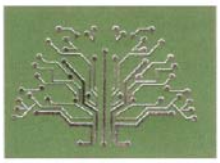

FREEMAN, CRAFT, MCGREGOR GROUP

Software Testing

Verity Voting System 3.0

Report Date: 2018-09-07

Version: 2.0

Status: FINAL

Classification: Public

atsec information security corporation
9130 Jollyville Road, Suite 260
Austin, TX 78759
Tel: +1 512 615 7300
Fax: +1 512 615 7301
www.atsec.com

Revision history

Version	Change date	Author(s)	Changes to previous version
1.0	2018-07-09	Ryan Hill	Initial draft
1.1	2018-07-27	Ryan Hill	Second draft
2.0	2018-09-07	Ryan Hill	Updated based on customer feedback.

Trademarks

Table of Contents

List of Tables

1 Executive Summary

This report was prepared by atsec information security corporation to review aspects of the security and integrity of the Verity Voting System v. 3.0. atsec is an independent, third-party company providing information-security assurance related services.

This report identifies security weaknesses and vulnerabilities found through static code review and by searches of public vulnerability sources. The search focused particularly on those that could be exploited to alter vote recording, vote results, critical election data such as audit logs, or to conduct a denial of service attack on the voting system.

It should be noted that the public vulnerability search is most likely to identify vulnerabilities that have been reported in commonly used commercial off the shelf system components.

The static code analysis revealed 10 issues, the public vulnerability search identified 9 vulnerabilities that could potentially be used for an attack on the voting system. Of the 10 issues found by static code analysis, 5 were assessed to be of medium severity and 5 were assessed to be of low severity.

At a high level, weaknesses and vulnerabilities were identified that can be attributed to difficulties resulting from an aging and repeatedly modified system. These include the following.

- Use of old 3rd party code containing publicly known vulnerabilities
- Instances of incorrect error or exception handling
- Use of outdated crypto algorithms and key lengths which become more susceptible to attack over time
- Hard coded passwords and unenforced password complexity rules
- Poor password storage methods
- Inclusion of code that is no longer used
- Possibly weak or duplicated initialization vectors
- SQL query best practices are not followed
- Instances of poor audit logging
- Inconsistent design such as varying requirements for password strength

In addition, numerous less severe but still noteworthy vulnerabilities were found related to code quality and non-conformance to the California Voting System Standards. See section 4 for all findings.

2 Introduction

This report was prepared by atsec information security corporation to review aspects of the security and integrity of the Verity Voting System v3.0. It has been prepared in support of a contract awarded to Freeman, Craft, McGregor Group, Inc. This project has a goal to provide voting system test support services to assist the California Secretary of State (SOS) with the evaluation of the Verity Voting System v3.0 for its suitability for use in the State of California in accordance with Elections Code sections 19001 et seq.

The source code review was performed by the following atsec information security corporation consultants.

- Fiona Pattinson (Project Manager)
- King Ables (Lead Reviewer)
- Jason Gorgeoulis (Reviewer)
- Demetrius Kellum (Reviewer)
- Sean Lewis (Reviewer)
- Dick Sikkema (Reviewer)
- Ryan Hill (Documentation Specialist)

This document identifies the security vulnerabilities found through static code review and by searches of public vulnerability sources that could be exploited to alter vote recording, vote results, critical election data, such as audit logs, or to conduct a denial of service attack on the voting system.

2.1 Scope and Basis

The Verity Voting System (EVS) v3.0 (hereafter referred to as the “voting system” or simply as the “system”) is a paper-based voting system made up of software, hardware, device, and peripheral components.

The system has the following software components:

- Verity Data—Ballot design software
- Verity Build—Election definition and media creation/ballot printing software
- Verity Central—Central ballot scanning and adjudication software
- Verity Count—Ballot tabulation and reporting software

The system can be set up to support one or more of the following hardware components:

- Verity Print—On-demand ballot printing device
- Verity Touch Writer—Accessible ballot marking device
- Verity Reader—Optional ballot verification device
- Verity Scan—Ballot scanning device
- Verity vDrive—Specially formatted USB media used for data transfers
- Verity Access—Accessibility device providing additional input options
- Verity Key—USB security key

atsec performed the source code review on the basis of an Agreement between Freeman, Craft, McGregor Group Inc., with the State of California, which states that the source code review includes examining the system in a manner that will provide the California Secretary of State with a basis for evaluating the extent to which the source code meets applicable standards. The threat model included in the Agreement is reproduced below and defines the threat parameters for the scope of this examination.

