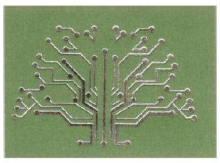

FREEMAN, CRAFT, MCGREGOR GROUP

Source Code Review

ES&S Unity v 3.4.1.0 Voting System

Report Date: 2016-11-10

Version: 1.4

Status: RELEASED

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	2016-03-29	Ryan Hill	Initial version
1.1	2016-04-07	Ryan Hill	Added client edits
1.2	2016-04-15	Ryan Hill	Address additional comments, organizational cleanup
1.3	2016-04-20	Ryan Hill	Address additional comments
1.4	2016-10-11	Ryan Hill	Address additional comments

Trademarks

Table of Contents

1 Executive Summary

This report was prepared by atsec information security corporation to review aspects of the security and integrity of the ES&S Unity v 3.4.1.0 Voting System. 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.

The static code analysis revealed twenty-seven issues, and the public vulnerability search identified twenty-two vulnerabilities that could potentially be used for an attack on the voting system. (Note that the table in section 4.2 only has twenty-six issues because finding 014 was deleted as it was a duplicate of finding 007.)

While the source code review team did not find any critical vulnerabilities, six of those reported were assessed as being of medium severity.

At a high level, weaknesses and vulnerabilities were identified that can be attributed to difficulties resulting from an aging and repeatedly maintained system. These include failure to keep documentation up to date and failure to upgrade cryptography as cryptographic algorithms become more susceptible to attacks over time.

Issues associated with passwords are still apparent. Some issues in this class were identified in a previous study of the system performed by atsec in 2008. The review team did not find identical issues, although issues were still found in regard to hard-coded passwords and weak passwords (e.g. a password with a length of three characters.)

In addition, numerous less severe but still noteworthy vulnerabilities were found related to code quality and non-conformance to the 2005 Voluntary Voting System Guidelines. 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 ES&S Unity v 3.4.1.0 Voting System. 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 ES&S Unity v 3.4.1.0 (EAC Certification Number: ESSUNITY3410) Voting System 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:

- King Ables
- Quentin Gouchet
- Ryan Hill
- Hedy Leung
- Sean Lewis
- Fiona Pattinson
- David Rumley
- Swapneela Unkule

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 ES&S Unity v 3.4.1.0 Voting System (hereafter referred to as the “voting system” or simply as the “system”) is a paper-based voting system made up of the Election Management System (EMS), precinct tabulators, a Help America Vote Act compliant ballot marking device, and central count tabulators.

The system has the following software components:

- Audit Manager (AM),
- Election Data Manager (EDM),
- Hardware Programming Manager (HPM),
- ES&S Ballot Image Manager (ESSIM),
- Election Reporting Manager (ERM),
- AutoMARK Information Management System (AIMS)
- Log Monitor Service, and
- VAT Previewer.

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

- DS200 Precinct Tabulator,
- Model 100 Precinct Tabulator,

- AutoMARK Voting Assist Terminal,
- Model 650 Central Tabulator, and
- DS850 Central Tabulator.

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 Statement of Work. 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 the:

- **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 the applicable standards in sections: 5 of Volume I, 7 of Volume I, and 5 of Volume II of the 2005 Voluntary Voting System Guidelines.
- 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 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 searched the following public lists to identify vulnerabilities that may affect the system.

- NIST's National Vulnerability Database
  https://nvd.nist.gov/
- Mitre Common Vulnerability and Exposures (CVEs) list
  https://cve.mitre.org/cve/cve.html

Although these lists 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 standard was used during this analysis.

- 2005 Voluntary Voting System Guidelines [VVSG1], [VVSG2] and supplemental interpretation statements found at:
  http://www.eac.gov/testing_and_certification/request_for_interpretations1.aspx.

The reviewers also compared the code against software engineering best practices. Examples of best practices can be found in the following books:

- The CERT Oracle Secure Coding Standard for Java [CERTJ]
- The CERT C Security Coding Standard [CERTC]

These standards are based on accepted industry best practices in developing C/C++ code and in managed code (e.g., Java, J#, C#). There is no widely-accepted coding standard specific to COBOL. The system does not appear to contain Java source code, but the standard still provided useful input into the analysis of all code quality, including COBOL.

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 Use Procedures, Installation Procedures, 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.

The design description also provided a mapping from the high-level features and interfaces of the product down to where the reviewers believed those features and interfaces are implemented.

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 above 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 with 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 indefinite 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 ES&S Unity

ES&S Unity is a suite of software and hardware components for conducting and reporting elections.

3.1 Voting System Functions

The ES&S Unity voting system provides a number of high-level functions necessary to conduct an election. These activities include the following.

