memory cards, and passwords must be restricted to single person entry with detailed logs.

DRE Conditions (iVotronic):

1) External Battery backup (UPS) devices required.
Insufficient internal power reserves to sustain minimum 3 hour continuous operation. Counties shall purchase and use an external power supply that meets or exceeds the vendors' recommendation for the component.

2) Intrusion seals for protection of Trusted Build firmware.
   a) Device has no provision of Trusted Build verification once installed. Counties will be required to maintain constant seals on voting device memory slot, back panel, and other entry points as indicated by the Secretary of State.
   b) Election official shall go into Unity software and change passwords for the iVotronic.

3) Ballot/Race conditions simulation.
Additional County testing shall be required to accommodate ballots with conditions listed. This shall include ordering a complete set of at least 5 ballots of each style that contain the prescribed design for that election. County officials shall mark each possible position for each race on the ballots. All ballots shall be tested internally prior to the public logic and accuracy test. The goal of the pretest is to ensure that all available positions are counting when marked correctly. All ballots in this detail shall be "marked" using the DRE device as applicable for similar testing.

4) V-VPAT paper record shall be handled per Rule 11.6.
Prescribed paper record is of the thermal type and requires special storage conditions to avoid legibility degradation.

5) Audit trail information:
   a) Counties will be required to produce certain reports identified in C.R.S. 1-7-509 using an external process which will include at a minimum exporting result from the Unity software for processing by

other methods.
b) Operators of the system shall also be required to maintain logs indicating use of the administrator functions of the device by either judges or other trusted staff.

6) V-VPAT Security.
a) The V-VPAT device provides no assurance that it cannot accommodate other devices, and/or the device is a standard communication port. This connection between the V-VPAT printer and the DRE unit shall be secured with tamper evident seals with proper chain of custody documentation to prevent and detect tampering.
b) Only the 9" screen shall be used when using this system. The vote data can be viewed by election judges when the paper is changed when the 4.5" screen is used.
c) The lock on the V-VPAT must be sealed with a tamper-evident seal.
d) Only firmware that is loaded during the Trusted Build shall be allowed on the V-VPAT device.

7) Accessible operation.
a) Due to the inability for the voter to pause and resume the audio text, election judges shall provide instructions specific to this fact to the voters and operations for repeating the text if text was missed, which shall include details on navigating forward and backwards through the system prompts.
b) A headset with an adjustable volume, which meets the State of Colorado specifications, must be provided.

8) Device Security Accessibility.
a) Device level administrative functions requiring access involving the use of keys, memory cards, and passwords must be restricted to single person entry with detailed logs.
b) Devices deployed in Colorado shall require the disabling of the PEB activation port due to security concerns discovered through functional testing. A common magnet (example = money clip) can cause a series of attacks and unauthorized control of the device.
c) An alternative security measure to 8(b) would be to protect the PEB slot by attaching a lockable cover similar to Figure 8.1 (padlock type) or Figure 8.2 (integral keyed lock).

Figure 8.1 – Showing PEB slot covered with padlock type enclosure.

Figure 8.2 – Showing PEB slot covered with integral keyed locked enclosure.

(Note: Figures are intended to illustrate concept implementation options, not as requirements of apparatus specification.)

The lock and cover used in Figures 8.1 and 8.2 provides a level of resistance to externally triggered system events requiring operational intervention. Both can be purchased at a hardware

store and attached to the device with industrial strength Velcro. Each allows lift-off box removal for DRE stowage.

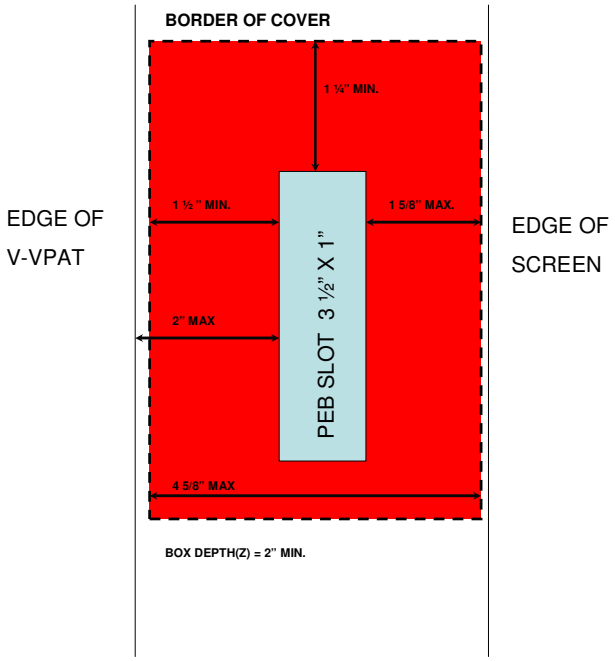


Figure 8.3 – Showing PEB port security cover dimensional requirements.

Dimensions shown in Figure 8.3 indicate **minimum** areal coverage to protect magnetic switch from false triggering resulting in system responses potentially requiring operational intervention, and also reflect the **maximum** coverage area for a greater measure of resistance from outlier false triggering without intruding upon viewabilty of the V-VPAT record and the touch-screen display.

Additional information on the need for the PEB cover, and the issues related can be found in the **Testing Board Comments** section of this document.

2007-CDOS-ESS-001-0403
PROJECT OVERVIEW BINDER “A.3”

COMMENTS

STATE OF COLORADO
Department of State

1700 Broadway, Suite 270
Denver, CO 80290

Mike Coffman
Secretary of State

Holly Lowder
Director, Elections

Testing Board Comments - Addendum

This section of the document will address some of the specific security issues as they relate to the PEB slot on the iVotronic device. This documentation shall serve as an addendum to the original Testing Board comments from the release of the original findings on Dec. 17th.

The iVotronic DRE uses an infrared serial port to communicate with proprietary device hardware tokens. These proprietary devices are referred to as Personalized Electronic Ballots or PEBs. The iVotronic hardware contains a magnetic reed switch which allows communication with the PEB. Along with the specific data necessary for the type of PEB, the PEB also contains a magnet which activates the appropriate reed switch inside the iVotronic.

PEBs received by the Testing Board were used for supervisor functions (including ballot activation) (containing a red band), or voter activation PEBs (containing a blue band). Information publicly available contains documentation of a third type (or kind) of PEB which is referred to as a Factory or QA PEB which can be used to override any passwords or election qualifier information typically used as security features for a deployed device.

The findings of the Testing Board are not to be deemed comprehensive, as other tests conducted by independent parties on the devices have taken additional steps to understand the programming process of the PEB and readings of the infrared port. While the Testing Board did not re-program any PEBs during testing, it has been made clear that the reprogramming of a PEB can take place with any device that transmits an infrared signal (cell phone, palm pilot, etc.). The materials to conduct this type of tests are not available to the Colorado Testing Board, but we recognize that additional negative security impacts may be possible given the technology used and the accessibility of the PEB port.

Specific findings of the PEB port by the Colorado Testing Board indicate that the token activation system by means of the PEB can be easily defeated. The Testing Board has proposed a solution to mitigate each of these problems in the **conditions for use** section of this document. The following list indicates the specific possibilities of attack:

<u>Prematurely Cast or Cancel Ballot</u> – Can be implemented by any magnet inserted into PEB slot. Presents chaos and voter confusion. Voter is presented with a non-voting screen asking voter to cancel or cast the ballot. Machine resets for next voter.

<u>Reboot (power cycle) terminal</u> – insertion of magnet plus pressing vote button will power on/off the voting terminal when between voters. A voter could, after successfully voting, power down the device. Upon power-up of device (which is accomplishable by magnet) instances of stack dump errors and other unrecoverable errors have occurred causing denial of service for the device.

<u>Unauthorized Recalibration</u> – Can be implemented by any supervisor programmed PEB, or QA PEB – no password is required and the ability is not county or election specific. Operators are capable of entering calibration screen with a foreign Supervisor PEB purchased on independent market, other jurisdiction, or

home made PEB emulator. The screen can be calibrated so that touching a desired point would be recorded by the machine as a touch at a different location. This could cause voter input for one candidate and selection recorded for a different candidate. During testing, the calibration was changed such that on entering vote mode with correct PEB, workers closed polls accidentally. This type of attack causes denial of service, chaos, confusion and mistrust of electronic voting devices by the voter.

Unauthorized control of device with QA PEB[1] –The use of a QA PEB overrides all passwords that may be programmed by the county. Control can be achieved by inserting the QA PEB at any time during voting process including between voters. As previously mentioned, the QA PEB can be used to override and/or change existing passwords on a voting device. This type of PEB could be obtained by way of a home made PEB emulator device (palm pilot) or a disgruntled ES&S employee. Acquiring such a PEB would allow unlimited access to all menu items and functions of the device which include modifying vote data, modifying firmware, and all service/administration menus.

Maliciously copied PEB[1] – Similar to the QA PEB, the ease of copying PEB data onto IR reading devices, allows for easily modifying the information and performing multiple attacks. Copied PEB data may need the Election Qualification Code (EQC) information from the Supervisor PEB for that election, or modifying the value indicating the "type" of PEB in the firmware which would create a QA PEB. Creating a PEB with customized data allows for attacks which include: Clearing Vote Data, Introducing malicious firmware, change vote data, introduce additional vote data, change future vote data, exposing buffer overflow errors, and denial of service (DoS) of voting device.

The proposed condition to provide physical protection for the PEB port is not ideal, but it does mitigate a considerable portion of potential attacks with common devices. The ideal solution to this problem would require a re-write of firmware to change the magnetic features of the PEB activation token. The presence of physical security observation is not a viable solution due to the ease of access to the PEB port, the observer would also compromise any voter privacy requirements.

Additionally, the Testing Board defers to the report from the Office of Cyber security which supports overall findings of security weakness in the ES&S system.

Finally, the Testing Board has included at the end of this section the report and findings from the Everest report available from the State of Ohio's Secretary of State Website, containing supporting documentation as well as detailed findings regarding the iVotronic: http://www.sos.state.oh.us/sos/info/everest.aspx (pages 1-118).

In addition to other changes to the Project Overview Binder, the Testing Board would like the opportunity to note that the Executive Summary report was not developed independently by the Testing Board.