2.2 Inputs

The reviewers were provided with a Technical Data Package (TDP) including the source code and a set of documents that support the findings in this report. These documents were examined during the source code review to better understand the voting system and identify discrepancies between the documentation and the source code. These documents are listed in the References section.

2.3 Threat Model

This assessment is centered on the threat model given in the Request for Quotation (RFQ). The system is expected to counter the following attacks.

- Alter vote recording
- Alter vote results
- Alter critical election data, such as audit logs
- Conduct a denial of service attack on the voting system

To the extent possible, vulnerabilities found have been reported with an indication of whether the exploitation of the vulnerability would require access by any of the following.

- **Voter:** Usually has low knowledge of the voting machine design and configuration. Some may have more advanced knowledge. May carry out attacks designed by others. They have access to the machine(s) for less than an hour.

- **Poll worker:** Usually has low knowledge of the voting machine design and configuration. Some may have more advanced knowledge. May carry out attacks designed by others. They have access to the machine(s) for up to one week, but all physical security has been put into place before the machines are received.

- **Elections official insider:** Wide range of knowledge of the voting machine design and configuration. May have unrestricted access to the machine for long periods of time. Their designated activities include:

  - Set up and pre-election procedures
  - Election operation
  - Post-election processing of results
  - Archiving and storage operations

- **Vendor insider:** Has great knowledge of the voting machine design and configuration. They have unlimited access to the machine before it is delivered to the purchaser and, thereafter, may have unrestricted access when performing warranty and maintenance service, and when providing election administration services.

The atsec team did not attempt to demonstrate exploitability of identified potential vulnerabilities. However, identified potential vulnerabilities were described along with the anticipated factors necessary to mount an attack.

2.4 Methodology

The atsec team was tasked with the Source Code review which included but was not limited to the following aspects.

- Evaluation of potential vulnerabilities and related issues (code quality and standards compliance), considering that an exploitable issue in a component that is not in itself security relevant could be used to subvert more critical data. This is an issue whenever the architecture of the system does not provide strong separation of the components.

- Adherence to other applicable coding format conventions and standards including best practices for the coding language used, and any IEEE, NIST, ISO or NSA standards or guidelines which the Contractor find reasonably applicable.

- Analysis of the program logic and branching structure.

- Search for exposures to commonly exploited vulnerabilities, such as buffer overflows, integer overflow, inappropriate casting or arithmetic.

- Evaluation of the use and correct implementation of cryptography and key management.
- Analysis of error and exception handling.
- Evaluation of the likelihood of security failures being detected.
    - Are audit mechanisms reliable and tamper resistant?
    - Is data that might be subject to tampering properly validated and authenticated?
- Evaluation of the risk that a user can escalate his or her capabilities beyond those authorized.
- Evaluation of whether the design and implementation follow sound, generally accepted engineering practices. Is code defensively written to protect against:
    - Bad data;
    - Errors in other modules;
    - Changes in environment;
    - User errors; and
    - Other adverse conditions.
- Evaluation of whether the system is designed in a way that allows meaningful analysis, including:
    - Is the architecture and code amenable to an external review (such as this one)?
    - Could code analysis tools be usefully applied?
    - Is the code complexity at a level that it obfuscates its logic?
- Search for embedded, exploitable code (such as “Easter eggs”) that can be triggered to affect the system.
- Search for dynamic memory access features which would permit the replacement of certificated executable code or control data or insertion of exploitable code or data.
- Search for use of runtime scripts, instructions, or other control data that can affect the operation of security relevant functions or the integrity of the data.

2.4.1 Potential vulnerabilities

The reviewers used the following public repositories to identify vulnerabilities that may affect the system.