- Creation and definition of ballots
- Programming of precinct scanners and other hardware
- Tabulation and reporting of election results
- Audit logs generated for operations, ballots, and user activities

Precinct scanners are used for processing ballots at each polling place. Election data is gathered for tabulation and reporting, and an audit record is generated for all activities. Election data is transferred between components via physical move of compact flash and USB flash drives.

3.2 Physical Components

Several components are used in conducting an election with ES&S Unity. Some are specialized hardware components built or assembled by ES&S, others are commercial off-the-shelf (COTS) products used to run ES&S Unity. The following are the specialized hardware components.

M100 Precinct Tabulator—An optical scanner used to process ballots at a polling location. The scanner tabulates vote entries, provides an opportunity to fix ballots with errors, such as overvoted ballots, and sorts out ballots with write-in candidates. The scanner sits on top of a storage bin for processed ballots. It is capable of generating running totals of poll results. It can store its poll results on a PC card for tabulation at a central count location, and generates date stamped audit logs of all scanner activities.

DS200 Precinct Tabulator—A touchscreen digital scanner that is designed to scan each paper ballot at the polling site. The image-scanner uses a set of two high-resolution cameras to simultaneously image the front and back of a ballot. The resulting ballot images are then decoded by a recognition engine. The tabulator stores current totals in both internal memory and on removable USB flash memory. It can also produce reports from the scanner's internal printer.

M650 Central Scanner & Tabulator—A central scanner and tabulator that is made to work with a jurisdiction of any size regardless of the ballot complexity. It uses optical mark read (OMR) scanning to securely process each ballot length from 14 to 19 inches. The M650 Central Scanner & Tabulator can process more than 300 ballots per minute.

DS850 High-Speed Tabulator—A high-speed digital scanner and ballot counter. While scanning, the DS850 will print a continuous audit log to an attached printer. The scanner stores all of the scanned data internally and to results collection media that officials can use to format and print results. The results collection media can be transferred to a PC that is running the Election Reporting Manager (ERM) software.

AutoMARK Voting Assist Terminal—An optical scan ballot marker that assists voters with disabilities.

3.3 Logical Components

The Election Management System (EMS) is a set of applications used during the pre-election design phase and post-election tabulation phase of an election.

3.3.1 Election Management System software

EMS client and server components can run on the same physical server (including the back-end server) or on one or more separate platforms. EMS components operate on a stand-alone, hardened system.

Access control to EMS client components is role-based where each user is a member of a single pre-defined role, created and managed by an EMS election administrator. Each role has its own set of permissions that govern what operations the user is allowed to perform.

The **Audit Manager (AM)** stores detailed logs of the actions performed in both the Election Data Manager and the Image Manager. It is used to view audit logs and control user access for both of those components.

The **AutoMARK Information Management System (AIMS)** manages election specific information for the Voter Assist Terminal (VAT), including information about precincts, splits, races, and candidates.

The **Election Data Manager (EDM)** is a single-entry database that stores all of a jurisdiction's precinct, office, and candidate information.

The **Election Reporting Manager (ERM)** is used to consolidate ballot tabulations, and report on totals. It is able to print out reports of polling and cast ballot results.

The **ES&S Ballot Image Manager (ESSIM)** displays ballot style information in a "what you see is what you get" (WYSIWYG) design interface. The ballot style information is created by Unity Election Data Manager. Users can change certain formatting (font, size, etc.) and elements of the individual components of the ballot.

The **Hardware Programming Manager (HPM)** uses the election specific database created by EDM and ESSIM to program the appropriate media for ES&S tabulation devices.

The **Log Monitor Service** is a background service provided by Microsoft Windows that will run in the background of any active Election Management software application. Log Monitor Service monitors the Windows Event Viewer to ensure that the each Election Management software is functioning properly.

The **VAT Preview** is part of AIMS that lets the user preview text, audio and the layout of the screen before downloading that data to the AutoMARK.

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, the types of data, and where the data goes.

3.4.1 Peripheral devices

Data is transferred between the systems using the appropriate media. The appropriate media varies depending on the device that is receiving the data. The appropriate media depends on the tabulator that the data is being transferred to/from. The following table summarizes the appropriate media for each device.

Device	Appropriate Media
DS200 and DS850	USB Flash Drive
M100	PCMCIA Card
M650	Zip Disk
AutoMark	CF Card

Ballot layout data will be transferred from the HPM to the appropriate tabulators. Election results data will be transferred from the appropriate tabulators to the ERM.

The HPM will transfer the ballot layout data via the appropriate media: a USB flash drive for the DS200 and DS850, a PCMCIA card for the M100, a CF card for the AutoMark or a Zip disk for the M650 tabulators.

The ERM can be configured to have a monitor display the ballot data as it is being tabulated by the DS200 and DS850.

4 Findings

4.1 Public Vulnerability Search

The following table lists the CVEs identified that could potentially impact the voting system.