1. This test was not conducted by the Testing Board primarily due to lack of tools and time to execute. This information is extrapolated from the everest documentation from the Ohio Secretary of State's office and is in-line with Colorado Findings regarding the ES&S iVotronic.

Test Board Comments - Original

The Testing Board has unveiled a known reality that no computer system is perfect. Additionally, we have discovered and documented that no system can currently meet the requirements of Rule 45 as applied in its strictest sense. Where possible, the Testing Board attempted to overcome these deficiencies in the form of "conditions for use" of the system – procedural workarounds.

The Testing Board recognizes that the conditions created are in essence a "last resort" workaround to accommodate requirements that do not meet specific sections of Colorado Revised Statutes 1-5-615. The preference of the Testing Board would be to have the specific deficiencies addressed with a system solution as required. Given the ability to mitigate deficiencies with procedural workarounds (C.R.S. 1-5-621), the Testing Board presents conditional use scenarios in the **"Conditions"** section that are directly tied to the recommendation status. Being that many workarounds address the security, auditability and availability of the system component, the Testing Board would firmly reject any option which removes, replaces or diminishes the conditional requirement and still allow the system to be used and recommended for certification. Any "Y" value in the **Recommendation** table would change to a "N" value with any change to the conditions.

These conditional procedures rely heavily on proper execution by county administrators and/or election judges. While we have faith that these dedicated workers will attempt to perform their duties to the best of their abilities, a majority of the conditions involve a human element which may or may not produce the acceptable outcome. This single factor alone causes concern that a security issue may not be resolvable in a post-election scenario.

Finally, it is of value to point out that the conditions that address security specific events are only addressing the attack scenario of a change in vote totals (refer to Cyber Security Report). The essence of the workaround in this case is to ensure that the vote totals calculated electronically are a match to the paper records. This requires absolute assurance that all paper records exist and are auditable for a successful outcome and high confidence in the report of votes by any given county.

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

Final Report
December 7, 2007

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

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

Project Personnel

Pennsylvania State University Team

Patrick McDaniel, *Principal Investigator & Team Lead*

Kevin Butler
Pennsylvania State University

William Enck
Pennsylvania State University

Harri Hursti
Independent Contractor

Steve McLaughlin
Pennsylvania State University

Patrick Traynor
Pennsylvania State University

University of Pennsylvania Team

Matt Blaze, *Team Lead*

Adam Aviv
University of Pennsylvania

Pavol Černý
University of Pennsylvania

Sandy Clark
University of Pennsylvania

Eric Cronin
University of Pennsylvania

Gaurav Shah
University of Pennsylvania

Micah Sherr
University of Pennsylvania

WebWise Security, Inc.

Giovanni Vigna, *Team Lead*

Richard Kemmerer
WebWise Security, Inc.

Davide Balzarotti
WebWise Security, Inc.

Greg Banks
WebWise Security, Inc.

Marco Cova
WebWise Security, Inc.

Viktoria Felmetsger
WebWise Security, Inc.

William Robertson
WebWise Security, Inc.

Fredrik Valeur
WebWise Security, Inc.

Policy and Document Consultants

Joseph Lorenzo Hall
University of California, Berkeley

Laura Quilter
University of California, Berkeley

CONTENTS

I Project Overview **1**

1 Executive Summary **3**

2 Overview **5**
2.1 Report Structure . . . 5
2.1.1 Reading Recommendations . . . 5
2.2 Evaluation Design . . . 6
2.3 Methodology . . . 6
2.3.1 Prior Studies . . . 7
2.4 Limitations . . . 8
2.5 Acknowledgments . . . 10

3 Threat Model **11**
3.1 Reference Model . . . 11
3.1.1 Pre-Voting . . . 12
3.1.2 Voting . . . 13
3.1.3 Post-Voting . . . 14
3.2 Attacker Goals . . . 14
3.2.1 Producing Incorrect Vote Counts . . . 15
3.2.2 Blocking Some or All Voters from Voting . . . 15
3.2.3 Cast Doubt on the Legitimacy of the Election Results . . . 15
3.2.4 Delay the Results of the Election from Becoming Known . . . 16
3.2.5 Violating Ballot Secrecy . . . 16
3.3 Potential Attackers . . . 17
3.3.1 Outsiders . . . 17
3.3.2 Voters . . . 18
3.3.3 Poll Workers . . . 19
3.3.4 Election Officials . . . 20
3.3.5 Vendor Employees . . . 20
3.4 Types of Attacks . . . 21
3.5 Procedures . . . 22
3.5.1 Ohio Procedures on Information Technology and Networking . . . 23
3.5.2 Mechanisms for Tamper Sealing . . . 24
3.5.3 Chain-of-Custody Logs . . . 25
3.5.4 Recounting Optical Scan Ballots . . . 26
3.6 Software Engineering and Maintenance . . . 26

II Analysis of the Election Systems & Software, Inc. Voting Systems **27**

4 ES&S Executive Summary **29**

5 ES&S System Overview **31**
5.1 Architectural Overview . . . 31
5.2 System Components . . . 33
5.2.1 Unity . . . 33
5.2.2 iVotronic . . . 35
5.2.3 Real-Time Audit Log Printer . . . 36
5.2.4 Personalized Electronic Ballot . . . 37
5.2.5 PEB Reader . . . 37
5.2.6 Compact Flash Cards . . . 38
5.2.7 Communication Pack . . . 38
5.2.8 Model 100 . . . 38
5.2.9 PCMCIA Memory Cards . . . 39
5.2.10 Model 650 . . . 39
5.2.11 Zip Disks . . . 39
5.2.12 AutoMARK Voter Assist Terminal . . . 39
5.3 Election Procedures . . . 40
5.3.1 Preparation . . . 40
5.3.2 Touchscreen (DRE) Voting . . . 40
5.3.3 Precinct-Based Optical Scan Voting . . . 41
5.3.4 Centrally Counted Optical Scan Voting . . . 42
5.4 Software Versions . . . 42
5.5 Unity Acronyms and File Extensions . . . 44

6 ES&S Systemic and Architectural Issues **49**
6.1 Ineffective Access Control . . . 49
6.1.1 iVotronic passwords and PEB-based access controls . . . 50
6.1.2 Physical security, locks and seals . . . 52
6.2 Critical Errors in Input Processing . . . 53
6.2.1 Unity . . . 53
6.2.2 iVotronic . . . 54
6.3 Ineffectively Protected Software and Firmware . . . 54
6.3.1 iVotronic firmware . . . 54
6.3.2 Unity software . . . 55
6.3.3 M100 firmware . . . 56
6.3.4 Viral propagation . . . 56
6.4 Ineffective Cryptography and Data Authentication . . . 57
6.4.1 Unauthenticated M100 data . . . 57
6.4.2 Ineffective iVotronic cryptography . . . 57
6.4.3 Poor data validation of precinct results in Unity . . . 58
6.5 Procedural Mitigations . . . 58

7 ES&S Specific Weaknesses and their Implications **59**
7.1 Unity . . . 59
7.1.1 Unity ERM buffer overflow when reading a Master PEB . . . 59
7.1.2 Data from M100 can cause a buffer overflow in Unity . . . 60
7.1.3 Unity accepts memory media with unauthorized precinct results . . . 60
7.1.4 Integrity of data on a PCMCIA card is not protected . . . 61
7.1.5 Processing audit data can cause a buffer overflow of a global variable . . . 61
7.1.6 Unity decrypts a PEB using the EQC from the PEB . . . 62
7.1.7 Unity contains large pieces of duplicated code . . . 62
7.1.8 Unity contains many small buffer overflows . . . 63
7.1.9 SQL injection to bypass authentication in EDM, ESSIM, and Audit Manager . . . 63
7.1.10 An M100 PCMCIA card can be read multiple times without warning . . . 63
7.1.11 Assumptions on the environment for Unity are unspecified . . . 64
7.1.12 iVotronic Image Manager bundled with vulnerable Java Runtime Environment . . . 64
7.2 iVotronic . . . 65
7.2.1 Election encryption/decryption key is recoverable from a PEB . . . 65
7.2.2 PEBs may be emulated using infrared-capable devices . . . 65
7.2.3 PEB Emulators can read and/or write arbitrary data on a PEB . . . 66
7.2.4 Emulated PEBs permit multiple votes . . . 66
7.2.5 Buffer overflow in poll opening process is exploitable . . . 67
7.2.6 Buffer overflow in “Hotspot” image processing is exploitable . . . 67
7.2.7 Buffer overflow in Supervisor iVotronic initialization process is exploitable . . . 68
7.2.8 Service Menu Mode in iVotronic does not require properly configured PEB . . . 69
7.2.9 (Emulated) Initialization PEB can close polls and reset passwords . . . 69
7.2.10 (Emulated) Factory QA PEBs bypass all password checks . . . 69
7.2.11 (Emulated) Factory QA PEB can clear a terminal, erasing all vote and audit data . . . 70
7.2.12 (Emulated) Factory QA PEB can lock a terminal . . . 71
7.2.13 iVotronic touchscreens can be miscalibrated to deny certain ballot selections . . . 71
7.2.14 Connection to RTAL/VVPAT printer is physically unprotected . . . 71
7.2.15 Audit data is insufficiently randomized . . . 73
7.2.16 VVPAT barcodes contain timestamps . . . 73
7.2.17 VVPAT barcodes contain vote information . . . 73
7.2.18 Accuracy testing mode detectable . . . 75
7.3 M100 . . . 75
7.3.1 The M100 does not password protect the procedure for firmware uploads . . . 75
7.3.2 The M100 does not perform cryptographic integrity checks on firmware uploads . . . 76
7.3.3 The M100 uses the same facility for configuring an election as it does for firmware uploads . . . 76
7.3.4 The M100 locks are easily manipulated . . . 76
7.3.5 The keys for the M100 locks are the same across all M100 machines . . . 76
7.3.6 CRC routines used on the M100 are easily derivable . . . 77
7.3.7 The M100 PCMCIA cards do not contain cryptographic integrity checks . . . 77
7.3.8 M100 uses data offsets defined on the PCMCIA card to determine pointers used by the firmware . . . 77
7.3.9 M100 accepts counterfeit ballots . . . 78
7.4 M650 . . . 79
7.4.1 M650 runs executable on Zip Disk on power up . . . 79