- MITRE Common Vulnerability and Exposures (CVEs)
- NIST NVD using Common Platform Enumeration (CPE) tool
- COTS component support sites (e.g., HP, JRSoftware, NHibernate)

Although this list may not have entries for the voting system itself, constituent software and commercial off-the-shelf (COTS) components that the voting system integrates may contain vulnerabilities. The review team identified such components that the system relies upon and conducted searches for these products as well.

2.4.2 Code quality

While performing the examination of the code for other activities, the reviewers identified and recorded areas within the code base that demonstrate poor code quality. Although poor code quality does not necessarily identify vulnerabilities, it does provide an indication that vulnerabilities may exist.

The following coding standards were used during this analysis.

- California Voting System Standards, October 2014

The reviewers also compared the code against the Verity Coding Standards that was found in the TDP.

The team also performed numerous informal static analysis activities on the source code to gather code quality data using customized command scripts.

2.4.3 Design

The source code review team used the technical data package, source code, and any material provided or otherwise publicly available to construct an understanding of the architecture and design of the voting system. This understanding included discovering the external interfaces and their security mechanisms and controls, particularly as much information as possible was gathered to support conclusions regarding the ability for a threat agent to tamper with or circumvent security controls.

Interfaces represent the primary attack surface of the voting system. Interfaces can include web-based interfaces, native graphic user interfaces, command line interfaces, or technical interfaces that are not designed for direct user interaction (e.g., database connections). Each of these interfaces was examined to identify the security controls that counter the threats.

Secure interfaces also depend on filtering out poorly structured or corrupt data. The review team specifically checked for input validation mechanisms and determined if related attacks, such as command injection are possible.

2.4.4 Cryptography

While cryptography is often the most difficult security mechanism to break directly, misuse of cryptographic primitives can render that protection weak or non-existent. The review team identified where cryptography is used throughout the source code and determined if its use is appropriate for the given purpose. For example, using a cryptographic hash function to protect passwords is appropriate while using an encryption algorithm with a hard-coded key is not.

2.4.5 Back doors

Those with access to the voting system during development and having malicious intent can place back doors into the source code so that they could gain unauthorized access to the voting system during operation. Back doors are extremely hard to find because a seasoned programmer can obfuscate code to look benign.
The review team marked areas of vulnerabilities as identified by command line searches, as described in section 4.5, for further scrutiny. For example, a particular area of code with poor code quality and access to sensitive information such as authentication credentials might be a good place to hide a back door. The reviewers gave such areas extra scrutiny by considering insider threats in addition to unintentional implementation flaws.

2.4.6 Measurement of findings

A summary of findings is listed in section 4. Each finding contains the following information.

- A description of the vulnerability or weakness
- An assessment of what threats are involved in the possible exploitation of the vulnerability or weakness
- A categorization of the findings, which can be:
    - A weakness in the source code. Weaknesses are issues identified in the source code that are not directly exploitable but may indicate the existence of exploitable vulnerabilities within the source code.
    - A non-conformity in the code quality standards. Non-conformities do not necessarily imply weaknesses, though the rationale for the requirement is often based on preventing weaknesses.
    - A potential vulnerability in the source code. The reviewers consider potential vulnerabilities to likely be exploitable.
    - A vulnerability in the source code. The reviewers have either shown or have referenced other parties who have asserted the vulnerability to be exploitable.
- A severity level of the findings, which can be either:

- A low severity finding. Low severity implies either the impact to the product is low or already mitigated by the system, or the difficulty in exploitation would likely require unrestricted access to the systems, expert knowledge of the system, or would require cost prohibitive resources.

- A medium severity finding. Medium severity implies either the impact of exploitation to the product would be significant, or the difficulty in exploitation would likely require extended access to the systems, informed knowledge of the system, or would require significant resources.

- A high severity finding. High severity implies either the impact of exploitation to the product would result in complete compromise of security, or the difficulty in exploitation would likely require little to no access or knowledge of the systems or little to no resources.

2.4.7 Depth of analysis

Because of the complexity and volume of the material to be reviewed, limited time available and broad scope (assessment of documents and quality of the code, along with source code review), the team concentrated on surveying a breadth of categories of vulnerabilities that they could identify, and only reviewed in depth enough samples of each of the categories to determine how that vulnerability was being handled. For all the categories, no attempt was made to enumerate how many instances existed. Other source code review projects would be likely to find more instances, but those findings should be within the listed categories.