CVE	Description	Rationale
CVE-2016-0020	Microsoft Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, and Windows 7 SP1 mishandle DLL loading, which allows local users to gain privileges via a crafted application, aka "MAPI DLL Loading Elevation of Privilege Vulnerability."	The system uses Windows 7 OS and makes use of dynamic-link libraries (DLLs). Further investigation by the developer is required to determine the impact of this CVE.
CVE-2016-0133	The USB Mass Storage Class driver in Microsoft Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8.1, Windows Server 2012 Gold and R2, Windows RT 8.1, and Windows 10 Gold and 1511 allows physically proximate attackers to execute arbitrary code by inserting a crafted USB device, aka "USB Mass Storage Elevation of Privilege Vulnerability."	The system uses Windows 7 OS and makes use of the USB port. Further investigation by the developer is required to determine the impact of this CVE.
CVE-2015-2371	The Windows Installer service in Microsoft Windows Server 2003 SP2 and R2 SP2, Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows 8.1, Windows Server 2012 Gold and R2, and Windows RT Gold and 8.1 allows local users to gain privileges via a custom action script associated with a .msi package, aka "Windows Installer EoP Vulnerability."	The system uses Windows 7 OS and MSI packages. Further investigation by the developer is required to determine the impact of this CVE.
CVE-2015-0537	Integer underflow in the base64-decoding implementation in EMC RSA BSAFE Micro Edition Suite (MES) 4.0.x before 4.0.8 and 4.1.x before 4.1.3, RSA BSAFE Crypto-C Micro Edition (Crypto-C ME) before 4.0.4 and 4.1, and RSA BSAFE SSL-C 2.8.9 and earlier allows remote attackers to cause a denial of service (memory corruption or segmentation fault) or possibly have unspecified other impact via crafted base64 data, a	The system uses version 3 of RSA BSAFE Crypto-C Micro Edition (Crypto-C ME). The CVE description states that “RSA BSAFE Crypto-C Micro Edition (Crypto-C ME) before 4.0.4 and 4.1” are affected. Hence the CVE is applicable.

CVE	Description	Rationale
	similar issue to CVE-2015-0292.	
CVE-2015-8153	An SQL injection vulnerability in Symantec Endpoint Protection Manager (SEPM) 12.1 before RU6-MP4 allows remote authenticated users to execute arbitrary SQL commands via unspecified vectors.	To mitigate this vulnerability, administrators can disable the Application and Device Control (ADC) driver or uninstall ADC in SEP 12.1 by following one of these options as stated in this security advisory. The documentation states that Symantec Endpoint Protection_12.1.4 is used. More evidence and a system check are required to see if this CVE is still applicable.
CVE-2013-5015	SQL injection vulnerability in the management console in Symantec Endpoint Protection Manager (SEPM) 11.0 before 11.0.7405.1424 and 12.1 before 12.1.4023.4080, and Symantec Protection Center Small Business Edition 12.x before 12.1.4023.4080, allows remote authenticated users to execute arbitrary SQL commands via unspecified vectors.	Symantec Endpoint Protection_12.1.4 is used. More evidence and a system check are required to see if this CVE is still applicable.
CVE-2013-5014	The management console in Symantec Endpoint Protection Manager (SEPM) 11.0 before 11.0.7405.1424 and 12.1 before 12.1.4023.4080, and Symantec Protection Center Small Business Edition 12.x before 12.1.4023.4080, allows remote attackers to read arbitrary files via XML data containing an external entity declaration in conjunction with an entity reference, related to an XML External Entity (XXE) issue.	Symantec Endpoint Protection_12.1.4 is used. More evidence and a system check are required to see if this CVE is still applicable.
CVE-2014-0323	win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows 8.1, Windows Server 2012 Gold and R2, and Windows RT Gold and 8.1 allows local users to obtain sensitive information from kernel memory or cause a denial of service (system hang) via a crafted application, aka "Win32k	The system uses Windows Server 2008 SP1. Hence the CVE is applicable.

CVE	Description	Rationale
	Information Disclosure Vulnerability."	
CVE-2013-5058	Integer overflow in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows 8.1, and Windows Server 2012 Gold and R2 allows local users to gain privileges via a crafted application, aka "Win32k Integer Overflow Vulnerability."	The system uses Windows Server 2008 SP1. Hence the CVE is applicable.
CVE-2013-3200	The USB drivers in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows Server 2012, and Windows RT allow physically proximate attackers to execute arbitrary code by connecting a crafted USB device, aka "Windows USB Descriptor Vulnerability."	The system uses Windows Server 2008 SP1 and Windows 7. Hence the CVE is applicable.
CVE-2013-1345	win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows Server 2012, and Windows RT does not properly handle objects in memory, which allows local users to gain privileges via a crafted application, aka "Win32k Vulnerability."	The system uses Windows Server 2008 SP1 and Windows 7. Hence the CVE is applicable. The other CVEs related to win32k.sys also apply CVE-2013-1344, CVE-2013-1343 CVE-2013-1342 CVE-2013-1341 CVE-2013-1340 CVE-2013-1334 CVE-2013-1300 CVE-2013-1283
CVE-2013-1280	The kernel in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, Windows Server 2012, and Windows RT does not properly handle objects in memory, which allows local users to gain privileges via a crafted application, aka "Windows Kernel Reference Count	The system uses Windows Server 2008 SP1 and Windows 7. Hence the CVE is applicable.