7.4.2 M650 does not authenticate election definition and election macro language (e-code) files . . . 79
7.4.3 M650 fails to validate variable-length strings . . . 80
7.4.4 M650 fails to protect against integer overflows . . . 81
7.4.5 M650 accepts counterfeit ballots . . . 81

8 ES&S Software Engineering Issues 83
8.1 Complexity . . . 83
8.1.1 Programming Languages . . . 83
8.1.2 Third-Party Software . . . 84
8.1.3 Insufficient Documentation . . . 84
8.2 Improper Message Passing . . . 85
8.3 Static Code Analysis . . . 86

9 ES&S Example Attack Scenarios 89
9.1 Tools . . . 89
9.1.1 A framework for delivering a malicious payload to iVotronic systems . . . 90
9.1.2 Pebserial: a tool for reading and writing PEBs . . . 90
9.1.3 The iVotronic Serial Debugger . . . 91
9.1.4 A tool to read and write M100 PCMCIA memory cards . . . 91
9.1.5 The JTAG hardware debugger . . . 91
9.1.6 A tool for extracting the QNX filesystem from the M100 . . . 92
9.2 Physical Security and Security Seals . . . 93
9.3 Successful Attack Scenarios . . . 93
9.3.1 Attack Scenario peb.1: Changing an Unattentive Voter's Vote . . . 93
9.3.2 Attack Scenario peb.2: Changing a Careful Voter's Vote . . . 94
9.3.3 Attack Scenario peb.3: Canceling the Vote of a Fleeing Voter . . . 95
9.3.4 Attack Scenario peb.4: Canceling a Vote by Faking a Fleeing Voter . . . 95
9.3.5 Attack Scenario flash.1: iVotronic Denial-of-Service . . . 96
9.3.6 Attack Scenario flash.2: Voter Confusion . . . 96
9.3.7 Attack Scenario unity.1: Unrestricted Access to Unity Ballot Preparation Software . 97
9.3.8 Attack Scenario unity.2: Compromising the Unity Election Reporting Manager . . . 97
9.3.9 Attack Scenario m100.1: Changing the Firmware on the M100 Scanner . . . 97
9.3.10 Attack Scenario m100.2: M100 Denial-of-Service . . . 98
9.3.11 Attack Scenario virus.1: Compromising Entire Election Process with a Virus . . . 98
9.4 Potential Attack Scenarios . . . 99
9.4.1 Attack Scenario flash.3: iVotronic Exploits Using a Flash Card as Delivery Mechanism 99

III Analysis of the Premier Elections Solutions, Inc. Voting Systems 101

10 Premier Executive Summary 103

11 Premier Study Overview 105
11.1 Part Structure . . . 105
11.2 Architecture . . . 105
11.2.1 Components at County Election Headquarters . . . 106
11.2.2 Components at Polling Place . . . 108

11.3 Election Procedures . . . 109
11.3.1 Pre-Election Procedures . . . 109
11.3.2 Polling Place Election Procedures . . . 110
11.3.3 Post-Election Procedures . . . 110
11.4 Verdasys Digital Guardian . . . 111

12 Premier Systemic and Architectural Issues 113
12.1 Ineffective Data and Privacy Security . . . 114
12.1.1 Memory Cards . . . 114
12.1.2 Voter Privacy . . . 114
12.1.3 GEMS . . . 114
12.2 Ineffective Cryptographic and Hardware Security . . . 115
12.2.1 Key Management . . . 115
12.2.2 Password Management . . . 115
12.2.3 Hardware Tokens . . . 116
12.3 Unsafe Software Management . . . 116
12.3.1 Installation Procedures . . . 116
12.3.2 AccuBasic . . . 116
12.4 Design and Code Quality Problems . . . 117
12.4.1 Inadequate Security . . . 118
12.4.2 Improper Use of Security Technology . . . 118
12.4.3 Unsafe Programming Practices . . . 119
12.5 Previously Unreviewed Components . . . 120
12.5.1 EMP . . . 120
12.5.2 Digital Guardian . . . 120
12.5.3 ExpressPoll . . . 120
12.5.4 Voter Card Encoder . . . 121
12.6 Audit Procedures . . . 121

13 Premier Issue Confirmation 123
13.1 Premier GEMS Vulnerabilities . . . 123
13.1.1 GEMS uses the Microsoft Jet data layer . . . 123
13.1.2 GEMS databases can be modified with access to the local hard disk . . . 124
13.1.3 GEMS relies on the graphical user interface (GUI) to enforce security . . . 125
13.1.4 Third parties translation services may introduce a virus into the system . . . 126
13.1.5 Race and candidate labels may be changed after GEMS has been “set-for-election” . 126
13.1.6 GEMS fails to filter login field for special characters . . . 127
13.1.7 Unsafe handling of election database content may allow a virus to control GEMS . . 128
13.1.8 Election database passwords are stored with insufficient protection . . . 129
13.1.9 Poor handling of integers may cause GEMS to crash . . . 130
13.2 Premier AV-OS PC Vulnerabilities . . . 130
13.2.1 The AV-OS memory card data is not authenticated . . . 130
13.2.2 The GEMS server and AV-OS PC communication is not authenticated . . . 131
13.2.3 The memory card checksums do not protect against malicious tampering . . . 132
13.2.4 The audit log is unprotected and old entries are overwritten . . . 133
13.2.5 The memory card “signature” does not prevent malicious tampering . . . 133
13.2.6 Improper string handling can result in arbitrary code execution . . . 134
13.2.7 Candidate vote counters are not checked for overflow . . . 134

13.2.8 The supervisor "password" is stored with inadequate protection 135
13.2.9 Candidate ballot coordinates can be modified to manipulate election results 136
13.2.10 Flaws in the AccuBasic interpreter may allow a virus to control an AV-OS PC 137
13.2.11 Memory card attacks could be masked by modifying the zero report AccuBasic script 137
13.2.12 The AV-OS PC software controls the physical paper ballot box deflector 138
13.2.13 A voter can cause the AV-OS PC to stop accepting ballots 139
13.2.14 Memory card contents can be accessed from the serial port 139
13.2.15 The AV-OS PC accepts the same ballot multiple times 140
13.2.16 The AV-OS PC printer can be temporarily disabled without the supervisor password 140
13.3 Premier AV-TSX Vulnerabilities 141
13.3.1 The AV-TSX will install an unauthenticated bootloader and operating system 141
13.3.2 The AV-TSX will install unauthenticated application updates 141
13.3.3 There are multiple buffer overflow vulnerabilities in the handling of .ins files 142
13.3.4 An unauthenticated user can read and or tamper with the memory of the AV-TSX . . 143
13.3.5 The cryptographic keys are not adequately protected 143
13.3.6 Malicious software could alter files critical to correctly reporting an election 144
13.3.7 The smart card authentication protocol can easily be broken 145
13.3.8 Security Cards can be forged and used to change keys 146
13.3.9 The System Setup Menu is accessible without using a smart card 147
13.3.10 The protected counter is not protected 148
13.3.11 SSL certificates are not sufficiently protected 148
13.3.12 OpenSSL is not initialized with adequate entropy 149
13.3.13 Flaws in the AccuBasic interpreter may allow a virus to control an AV-TSX 150
13.3.14 Failure to check data on memory cards may allow a virus to run on the AV-TSX . . . 150
13.3.15 Unsafe handling of election resource files may allow a virus to control an AV-TSX . 151
13.3.16 Unsafe handling of election database files may allow a virus to control an AV-TSX . 152
13.3.17 A voter may be able to gain control of an AV-TSX 152
13.3.18 A malicious GEMS server can cause an AV-TSX to crash on download 153
13.3.19 Ballots are stored in the order in which they are cast 154
13.3.20 Cast ballots and VVPAT barcodes contain a timestamp 154
13.3.21 An attacker can reconstruct the order in which ballots were cast 155
13.3.22 The AV-TSX does not securely delete files 156
13.3.23 Logic errors in the bootloader create a vulnerability when loading bitmaps 156
13.3.24 There are a number of errors in the AV-TSX startup code 157

14 Premier New Issues **159**
14.1 Premier EMP Vulnerabilities 159
14.1.1 Tampering with a memory card allows attackers to crash or take control of an EMP server. 159
14.1.2 A single Data Key is used to encrypt all ballots in a county 160
14.1.3 The warning that a default key is in use is not sufficient 161
14.1.4 A malformed IP address can freeze the EMP server 162
14.1.5 The contents of the logs on the EMP server are not authenticated 162
14.1.6 The EMP server shares the same default SSL Certificate as the AV-TSX 163
14.1.7 The System Key for the EMP is insecure 164
14.1.8 The integrity of the Data Key is not protected 164
14.1.9 GEMS trusts the EMP to deliver correct ballot definitions and results 165
14.1.10 A malicious GEMS server can crash an EMP server 166