3 Description of the Verity Voting System

The Verity Voting System is a suite of software, hardware, device and peripheral components for conducting and reporting elections.

3.1 Voting System Functions

The Verity Voting System provides a number of high-level functions necessary to conduct an election. These activities include the following.

- Ballot data creation
- Election definition and ballot production
- Device configuration
- Polling-place-based ballot printing
- Polling place Ballot Marking Device
- Polling place ballot review
- Polling place digital scanning for paper ballots
- High-speed, large-volume ballot scanning
- Ballot Adjudication
- Counting of votes/tabulation
- Consolidation and reporting of results and audit logs
- Audits and recounts

3.2 Physical Components

Several components are used in conducting an election with Verity. Some are specialized hardware components built or assembled by Verity, others are COTS products used to run Verity. The following are the specialized hardware components.

Verity Print—An on-demand device for printing and issuing blank paper ballots to voters. The voter completes their ballot and casts it using either Verity Scan or by putting it in a ballot box to be scanned centrally.

Verity Touch Writer—An accessible device for marking digital ballots using a touch screen. After confirming their selections the vote prints the ballot on the attached printer and casts it.

Verity Reader—A device for optionally verifying a ballot. A voter can insert their marked paper ballot to verify how it will be counted and hear audio read-back of their choices. It does not store or tabulate votes.

Verity Scan—A polling-place-based digital device for scanning and casting either hand-marked or printed ballots. The voter can check and correct the ballot before casting and the scanned ballot is depositing into a secure ballot box for storage.

Verity vDrive—A specially formatted USB media used to transfer the election ballot styles to voting devices, to transfer cast vote records to Verity Count for tabulation, and to collect and transfer audit logs.

Verity Access—An accessibility device attached to each Verity Touch Writer and Verity Reader device to provide the reader with additional input options including a scrolling wheel and select button, headphones, and a connection that may be used with tactile buttons or sip-and-puff devices.

Verity Key—A USB device for Verity's two-factor authentication process. Critical operations require the Verity Key to be inserted and a passcode to be entered.

3.3 Logical Components

Verity Data—Ballot design software for entering, importing, and managing election data, jurisdiction data and translations as well as recording and importing audio. It also allows users to choose ballot templates, view ballot previews and lock the election data so that it can be opened in Verity Build.

Verity Build—Election definition, media creation, and ballot printing software for opening elections, proofing data, configuring device settings, printing ballots and writing vDrives and Verity Keys. It also performs the final steps to prepare ballots for elections.

Verity Central—Central ballot scanning and adjudication software for scanning and reviewing ballots, resolving write-in votes and voter intent issues, and writing cast vote records to vDrive for tabulation in Verity Count.

Verity Count—Ballot tabulation and reporting software for reading vDrives, tabulating ballots, resolving write-in votes, printing reports, and exporting election results.

3.4 Interfaces

The voting system moves data between external interfaces and internal components in a variety of ways: peripheral devices, files, and databases. This section will discuss these interfaces in more detail.

3.4.1 Network interfaces

The logical components located at the Data/Build (can be one machine), Central, and Count sites will use Ethernet for network connectivity. Hart will provide the hardware necessary to instantiate a closed network at each individual location. The use of non-hardwired connectivity will not be permitted and such functionality has been disabled (e.g., wireless, Bluetooth).

The communication channels to other IT entities will be protected using TLS over TCP/IP.

3.4.2 Peripheral devices

Data is moved between logical components using different media. Depending on the purpose of the data, the appropriate transport mechanism is chosen. Data will either be transported as a digital file or physically printed material. Table 1 summarizes the appropriate export media for each device or area.

Component	Export mechanism
Verity Data and Verity Build (these are on the same system)	• Verity vDrive (election definitions) • Verity Keys (voting equipment programming) • Pre-voting paper ballots
Verity Print	Pre-voting ballots
Verity Central	Verity vDrive (Cast Vote Records)
Verity Touch Writer	• Voter marked ballots (digital form) • Verity vDrive (audit logs)
Verity Scan	Verity vDrive (Cast Vote Records and audit logs)

Table 1: Device/Area Media

3.4.3 Files

Many file types are used by various components of the voting system and are transferred by a variety of interfaces and media. The following types of data are stored in the voting system.

- Election data and configuration files
- Media files (e.g., audio)
- OS/WES7 configuration files
- Device configuration files
- Election results files
- Cast Vote Record (CVR) data files
- Audit logs

- Extensible Markup Language (XML)
- PMQ files
- Signature files

3.4.4 Databases

The Verity voting system uses two different Microsoft SQLServer databases: SQLServer 2012 and SQLServer Compact. Microsoft SQLServer 2012 was stated as the version used. However, it is unclear which version of SQLServer Compact is used.

The following applications that will have access to the Microsoft SQLServer 2012 database.

- Verity Build
- Verity Central
- Verity Count
- User Management
- Election Manager

Verity Device databases are created using Microsoft SQLServer Compact. These devices include the following.

- Verity Scan
- Verity Touch Writer
- Verity Reader

4 Findings

4.1 Public Vulnerability Search

Table 6 lists the publicly known vulnerabilities identified that could potentially impact the voting system.

Name (Vulnerable component)	Description	Additional information
CVE-2016-2243 (HP Z240 Workstation)	Local users can cause the system to fail to recover the BIOS.	https://support.hp.com/us-en/document/c05012469 https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-2243
Meltdown/Spectre (HP Z240 Workstation)	The computer system is susceptible to the Meltdown/Spectre malware.	https://support.hp.com/us-en/document/c05869091
Outdated version (SQLServer 2012)	The software version being used (11.0.2100) is out dated. There have been numerous Service Packs and Security updates since this release. The reviewer suggests updating to address any discovered (and potentially undiscovered) vulnerabilities. The latest service pack is Service Pack 4.	https://support.microsoft.com/en-us/help/321185/how-to-determine-the-version-edition-and-update-level-of-sql-server-an
SP1 installation failure (Windows Embedded System 7)	Certain updates related to SP1 may fail when updating the OS. Some updates that could fail are: • KB2949927: Availability of SHA-2 Hashing Algorithm for Windows 7 and Windows Server 2008 R2 • KB3033929: Security Update for Windows 7 for x64-based Systems	https://support.microsoft.com/en-us/help/3189682/serviceability-update-for-windows-embedded-standard-7-sp1-posready-7-a

Name (Vulnerable component)	Description	Additional information
	• KB3110329: Security Update for Windows 7 The reviewer is providing this information to ensure the developer is aware of this situation and develop an appropriate mitigation strategy.	
Meltdown (Windows Embedded System 7)	The operating system used is susceptible to the Meltdown malware.	https://news.softpedia.com/news/microsoft-re-issues-kb4056894-kb4056892-meltdown-spectre-for-some-amd-chips-519350.shtml Meltdown patch (KB4056894): https://www.catalog.update.microsoft.com/Search.aspx?q=4056894
Bugs (NHiberate)	The reviewer was unable to determine the version of NHibernate the system is using. The link provided will list bugs found in all versions of the software. The reviewer suggests the developer look at the list and find bugs against the version of NHiberate being used.	https://github.com/nhibernate/nhibernate-core/issues?q=is%3Aopen+is%3Aissue+label%3A%22t%3A+Bug%22
Bugs (Fluent NHibernate)	The reviewer was unable to determine the version of Fluent NHibernate the system is using. The link provided will list bugs found in all versions of the software. The reviewer suggests the developer look at the list and find bugs against the version of Fluent NHiberate being used.	https://github.com/FluentNHibernate/fluent-nhibernate/issues?q=is%3Aopen+is%3Aissue+label%3Abug