14.1.11 A compromised AV-TSX can crash an EMP server . . . 166
14.2 Premier General Vulnerabilities . . . 167
14.2.1 Premier may follow insecure media format procedures . . . 167
14.3 Premier GEMS Vulnerabilities . . . 167
14.3.1 Vulnerabilities in Microsoft Windows provided DLL files affect GEMS . . . 167
14.3.2 Many GEMS servers state-wide use the same BIOS password . . . 168
14.4 Premier AV-OS PC Vulnerabilities . . . 168
14.4.1 Forged ballots can be forced into the ballot box . . . 168
14.4.2 The AV-OS PC ballot box collecting votes allows vote order to be reconstructed . . . 169
14.4.3 The Hart ballot box key works in the Premier ballot box . . . 170
14.5 Premier VCEncoder Vulnerabilities . . . 171
14.5.1 Access to the Voter Card Encoder is not protected by a PIN . . . 171
14.5.2 Once the VCE is enabled, no mechanism prevents smart cards from being encoded . 171
14.5.3 Software can be loaded by anyone with access to the VCE . . . 172
14.5.4 The VCE accepts the default smart card key . . . 173
14.6 Premier ExpressPoll Vulnerabilities . . . 173
14.6.1 The ExpressPoll runs a webserver . . . 173
14.6.2 The ExpressPoll will install an unauthenticated bootloader and operating system . . 174
14.6.3 The ExpressPoll uses an unprotected database for poll data . . . 174
14.6.4 The ExpressPoll uses an unprotected resource file . . . 175
14.6.5 The ExpressPoll can be rendered useless in transit . . . 175
14.6.6 The ExpressPoll audit logs are not protected . . . 176
14.6.7 ExpressPoll audit logs violate voter privacy . . . 176
14.7 Premier Digital Guardian Vulnerabilities . . . 177
14.7.1 Users with administrative access can circumvent boot restrictions . . . 177
14.7.2 The *GEMSUser* account is in the *Administrators* group . . . 178
14.7.3 Digital Guardian can be disabled by booting from external media . . . 178
14.7.4 An administrative user can disable the Digital Guardian device drivers once . . . 179
14.7.5 Users with administrative access can circumvent many Digital Guardian controls . . 179
14.7.6 The Digital Guardian policy denies only specific known unwanted applications . . . 180
14.7.7 Digital Guardian execution restrictions are circumventable by copying files . . . 181
14.7.8 *GEMSUser* can use the CD burning application to modify the election database . . . 182
14.7.9 The election database protections only apply to files on the local hard drive . . . 182
14.7.10 Digital Guardian logging is disabled . . . 183
14.7.11 Digital Guardian does not immediately detect if GEMS is replaced . . . 183
14.8 Premier AV-TSX Vulnerabilities . . . 184
14.8.1 The wires connecting the VVPAT printer can be cut without opening the AV-TSX . . 184
14.8.2 The plastic housing protecting the printer can easily be removed . . . 184
14.8.3 An adversary can destroy paper copies of cast ballots without opening the AV-TSX . 185
14.8.4 The power button is accessible when all panels on the AV-TSX are closed and locked 187
14.8.5 The bootloader can be manipulated to give access to the file system on the AV-TSX . 187
14.8.6 The memory of an AV-TSX can be erased by a memory card . . . 188
14.8.7 A voter can gain administrative access to the AV-TSX . . . 189
14.8.8 A voter can cast an unlimited number of votes without any tools or knowledge . . . 189
14.8.9 The smart card reader can be jammed by an attacker . . . 190

15 Premier Combined Implications and Attack Scenarios **191**
15.1 Casting an Unlimited Number of Ballots . . . 191
15.2 Pre-Stuffing an Election . . . 192
15.3 Voting Machine Virus . . . 193
15.3.1 Infecting the GEMS Server . . . 193
15.3.2 Infecting the EMP Server . . . 194
15.3.3 Infecting the AV-TSX . . . 194
15.4 Denying Voters the Ability to Vote . . . 194

IV Analysis of the Hart InterCivic, Inc. Voting Systems **195**

16 Hart Executive Summary **197**

17 Hart Study Overview **199**
17.1 Part Structure . . . 199
17.2 Architecture . . . 199
17.2.1 Components at County Election Headquarters . . . 200
17.2.2 Components in Use at Polling Locations . . . 201
17.3 General Election Procedure . . . 202
17.3.1 Pre-Election Procedures . . . 202
17.3.2 Election Day Procedures . . . 203
17.3.3 Post-Election Procedures . . . 203
17.4 Data Security . . . 204
17.4.1 Other Components in the Hart System . . . 204

18 Hart Systemic and Architectural Issues **207**
18.1 Ineffective Data, Software, and Firmware Protections . . . 208
18.1.1 Data . . . 208
18.1.2 Communication . . . 208
18.1.3 Software and Firmware . . . 209
18.2 Ineffective Authentication . . . 209
18.3 Undocumented, Unprotected, and Unsafe Features . . . 210
18.3.1 Autovote . . . 210
18.3.2 Windows Registry Misuse . . . 210
18.3.3 Remote eScan access . . . 211
18.3.4 Adjust Vote Totals . . . 211
18.4 System Design, Development, and Maintenance Issues . . . 211
18.5 Audit Procedures . . . 212

19 Hart Issue Confirmation **215**
19.1 Hart General Vulnerabilities . . . 215
19.1.1 Corrupt MBBs can cause Tally to crash . . . 215
19.1.2 Database passwords are stored insecurely . . . 216
19.1.3 The EMS databases are not encrypted . . . 217
19.1.4 New Users can be inserted into the database . . . 217
19.1.5 Back-end Windows systems may be insecure . . . 218
19.1.6 Election management computers may be connected to the Internet . . . 219

19.1.7 eCM keys are extracted and stored insecurely . . . 221
19.1.8 Vote order can be determined from an MBB . . . 221
19.1.9 Voting and audit data not secured while voting . . . 222
19.1.10 The internal vote count on the eScan, eSlate, and JBC can be modified . . . 222
19.1.11 The EMS software uses a vulnerable version of OpenSSL . . . 223
19.1.12 Pervasive failure to follow safe programming practices . . . 224
19.2 Hart eCM Vulnerabilities . . . 225
19.2.1 The same symmetric eCM key is used county-wide . . . 225
19.2.2 eCM keys are stored insecurely on the eCM manager . . . 225
19.3 Hart SERVO Vulnerabilities . . . 226
19.3.1 Buffer overflows in SERVO . . . 226
19.4 Hart eScan Vulnerabilities . . . 227
19.4.1 The eScan is managed via an accessible Ethernet port . . . 227
19.5 Hart eSlate Vulnerabilities . . . 228
19.5.1 The eSlate is managed via a serial port connection to the JBC . . . 228
19.5.2 Communication between the eSlate and JBC is insecure . . . 228
19.5.3 Compromised eSlates can provide votes without voter records . . . 229
19.5.4 The periodic memory integrity checks used by eSlate and JBC are ineffective . . . 230
19.6 Hart Servo Vulnerabilities . . . 231
19.6.1 SERVO-based device firmware checking can be spoofed . . . 231
19.7 Hart JBC Vulnerabilities . . . 232
19.7.1 The JBC internal version checks can never fail. . . . 232
19.7.2 JBC-based eSlate firmware checking can be spoofed . . . 232
19.7.3 The JBC is managed via an accessible parallel port . . . 233
19.7.4 The JBC Voter Registration Interface can be used to generate voter access codes . . . 233
19.7.5 Format String vulnerabilities in the JBC report mode . . . 234
19.7.6 Voter codes are not selected randomly . . . 235
19.8 Hart VBO Vulnerabilities . . . 236
19.8.1 The VBO record is easily manipulatable by an eSlate program . . . 236
19.8.2 VBO printer allows the paper to be rewound . . . 236
19.8.3 VBO ballots are printed sequentially . . . 237
19.9 Hart Tally Vulnerabilities . . . 237
19.9.1 The Tally interface allows a Tally administrator to “adjust vote totals” . . . 237
19.9.2 Precinct IDs are improperly handled by the system . . . 238
19.9.3 Users can tally unclosed or corrupted MBBs . . . 238
19.9.4 MBBs are only processed if the Tally database allows it . . . 239
19.9.5 Rally and Tally allow a user to accept previously unrecognized certificates . . . 240
19.10 Hart Ballot Now Vulnerabilities . . . 240
19.10.1 Ballot Now ballot counters are stored in a database . . . 240

20 Hart New Issues **243**
20.1 Hart General Vulnerabilities . . . 243
20.1.1 An MBB image can be copied and restored without any credentials. . . . 243
20.1.2 EMS systems make improper use of the Windows registry . . . 244
20.1.3 Hart EMS passwords can be bypassed . . . 245
20.1.4 Hart EMS audit logs can be modified or erased . . . 246
20.2 Hart eCM Vulnerabilities . . . 247
20.2.1 eCM keys may be quietly recorded to a debug file . . . 247

20.3 Hart eScan Vulnerabilities . . . 248
20.3.1 The flash memory containing the eScan executable and file system can be replaced . 248
20.3.2 The eScan runs a telnet server . . . 248
20.3.3 The eScan scanner surface can be occluded to affect ballot processing . . . 249
20.3.4 The eScan ballot box collecting votes allows vote order to be reconstructed . . . 250
20.3.5 The eScan ballot box is vulnerable to attacks that may destroy ballots . . . 250
20.3.6 The eScan may be modified to allow casting of duplicate ballots . . . 251
20.3.7 The eScan has an open interface allowing erasure of vote and audit log records . . . 252
20.3.8 The Premier ballot box key works in the Hart ballot box . . . 253
20.3.9 A voter can cast a ballot and then retrieve it from the ballot box . . . 253
20.4 Hart JBC Vulnerabilities . . . 254
20.4.1 The JBC can rapidly create an unlimited number of access codes on election day . . 254
20.4.2 The JBC has an open interface allowing erasure of vote and audit log records . . . 255
20.4.3 JBC/eSlate voting can be completely automated . . . 256
20.5 Hart VBO Vulnerabilities . . . 257
20.5.1 The VBO Printer is controlled via an accessible 1/8” port . . . 257
20.5.2 The VBO printer can be disabled by cutting the wires to it . . . 258
20.5.3 Damaging contacts on the eSlate booth can cause the eSlate-JBC connection to fail . 259
20.5.4 The paper roll in the VBO can be tampered with or removed . . . 259
20.5.5 The serial number of the VBO can be modified . . . 260
20.5.6 The VVPAT paper record may be forged . . . 261
20.6 Hart Tally Vulnerabilities . . . 262
20.6.1 Tallied MBBs can be used in election . . . 262
20.6.2 The Tally user interface is completely configured in the registry . . . 263
20.7 Hart Ballot Now Vulnerabilities . . . 264
20.7.1 The Ballot Now username and password can be bypassed . . . 264
20.7.2 Ballot Now hidden menu options are controlled by registry entries . . . 264

21 Hart Combined Implications and Attack Scenarios 267
21.1 Voting Machine Viruses . . . 267
21.2 Automated and Mass Voting in a Precinct . . . 267
21.3 Forging Entire DRE Precinct Results . . . 268
21.4 Vote Adjustments at Election Headquarters . . . 269
21.4.1 Insider Attack . . . 269
21.4.2 Outsider Attack . . . 269
21.5 Attacks That Delay Elections . . . 270
21.6 Denial of service or destruction of election results in a polling place . . . 270