Name (Vulnerable component)	Description	Additional information
DLL Hijacking (InnoSetup/Inno Script Setup)	The software is susceptible to including a Trojan horse DLL that is located in an untrusted path on the system.	https://packetstormsecurity.com/files/134694/jrsoft-dllhijack.txt http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-4833 http://jrsoftware.org/files/is5-whatsnew.htm http://news.jrsoftware.org/news/innosetup/msg103180.html http://news.jrsoftware.org/news/innosetup/msg103182.html
Malformed Windows binary not protected (McAfee Application Control for Devices)	The reviewer noticed how the developer has whitelisted what files should run on the system. The following CVE is something to be aware of in the event a file is added that the program may think is non-executable.	http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-9920 https://kc.mcafee.com/corporate/index?page=content&id=SB10077

Table 2: Potential Vulnerabilities Identified

4.2 Static Code Analysis & Documentation Review

Table 7 summarizes the findings that arose from the source code review team's assessment of the voting system. Potential exploitation of a weakness or vulnerability and type of attacker is noted where applicable.

Description	Assessment	Categorization
Usage of SHA-1 to sign data and sign hash. Weakness of SHA-1 could allow for corruption of data.	The use of SHA-1 for signature generation is generally disallowed by NIST (except where specifically allowed in NIST protocol-specific guidance); however, its use for digital signature verification is allowed for legacy use (i.e., the verification of already-generated digital signatures). *Note: NIST still does allow testing of SHA-1 although NIST have requested federal agencies to stop using SHA-1.*	**Type:** potential vulnerability and non-conformity (FIPS) **Severity:** low

Description	Assessment	Categorization
	NIST have provided guidance on mitigating the risk of using SHA-1 in SP800-131a.	
Microsoft SQL Server 2012 (MSSQL) is not patched.	Not having the latest version of the software leaves the system open to undiscovered and unpatched vulnerabilities. The version of MSSQL installed is 11.0.2100. There have been numerous Service Packs and Security updates since this release. In order to address any discovered (and potentially undiscovered) vulnerabilities, updating to the latest Service Pack is recommended (Service Pack 4).	**Type:** potential vulnerability **Severity:** medium
The ability to write directly to memory is a potential vulnerability.	The code has access to the system memory. Malicious code could be interjected and executed.	**Type:** potential vulnerability **Severity:** low
Although compliant with CVSS 7.2.3 h) requirements, password complexity rules do not follow best practices. The password complexity criteria allow for a reduced strength password. If the system becomes compromised, this will allow a threat agent to determine a user's password more quickly.	The algorithm used to determine if a user's password is strong enough is not using best practices. The algorithm does not enforce the use of lower case lettering. This will allow a weaker password, by default. Also, the minimum length acceptable is too small. Current practice for secure password length is 12 to 15 characters.	**Type:** potential vulnerability **Severity:** low
Password information is not being stored using an appropriate/suitable class. Password	The password field is being stored in the String class. The All-in-one Code Framework Coding Standards states that MSDN provides guidelines for using the .NET	**Type:** potential vulnerability **Severity:** medium

Description	Assessment	Categorization
information is being stored in an unsecure part of memory while being used. Code could be interjected, using other parts of the code base, to access this sensitive information.	framework. Since passwords are sensitive information, a different class more suitable to this type of information should be used, such as the SecureString class. From the Microsoft MSDN site: *When created properly, a SecureString instance provides more data protection than a String. When creating a string from a character-at-a-time source, String creates multiple intermediate in memory, whereas SecureString creates just a single instance. Garbage collection of String objects is non-deterministic. In addition, because its memory is not pinned, the garbage collector will make additional copies of String values when moving and compacting memory. In contrast, the memory allocated to a SecureString object is pinned, and that memory can be freed by calling the Dispose method.*	
Usage of authorization check for a plugin could allow unauthorized use of the plugin and manipulation by an unauthorized user.	There is use of AuthorizationException in the modules. A catch of the exception was detected but it did not handle the exception.	**Type:** vulnerability **Severity:** medium
The SQL query strings are being constructed in a way that is vulnerable to SQL injection attacks. CVSS 5.2.8	SQL statements are being constructed using '+' or append method on a class. Best practice (OWASP) is to use parameterized queries for constructing SQL statements.	**Type:** vulnerability **Severity:** medium