V Appendices 271

22 ES&S Private Report 273

23 Premier Equipment 279

24 Premier Private Report - Issue Confirmation **283**
24.1 Premier GEMS Vulnerabilities . . . 283
24.1.1 GEMS uses the Microsoft Jet data layer . . . 283
24.1.2 GEMS databases can be modified with access to the local hard disk . . . 283
24.1.3 GEMS relies on the graphical user interface (GUI) to enforce security . . . 283
24.1.4 Third parties translation services may introduce a virus into the system . . . 283
24.1.5 Race and candidate labels may be changed after GEMS has been "set-for-election" . . . 283
24.1.6 GEMS fails to filter login field for special characters . . . 284
24.1.7 Unsafe handling of election database content may allow a virus to control GEMS . . . 284
24.1.8 Election database passwords are stored with insufficient protection . . . 284
24.1.9 Poor handling of integers may cause GEMS to crash . . . 284
24.2 Premier AV-OS PC Vulnerabilities . . . 284
24.2.1 The AV-OS memory card data is not authenticated . . . 284
24.2.2 The GEMS server and AV-OS PC communication is not authenticated . . . 284
24.2.3 The memory card checksums do not protect against malicious tampering . . . 284
24.2.4 The audit log is unprotected and old entries are overwritten . . . 284
24.2.5 The memory card "signature" does not prevent malicious tampering . . . 285
24.2.6 Improper string handling can result in arbitrary code execution . . . 285
24.2.7 Candidate vote counters are not checked for overflow . . . 285
24.2.8 The supervisor "password" is stored with inadequate protection . . . 285
24.2.9 Candidate ballot coordinates can be modified to manipulate election results . . . 285
24.2.10 Flaws in the AccuBasic interpreter may allow a virus to control an AV-OS PC . . . 285
24.2.11 Memory card attacks could be masked by modifying the zero report AccuBasic script 285
24.2.12 The AV-OS PC software controls the physical paper ballot box deflector . . . 285
24.2.13 A voter can cause the AV-OS PC to stop accepting ballots . . . 286
24.2.14 Memory card contents can be accessed from the serial port . . . 286
24.2.15 The AV-OS PC accepts the same ballot multiple times . . . 286
24.2.16 The AV-OS PC printer can be temporarily disabled without the supervisor password 286
24.3 Premier AV-TSX Vulnerabilities . . . 286
24.3.1 The AV-TSX will install an unauthenticated bootloader and operating system . . . 286
24.3.2 The AV-TSX will install unauthenticated application updates . . . 286
24.3.3 There are multiple buffer overflow vulnerabilities in the handling of .ins files . . . 286
24.3.4 An unauthenticated user can read and or tamper with the memory of the AV-TSX . . . 287
24.3.5 The cryptographic keys are not adequately protected . . . 287
24.3.6 Malicious software could alter files critical to correctly reporting an election . . . 287
24.3.7 The smart card authentication protocol can easily be broken . . . 287
24.3.8 Security Cards can be forged and used to change keys . . . 287
24.3.9 The System Setup Menu is accessible without using a smart card . . . 287
24.3.10 The protected counter is not protected . . . 287
24.3.11 SSL certificates are not sufficiently protected . . . 287
24.3.12 OpenSSL is not initialized with adequate entropy . . . 288
24.3.13 Flaws in the AccuBasic interpreter may allow a virus to control an AV-TSX . . . 288
24.3.14 Failure to check data on memory cards may allow a virus to run on the AV-TSX . . . 288
24.3.15 Unsafe handling of election resource files may allow a virus to control an AV-TSX . . . 288
24.3.16 Unsafe handling of election database files may allow a virus to control an AV-TSX . . . 288
24.3.17 A voter may be able to gain control of an AV-TSX . . . 288
24.3.18 A malicious GEMS server can cause an AV-TSX to crash on download . . . 288
24.3.19 Ballots are stored in the order in which they are cast . . . 288

24.3.20 Cast ballots and VVPAT barcodes contain a timestamp . . . 289
24.3.21 An attacker can reconstruct the order in which ballots were cast . . . 289
24.3.22 The AV-TSX does not securely delete files . . . 289
24.3.23 Logic errors in the bootloader create a vulnerability when loading bitmaps . . . 289
24.3.24 There are a number of errors in the AV-TSX startup code . . . 289

25 Premier Private Report - New Issues 291
25.1 Premier EMP Vulnerabilities . . . 291
25.1.1 Tampering with a memory card allows attackers to crash or take control of an EMP server. . . . 291
25.1.2 A single Data Key is used to encrypt all ballots in a county . . . 291
25.1.3 The warning that a default key is in use is not sufficient . . . 291
25.1.4 A malformed IP address can freeze the EMP server . . . 291
25.1.5 The contents of the logs on the EMP server are not authenticated . . . 291
25.1.6 The EMP server shares the same default SSL Certificate as the AV-TSX . . . 292
25.1.7 The System Key for the EMP is insecure . . . 292
25.1.8 The integrity of the Data Key is not protected . . . 292
25.1.9 GEMS trusts the EMP to deliver correct ballot definitions and results . . . 292
25.1.10 A malicious GEMS server can crash an EMP server . . . 292
25.1.11 A compromised AV-TSX can crash an EMP server . . . 292
25.2 Premier General Vulnerabilities . . . 292
25.2.1 Premier may follow insecure media format procedures . . . 292
25.3 Premier GEMS Vulnerabilities . . . 293
25.3.1 Vulnerabilities in Microsoft Windows provided DLL files affect GEMS . . . 293
25.3.2 Many GEMS servers state-wide use the same BIOS password . . . 293
25.4 Premier AV-OS PC Vulnerabilities . . . 293
25.4.1 Forged ballots can be forced into the ballot box . . . 293
25.4.2 The AV-OS PC ballot box collecting votes allows vote order to be reconstructed . . . 293
25.4.3 The Hart ballot box key works in the Premier ballot box . . . 293
25.5 Premier VCEncoder Vulnerabilities . . . 293
25.5.1 Access to the Voter Card Encoder is not protected by a PIN . . . 293
25.5.2 Once the VCE is enabled, no mechanism prevents smart cards from being encoded . 293
25.5.3 Software can be loaded by anyone with access to the VCE . . . 294
25.5.4 The VCE accepts the default smart card key . . . 294
25.6 Premier ExpressPoll Vulnerabilities . . . 294
25.6.1 The ExpressPoll runs a webserver . . . 294
25.6.2 The ExpressPoll will install an unauthenticated bootloader and operating system . . 294
25.6.3 The ExpressPoll uses an unprotected database for poll data . . . 294
25.6.4 The ExpressPoll uses an unprotected resource file . . . 294
25.6.5 The ExpressPoll can be rendered useless in transit . . . 294
25.6.6 The ExpressPoll audit logs are not protected . . . 294
25.6.7 ExpressPoll audit logs violate voter privacy . . . 295
25.7 Premier Digital Guardian Vulnerabilities . . . 295
25.7.1 Users with administrative access can circumvent boot restrictions . . . 295
25.7.2 The *GEMSUser* account is in the *Administrators* group . . . 295
25.7.3 Digital Guardian can be disabled by booting from external media . . . 295
25.7.4 An administrative user can disable the Digital Guardian device drivers once . . . 295
25.7.5 Users with administrative access can circumvent many Digital Guardian controls . . 295

25.7.6 The Digital Guardian policy denies only specific known unwanted applications . . . 295
25.7.7 Digital Guardian execution restrictions are circumventable by copying files 295
25.7.8 *GEMSUser* can use the CD burning application to modify the election database . . . 296
25.7.9 The election database protections only apply to files on the local hard drive 296
25.7.10 Digital Guardian logging is disabled 296
25.7.11 Digital Guardian does not immediately detect if GEMS is replaced 296
25.8 Premier AV-TSX Vulnerabilities 296
25.8.1 The wires connecting the VVPAT printer can be cut without opening the AV-TSX . . 296
25.8.2 The plastic housing protecting the printer can easily be removed 296
25.8.3 An adversary can destroy paper copies of cast ballots without opening the AV-TSX . 296
25.8.4 The power button is accessible when all panels on the AV-TSX are closed and locked 296
25.8.5 The bootloader can be manipulated to give access to the file system on the AV-TSX . 297
25.8.6 The memory of an AV-TSX can be erased by a memory card 297
25.8.7 A voter can gain administrative access to the AV-TSX 297
25.8.8 A voter can cast an unlimited number of votes without any tools or knowledge . . . 297
25.8.9 The smart card reader can be jammed by an attacker 297

26 Hart Equipment 299

27 Hart Private Report - Issue Confirmation 303
27.1 Hart General Vulnerabilities 303
27.1.1 Corrupt MBBs can cause Tally to crash 303
27.1.2 Database passwords are stored insecurely 303
27.1.3 The EMS databases are not encrypted 303
27.1.4 New Users can be inserted into the database 303
27.1.5 Back-end Windows systems may be insecure 303
27.1.6 eCM keys are extracted and stored insecurely 304
27.1.7 Vote order can be determined from an MBB 304
27.1.8 Voting and audit data not secured while voting 304
27.1.9 The internal vote count on the eScan, eSlate, and JBC can be modified 304
27.1.10 The EMS software uses a vulnerable version of OpenSSL 304
27.1.11 Pervasive failure to follow safe programming practices 304
27.2 Hart eCM Vulnerabilities 304
27.2.1 eCM keys are stored insecurely on the eCM manager 304
27.3 Hart SERVO Vulnerabilities 305
27.3.1 Buffer overflows in SERVO 305
27.4 Hart eScan Vulnerabilities 305
27.4.1 The eScan is managed via an accessible Ethernet port 305
27.5 Hart eSlate Vulnerabilities 305
27.5.1 The eSlate is managed via a serial port connection to the JBC 305
27.5.2 Communication between the eSlate and JBC is insecure 305
27.5.3 Compromised eSlates can provide votes without voter records 305
27.5.4 The periodic memory integrity checks used by eSlate and JBC are ineffective 305
27.6 Hart Servo Vulnerabilities 306
27.6.1 SERVO-based device firmware checking can be spoofed 306
27.7 Hart JBC Vulnerabilities 306
27.7.1 The JBC internal version checks can never fail. 306
27.7.2 JBC-based eSlate firmware checking can be spoofed 306