Description	Assessment	Categorization
Unable to determine security policy regarding configuration file access and modification.	The security policy was unable to be fully understood based on the PDFs from the technical document package.	**Type:** potential vulnerability **Severity:** low
There are instances where exception/error handling doesn't follow a consistent implementation. CVSS 5.2.5	There are instances where exception/error handling doesn't follow a consistent implementation. Additionally, these instances do not follow the guidelines described in the Verity Logging Technical Requirements Document (section 5.3: Events that must be logged). For example, there are catch blocks that do not write to any log files, there are instances where exceptions are caught and assigned to variables where nothing is done with that variable, and there are empty catch blocks.	**Type:** non-conformity, potential vulnerability **Severity:** low
Product is no longer FIPS 140-2 certified or conformant as claimed. Currently Non-approved algorithms may contain exploitable weaknesses and any data protected by encryption may be at risk. CVSS 7.5.4 a) iii)	Product documentation states all cryptographic modules used are FIPS 140-2 validated. The FIPS 140-2 certificates listed in "Verity Operational Environment 4005515 C00.pdf," Appendix A, (#1319,#1326,#1327,#1328,#1329,#1330,#1331) are all now on the CMVP historical list. Each historical certificate carries the following disclaimer: "Historical - The referenced cryptographic module should not be included by Federal Agencies in new procurements. Agencies may make a risk determination on whether to continue using this module based on their own assessment of where and how it is used." FIPS requirements have changed since 2011 when these certificates were issued.	**Type:** non-conformity, potential vulnerability **Severity:** medium

Description	Assessment	Categorization
	Technically, all modules used have been validated as conformant in 2011, but now the certifications have expired. *Note: CVSS is not clear in regard to if historical FIPS 140-2 certifications are allowed.*	

Table 3: Summary of issues discovered during the static code analysis

Glossary

AES	Advanced Encryption Standard
API	Application Programming Interface
CAPI	Crypto API
CBC	Cipher Block Chaining
CMVP	Cryptographic Module Validation Program
COTS	Commercial Off-The-Shelf
CPE	Common Platform Enumeration
CRC	Cyclic Redundancy Check
CTR	Counter
CVE	Common Vulnerability and Exposures
CVR	Cast Vote Record
CWE	Common Weakness Enumeration
TDES	Triple-Data Encryption Standard
EC	Elliptic Curve
ECDSA	Elliptic Curve Digital Signature Algorithm
EDM	Election Data Manager
ELS	Event Log Service
EMS	Election Management System
EQC	Election Qualification Code
ERM	Election Reporting Manager
EVS	Verity Voting System
FIPS	Federal Information Processing Standard
HMAC	Hash Message Authentication Code
HTTP	Hyper Text Transfer Protocol

HTTPS	Hyper Text Transfer Protocol Secure
IP	Internet Protocol
IV	Initialization Vector
KDF	Key Derivation Function
LAN	Local Area Network
OS	Operating System
PBKDF	Password-Based Key Derivation Function
PC	Personal Computer
PKI	Public Key Infrastructure
PRF	Pseudo-Random Function
PRNG	Pseudorandom Number Generator
RCV	Ranked Choice Voting
RMS	Removable Media Service
RNG	Random Number Generator
RSA	Rivest-Shamir-Adleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standards
SOS	Secretary of State
TCP	Transmission Control Protocol
TDP	Technical Data Package
TRD	Technical Requirements Document
USB	Universal Serial Bus
VAT	Voter Assist Terminal
XML	Extensible Markup Language

References

Documentation provided for the source code review included Verity EVS product documentation and other publicly available standards documents. The atsec source code review team also consulted other publicly available documents listed in the last group.