27.7.3 The JBC is managed via an accessible parallel port . . . 306
27.7.4 The JBC Voter Registration Interface can be used to generate voter access codes . . 306
27.7.5 Format String vulnerabilities in the JBC report mode . . . 306
27.7.6 Voter codes are not selected randomly . . . 306
27.8 Hart VBO Vulnerabilities . . . 307
27.8.1 The VBO record is easily manipulatable by an eSlate program . . . 307
27.8.2 VBO printer allows the paper to be rewound . . . 307
27.8.3 VBO ballots are printed sequentially . . . 307
27.9 Hart Tally Vulnerabilities . . . 307
27.9.1 The Tally interface allows a Tally administrator to “adjust vote totals” . . . 307
27.9.2 Precinct IDs are improperly handled by the system . . . 307
27.9.3 Users can tally unclosed or corrupted MBBs . . . 307
27.9.4 MBBs are only processed if the Tally database allows it . . . 307
27.9.5 Rally and Tally allow a user to accept previously unrecognized certificates . . . 308
27.10 Hart Ballot Now Vulnerabilities . . . 308
27.10.1 Ballot Now ballot counters are stored in a database . . . 308

28 Hart Private Report - New Issues 309
28.1 Hart General Vulnerabilities . . . 309
28.1.1 An MBB image can be copied and restored without any credentials. . . . 309
28.1.2 EMS systems make improper use of the Windows registry . . . 309
28.1.3 Hart EMS passwords can be bypassed . . . 309
28.1.4 Hart EMS audit logs can be modified or erased . . . 309
28.2 Hart eCM Vulnerabilities . . . 309
28.2.1 eCM keys may be quietly recorded to a debug file . . . 309
28.3 Hart eScan Vulnerabilities . . . 310
28.3.1 The flash memory containing the eScan executable and file system can be replaced . 310
28.3.2 The eScan runs a telnet server . . . 310
28.3.3 The eScan scanner surface can be occluded to affect ballot processing . . . 310
28.3.4 The eScan ballot box collecting votes allows vote order to be reconstructed . . . 310
28.3.5 The eScan ballot box is vulnerable to attacks that may destroy ballots . . . 310
28.3.6 The eScan may be modified to allow casting of duplicate ballots . . . 310
28.3.7 The eScan has an open interface allowing erasure of vote and audit log records . . . 310
28.3.8 The Premier ballot box key works in the Hart ballot box . . . 310
28.3.9 New Vulnerability 09 . . . 311
28.4 Hart JBC Vulnerabilities . . . 311
28.4.1 The JBC can rapidly create an unlimited number of access codes on election day . . 311
28.4.2 The JBC has an open interface allowing erasure of vote and audit log records . . . 311
28.4.3 JBC/eSlate voting can be completely automated . . . 311
28.5 Hart VBO Vulnerabilities . . . 311
28.5.1 The VBO Printer is controlled via an accessible 1/8” port . . . 311
28.5.2 The serial number of the VBO can be modified . . . 311
28.5.3 The VVPAT paper record may be forged . . . 311
28.6 Hart Tally Vulnerabilities . . . 312
28.6.1 Tallied MBBs can be used in election . . . 312
28.6.2 The Tally user interface is completely configured in the registry . . . 312
28.7 Hart Ballot Now Vulnerabilities . . . 312
28.7.1 The Ballot Now username and password can be bypassed . . . 312

28.7.2 Ballot Now hidden menu options are controlled by registry entries 312

29 Provided Source Code and Equipment **313**

Part I

Project Overview

CHAPTER 1

EXECUTIVE SUMMARY

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

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

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

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

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

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

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

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

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

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

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

CHAPTER 2

OVERVIEW

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

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

2.1 Report Structure

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

2.1.1 Reading Recommendations

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

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

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

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

2.2 Evaluation Design

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

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

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

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

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

2.3 Methodology

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

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

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

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

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

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

2.3.1 Prior Studies

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

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

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

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

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

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

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

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

2.4 Limitations

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  Hart InterCivic provided no access to the unredacted reports.

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

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

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

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

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

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

2.5 Acknowledgments

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

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

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

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

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

CHAPTER 3

THREAT MODEL

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

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

3.1 Reference Model

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

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

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

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

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

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

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

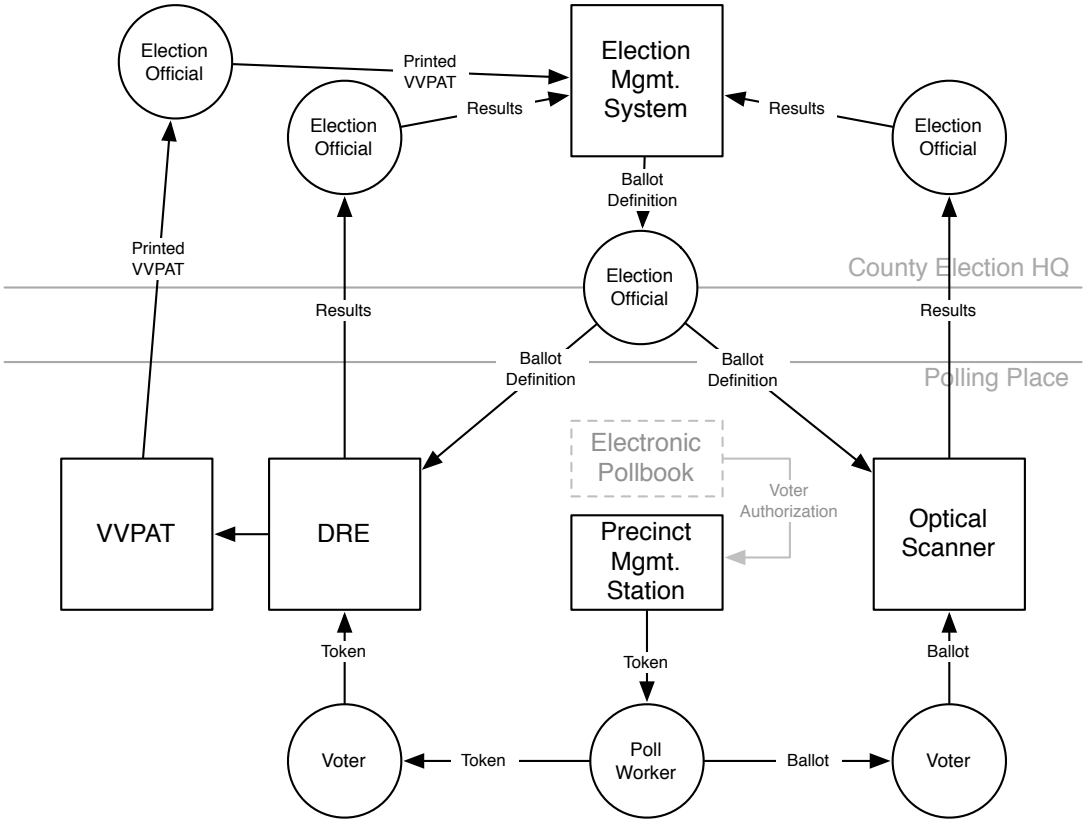


Figure 3.1: Reference architecture

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

The relationship between these components is shown in Figure 3.1.

3.1.1 Pre-Voting

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

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

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

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

3.1.2 Voting

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

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

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

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

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

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

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

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

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

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

3.1.3 Post-Voting

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

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

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

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

3.2 Attacker Goals

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

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

- Violate the secrecy of the ballot.

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

3.2.1 Producing Incorrect Vote Counts

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

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

Which attacks are feasible depends on the attacker capabilities.

3.2.2 Blocking Some or All Voters from Voting

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

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

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

3.2.3 Cast Doubt on the Legitimacy of the Election Results

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

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

- Falsify seemingly legitimate, but uncounted, ballot material.

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

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

3.2.4 Delay the Results of the Election from Becoming Known

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

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

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

3.2.5 Violating Ballot Secrecy

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

- Vote buying / voter coercion
- Information gathering

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

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

3.3 Potential Attackers

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

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

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

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

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

3.3.1 Outsiders

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

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

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

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

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

3.3.2 Voters

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

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

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

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

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

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

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

3.3.3 Poll Workers

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

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

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

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

3.3.4 Election Officials

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

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

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

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

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

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

3.3.5 Vendor Employees

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

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

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

3.4 Types of Attacks

We can categorize attacks along several dimensions:

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

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

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

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

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

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

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

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

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

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

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

3.5 Procedures

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

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

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

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

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

3.5.1 Ohio Procedures on Information Technology and Networking

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

- **General Internet Access and Networking**

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

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

- **Downloading Software**

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

- **Modem Access**

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

3.5.2 Mechanisms for Tamper Sealing

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

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

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

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

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

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

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

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

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

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

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

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

3.5.3 Chain-of-Custody Logs

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

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

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

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

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

3.5.4 Recounting Optical Scan Ballots

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

3.6 Software Engineering and Maintenance

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

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

Part II

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

CHAPTER 4

ES&S EXECUTIVE SUMMARY

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

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

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

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

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

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

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

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

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

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

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

CHAPTER 5

ES&S SYSTEM OVERVIEW

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

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

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

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

5.1 Architectural Overview

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

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

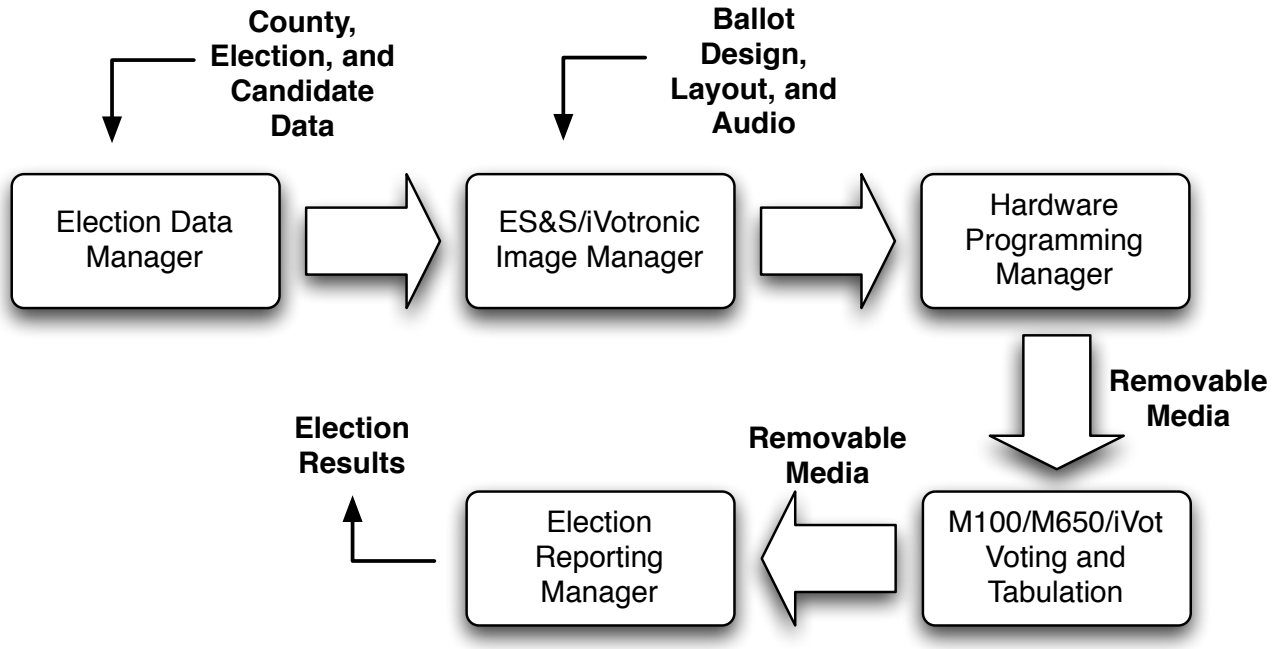


Figure 5.1: The high level overview of the ES&S Unity voting system with user input to each stage of the election.

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

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

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

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

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

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

5.2 System Components

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

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

5.2.1 Unity

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

Election Data Manager

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

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

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

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

ES&S Image Manager

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

iVotronic Image Manager

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

Hardware Programming Manager

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

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

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

Election Reporting Manager

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

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

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

Audit Manager

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

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

AutoMARK Information Management System

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

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

5.2.2 iVotronic

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

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

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

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

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

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

iVotronic Supervisor Terminal

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

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

iVotronic Voter Terminal

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

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

5.2.3 Real-Time Audit Log Printer

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

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

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

5.2.4 Personalized Electronic Ballot

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

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

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

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

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

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

5.2.5 PEB Reader

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

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

5.2.6 Compact Flash Cards

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

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

5.2.7 Communication Pack

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

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

5.2.8 Model 100

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

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

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

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

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

5.2.9 PCMCIA Memory Cards

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

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

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

5.2.10 Model 650

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

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

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

5.2.11 Zip Disks

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

5.2.12 AutoMARK Voter Assist Terminal

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

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

5.3 Election Procedures

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

5.3.1 Preparation

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

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

5.3.2 Touchscreen (DRE) Voting

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

Preparation

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

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

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

The iVotronic allows the following election tests:

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

Election Day

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

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

Vote Tallying

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

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

5.3.3 Precinct-Based Optical Scan Voting

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

Preparation

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

Election Day

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

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

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

Vote Tallying

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

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

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

5.3.4 Centrally Counted Optical Scan Voting

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

Preparation

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

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

Election Day Vote Tallying

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

5.4 Software Versions

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

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

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

5.5 Unity Acronyms and File Extensions

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

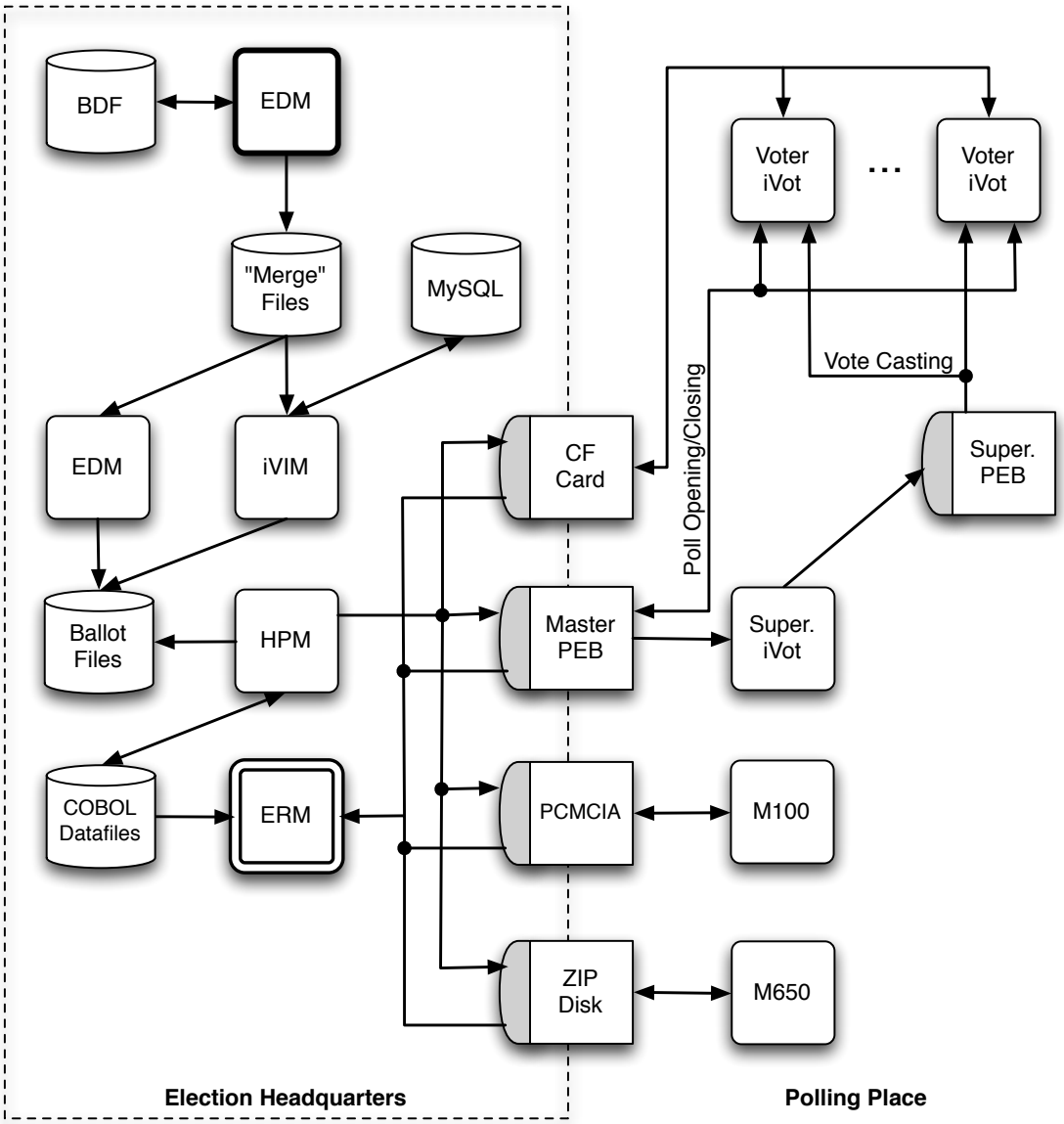


Figure 5.2: The major components and the flow of election data for counties using the ES&S Unity voting system. Arrows indicate direction of data flow. An election begins with configuration using the Election Data Manager (EDM), and ends once results are tallied and reported in the Election Reporting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

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

Figure 5.4: A red Supervisor PEB

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

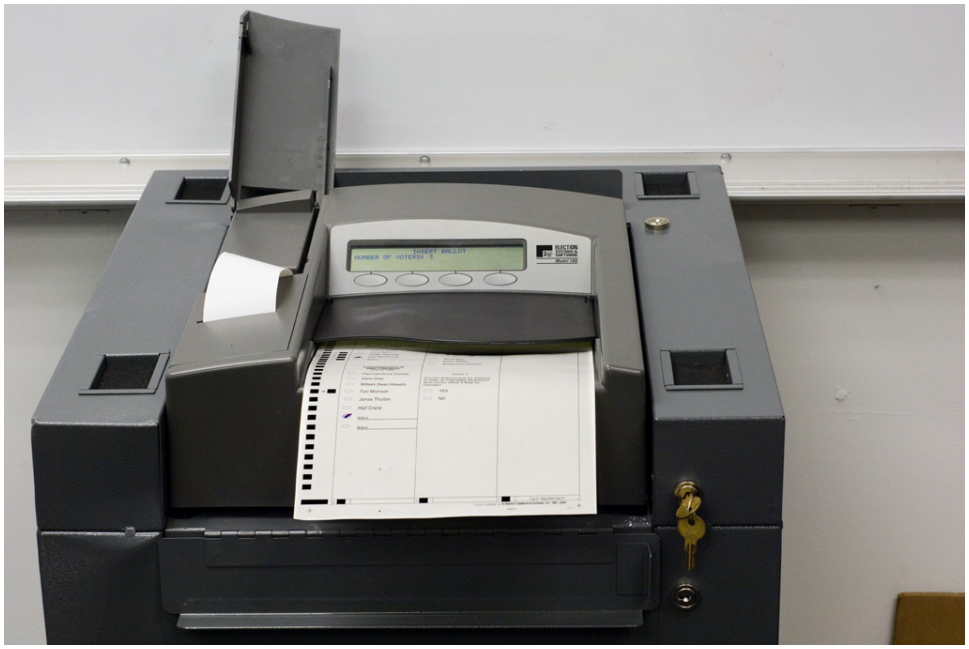


Figure 5.6: The ES&S M100 precinct optical scan ballot counter

Figure 5.7: A PCMCIA SRAM flash card

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

CHAPTER 6

ES&S SYSTEMIC AND ARCHITECTURAL ISSUES

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

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

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

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

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

6.1 Ineffective Access Control

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

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

6.1.1 iVotronic passwords and PEB-based access controls

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

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

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

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

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

Unauthorized screen calibration and configuration

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

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

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

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

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

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

Undocumented PEB features can be used to bypass password checks

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

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

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

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

Unauthorized PEB copying and alteration

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

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

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

6.1.2 Physical security, locks and seals

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

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

iVotronic

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

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

Locks and seals

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

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

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

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

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

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

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

6.2 Critical Errors in Input Processing

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

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

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

6.2.1 Unity

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

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

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

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

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

and M100-based systems.

6.2.2 iVotronic

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

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

6.3 Ineffectively Protected Software and Firmware

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

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

6.3.1 iVotronic firmware

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

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

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

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

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

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

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

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

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

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

6.3.2 Unity software

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

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

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

Attacks via input from precincts

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

Windows environment vulnerabilities

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

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

6.3.3 M100 firmware

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

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

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

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

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

6.3.4 Viral propagation

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

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

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

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

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

6.4 Ineffective Cryptography and Data Authentication

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

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

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

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

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

6.4.1 Unauthenticated M100 data

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

6.4.2 Ineffective iVotronic cryptography

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

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

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

6.4.3 Poor data validation of precinct results in Unity

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

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

6.5 Procedural Mitigations

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

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

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

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

CHAPTER 7

ES&S SPECIFIC WEAKNESSES AND THEIR IMPLICATIONS

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

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

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

7.1 Unity

7.1.1 Unity ERM buffer overflow when reading a Master PEB

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

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

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

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

7.1.2 Data from M100 can cause a buffer overflow in Unity

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

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

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

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

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

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

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

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

7.1.3 Unity accepts memory media with unauthorized precinct results

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

[1] See Section 22.7.1.2 in the confidential Annex.

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

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

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

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

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

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

7.1.4 Integrity of data on a PCMCIA card is not protected

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

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

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

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

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

[2] See Section 22.7.1.5 in the confidential Annex.

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

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

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

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

7.1.6 Unity decrypts a PEB using the EQC from the PEB

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

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

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

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

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

7.1.7 Unity contains large pieces of duplicated code

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

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

[3] See Section 22.7.1.6 in the confidential Annex.
[4] See Section 22.7.1.7 in the confidential Annex.

7.1.8 Unity contains many small buffer overflows

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

7.1.11 Assumptions on the environment for Unity are unspecified

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

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

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

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

7.1.12 iVotronic Image Manager bundled with vulnerable Java Runtime Environment

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

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

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

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

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

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

7.2 iVotronic

7.2.1 Election encryption/decryption key is recoverable from a PEB

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

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

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

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

7.2.2 PEBs may be emulated using infrared-capable devices

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

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

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

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

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

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

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

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

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

7.2.4 Emulated PEBs permit multiple votes

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

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

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

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

7.2.5 Buffer overflow in poll opening process is exploitable

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

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

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

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

7.2.6 Buffer overflow in "Hotspot" image processing is exploitable

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

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

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

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

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

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

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

7.2.7 Buffer overflow in Supervisor iVotronic initialization process is exploitable

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

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

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

[11] See Section 22.7.2.7 in the confidential Annex.

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

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

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

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

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

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

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

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

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

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

7.2.10 (Emulated) Factory QA PEBs bypass all password checks

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

[12] See Section 22.7.2.10 in the confidential Annex.

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

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

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

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

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

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

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

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

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

7.2.12 (Emulated) Factory QA PEB can lock a terminal

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

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

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

7.2.13 iVotronic touchscreens can be miscalibrated to deny certain ballot selections

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

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

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

7.2.14 Connection to RTAL/VVPAT printer is physically unprotected

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

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

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

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

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

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

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

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

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

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

7.2.15 Audit data is insufficiently randomized

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

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

Prerequisites: Access to audit information is required.

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

7.2.16 VVPAT barcodes contain timestamps

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

Prerequisites: Access to VVPAT records is required.

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

7.2.17 VVPAT barcodes contain vote information

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

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

[14] See Section 22.7.2.15 in the confidential Annex.

[15] See Section 22.7.2.16 in the confidential Annex.

[16] See Section 22.7.2.17 in the confidential Annex.

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

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

Figure 7.3: A VVPAT barcode encoding voter selections

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

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

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

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

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

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

7.2.18 Accuracy testing mode detectable

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

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

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

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

7.3 M100

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

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

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

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

Possible attacks include:

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

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

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

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

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

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

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

Prerequisites: A deliberately altered PCMCIA card

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

7.3.4 The M100 locks are easily manipulated

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

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

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

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

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

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

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

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

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

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

7.3.6 CRC routines used on the M100 are easily derivable

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

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

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

7.3.7 The M100 PCMCIA cards do not contain cryptographic integrity checks

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

Prerequisites: Knowledge of the CRCs as in Section 7.3.6.

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

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

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

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

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

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

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

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

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

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

7.3.9 M100 accepts counterfeit ballots

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

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

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

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

7.4 M650

7.4.1 M650 runs executable on Zip Disk on power up

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

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

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

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

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

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

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

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

[21] See Section 22.7.4.1 in the confidential Annex.

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

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

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

7.4.3 M650 fails to validate variable-length strings

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

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

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

[23] See Section 22.7.4.3 in the confidential annex.

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

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

7.4.4 M650 fails to protect against integer overflows

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

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

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

7.4.5 M650 accepts counterfeit ballots

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

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

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

[25] See Section 22.7.4.4 in the confidential Annex.

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

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

CHAPTER 8

ES&S SOFTWARE ENGINEERING ISSUES

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

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

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

8.1 Complexity

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

8.1.1 Programming Languages

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

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

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

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

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

8.1.2 Third-Party Software

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

8.1.3 Insufficient Documentation

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

- Defining precincts, districts, and polling places

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

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

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

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

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

8.2 Improper Message Passing

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

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

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

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

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

8.3 Static Code Analysis

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

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

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

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

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

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

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

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

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

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

CHAPTER 9

ES&S EXAMPLE ATTACK SCENARIOS

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

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

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

9.1 Tools

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

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

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

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

9.1.1 A framework for delivering a malicious payload to iVotronic systems

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

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

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

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

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

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

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

9.1.2 Pebserial: a tool for reading and writing PEBs

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

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

- it writes blocks of data to the PEB.

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

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

9.1.3 The iVotronic Serial Debugger

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

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

9.1.4 A tool to read and write M100 PCMCIA memory cards

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

9.1.5 The JTAG hardware debugger

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

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

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

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

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

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

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

9.1.6 A tool for extracting the QNX filesystem from the M100

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

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

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

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

9.2 Physical Security and Security Seals

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

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

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

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

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

9.3 Successful Attack Scenarios

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

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

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

as presented in Section 9.1.1.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

9.3.6 Attack Scenario flash.2: Voter Confusion

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

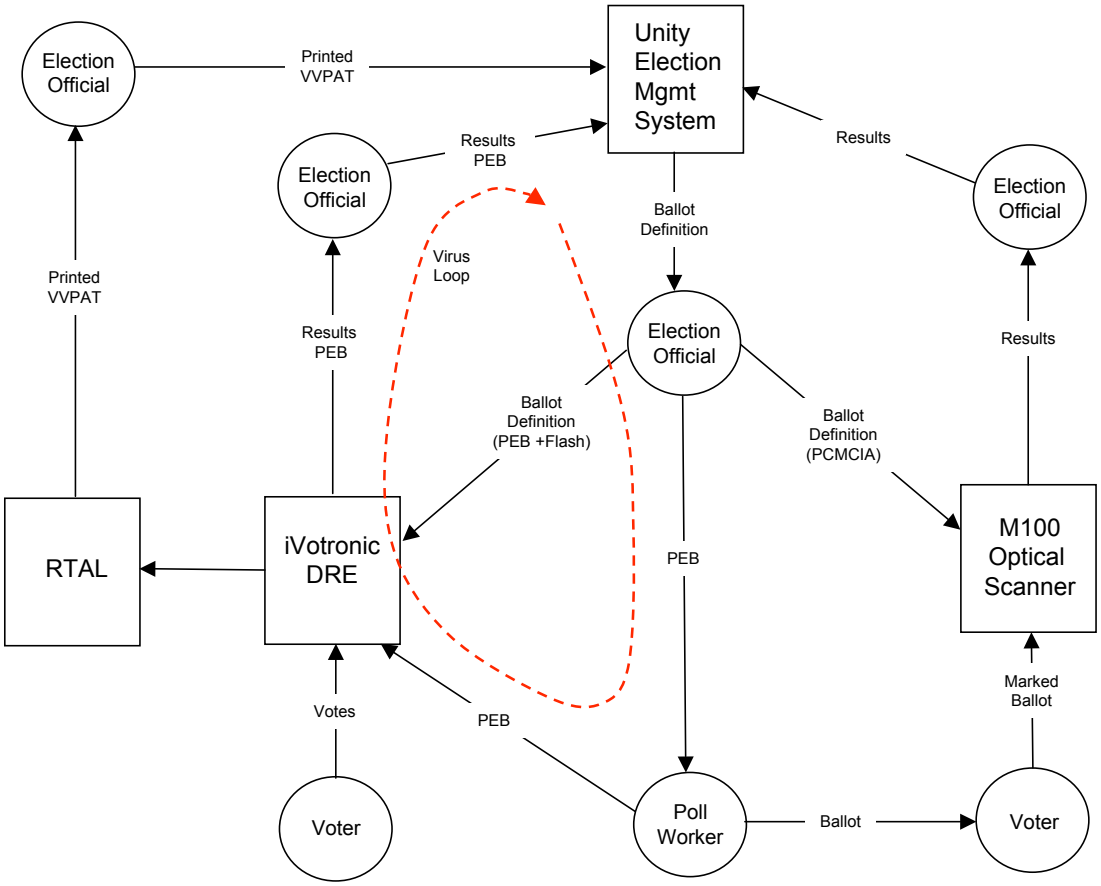


Figure 9.1: ES&S Architecture and Virus Loop

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

9.4 Potential Attack Scenarios

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

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