Verity Documents

PC Application Framework UI Design Document, Version 5

Print Design

Reader Design

Release Notes Verity Voting 3.0.0, Version A.00, Published 09-08-2016

Scan Design

Touch Writer Design

Verity 3.0 Desktop Database Schema

Verity 3.0 Device Database Schema

Verity Airgap Interface Technical Reference, Version A.02, Published 2014

Verity API Specification, Version A.03, Published 09-06-2017

Verity Base Station Microcontroller, Version A.01, Published 05-26-2015

Verity Build 3.0 Modification TRD, Version A.01, Published 09-01-2017

Verity Build TRD, Version A.10, Published 10-14-2015

Verity Build User Interface Design Document, Version 11

Verity California Use Procedures, Verity Voting 3.0, Published 2017

Verity Central 3.0 Modification TRD, Version A.01, Published 07-18-2017

Verity Central TRD, Version A.07, Published 10-22-2015

Verity Central User Interface Specification, Version 8

Verity Coding Standard, Version A.15, Published 03-30-2018

Verity Count 3.0 Modification TRD, Version A.01, Published 09-01-2017

Verity Count TRD, Version A.10, Published 11-02-2015

Verity Count User Interface Specification, Version 9

Verity Data 3.0 Modification TRD, Version A.01, Published 09-01-2017

Verity Data TRD, Version A.06

Verity Desktop User Interface Design Document, Version 3

Verity Device Suite 3.0 Modification TRD, Version A.01, Published 09-01-2017

Verity Device Suite TRD, Version A.09, Published 11-03-2015

Verity Election Definition Data TRD, Version A.01, Published 10-14-2015

Verity Election Management TRD, Version A.06, Published 10-21-2015

Verity Election Management User Interface Design Document, Version 1

Verity Key Design Technical Document, Version A.01, Published 05-26-2015

Verity Logging 3.0 Modification TRD, Version A.01, Published 09-01-2017

Verity Logging Design Technical Document, Version 1.03, Published 05-26-2015

Verity Logging TRD, Version A.04, Published 10-21-2015

Verity Print TRD, Version 6, Published 11-03-2015

Verity Reader TRD, Version A.01, Published 09-01-2017

Verity Risk and Threat Assessment, Version B.00, Published 02-14-2017

Verity Scan 3.0 Modification TRD, Version A.01, Published 09-01-2017

Verity Scan TRD, Version A.07, Published 10-27-2015

Verity Security Requirements Document, Version A.07, Published 10-28-2015

Verity Shared Device User Interface Design Document, 7 Version

Verity Software Architecture & Design, Version C.00, 02-14-2017

Verity System Design Verity Electronics Specification, Version A.13, Published 05-01-2017

Verity Touch Writer 3.0 Modification TRD, Version A.01, Published 09-01-2017

Verity Touch Writer TRD, Version A.09, Published 11-03-2015

Verity Touch Writer User Interface Design Document, Version 12

Verity User Management 3.0 Modification, Version A.01, Published 09-01-2017

Verity User Management TRD, Version A.01, Published 10-21-15

Verity User Management User Interface Design Document, Version 2

Verity Voting 3.0 System Limits, Version C.00, Published 08-30-2017

Verity Voting Verity Operational Environment, Version C.00, Published 09-11-2017

Public Documents

All-In-One Code Framework Coding Standards, Published 2014

California Voting System Standards, Published October 2014

National Institute of Standards and Technology, Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program, January 2016, http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf

National Institute of Standards and Technology, FIPS 140-2 Security Requirements for Cryptographic Modules, May 2001, http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

National Institute of Standards and Technology, FIPS 140-2 Annex A: Approved Security Functions, December 2002, http://csrc.nist.gov/publications/fips/fips140-2/fips1402annexa.pdf

National Institute of Standards and Technology, FIPS 180-4 Secure Hash Standard (SHS), March 2012, http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf

National Institute of Standards and Technology, FIPS 186-4 Digital Signature Standard (DSS), July 2013, http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

National Institute of Standards and Technology, FIPS 197 Advanced Encryption Standard, November 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

National Institute of Standards and Technology, FIPS 198-1 The Keyed-Hash Message Authentication Code (HMAC), July 2008, http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf

National Institute of Standards and Technology, NIST Special Publication 800-57, Recommendation for Key Management—Part 1: General (Revised), January 2016, http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf

National Institute of Standards and Technology, NIST Special Publication 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, June, 2015, http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

National Institute of Standards and Technology, NIST Special Publication 800-131A, Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths, November, 2015, https://csrc.nist.gov/publications/detail/sp/800-131a/rev-1/final

OWASP, Open Web Application Security Project, The OWASP SQL and database Scripting Technology Knowledge Base, https://www.owasp.org/index.php/Query_Parameterization_Cheat_Sheet