CVE	Description	Rationale
	Vulnerability."	
CVE-2013-1279	Race condition in the kernel in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, Windows Server 2012, and Windows RT allows local users to gain privileges via a crafted application that leverages incorrect handling of objects in memory, aka "Kernel Race Condition Vulnerability," a different vulnerability than CVE-2013-1278.	The system uses Windows Server 2008 SP1 and Windows 7. Hence the CVE is applicable.
CVE-2011-2018	The kernel in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, and Windows 7 Gold and SP1 does not properly initialize objects, which allows local users to gain privileges via a crafted application, aka "Windows Kernel Exception Handler Vulnerability."	The system uses Windows Server 2008 SP1 and Windows 7. Hence the CVE is applicable.
CVE-2012-0001	The kernel in Microsoft Windows XP SP2, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 does not properly load structured exception handling tables, which allows context-dependent attackers to bypass the SafeSEH security feature by leveraging a Visual C++ .NET 2003 application, aka "Windows Kernel SafeSEH Bypass Vulnerability."	The system uses Visual Studio .NET 2003. Hence the CVE is applicable.
CVE-2010-3190	Untrusted search path vulnerability in the Microsoft Foundation Class (MFC) Library in Microsoft Visual Studio .NET 2003 SP1; Visual Studio 2005 SP1, 2008 SP1, and 2010; and Visual C++ 2005 SP1, 2008 SP1, and 2010 allows local users to gain privileges via a Trojan horse dwmapi.dll file in the current working directory during execution of an MFC application such as AtlTraceTool8.exe (aka ATL MFC Trace Tool), as	The system uses Visual Studio .NET 2003. Hence the CVE is applicable.

CVE	Description	Rationale
	demonstrated by a directory that contains a TRC, cur, rs, rct, or res file, aka "MFC Insecure Library Loading Vulnerability."	
CVE-2009-0090	Microsoft .NET Framework 1.0 SP3, 1.1 SP1, and 2.0 SP1 does not properly validate .NET verifiable code, which allows remote attackers to obtain unintended access to stack memory, and execute arbitrary code, via (1) a crafted XAML browser application (XBAP), (2) a crafted ASP.NET application, or (3) a crafted .NET Framework application, aka "Microsoft .NET Framework Pointer Verification Vulnerability."	The system uses .NET Framework 1.0. Hence the CVE is applicable.
CVE-2012-1527	Integer underflow in Windows Shell in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, and Windows Server 2012 allows local users to gain privileges via a crafted briefcase, aka "Windows Briefcase Integer Underflow Vulnerability."	The system uses Windows Server 2008 SP1 and Windows 7. Hence the CVE is applicable.
CVE-2008-4110	Buffer overflow in the SQLVDIRLib.SQLVDirControl ActiveX control in Tools\Binn\sqlvdir.dll in Microsoft SQL Server 2000 (aka SQL Server 8.0) allows remote attackers to cause a denial of service (browser crash) or possibly execute arbitrary code via a long URL in the second argument to the Connect method. NOTE: this issue is not a vulnerability in many environments, since the control is not marked as safe for scripting and would not execute with default Internet Explorer settings.	The system uses SQL Server 2000. Hence the CVE is applicable.
CVE-2003-0232	Microsoft SQL Server 7, 2000, and MSDE allows local users to execute arbitrary code via a certain request to the Local Procedure Calls (LPC) port that leads to a buffer overflow.	The system uses SQL Server 2000. Hence the CVE is applicable.

CVE	Description	Rationale
CVE-2003-0231	Microsoft SQL Server 7, 2000, and MSDE allows local or remote authenticated users to cause a denial of service (crash or hang) via a long request to a named pipe.	The system uses SQL Server 2000. Hence the CVE is applicable.
CVE-2002-1138	Microsoft SQL Server 7.0 and 2000, including Microsoft Data Engine (MSDE) 1.0 and Microsoft Desktop Engine (MSDE) 2000, writes output files for scheduled jobs under its own privileges instead of the entity that launched it, which allows attackers to overwrite system files, aka "Flaw in Output File Handling for Scheduled Jobs."	The system uses SQL Server 2000. Hence the CVE is applicable.

Table 1: CVEs identified during the public vulnerability search

4.2 Static Code Analysis & Documentation Review

The following table 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.

ID	Description	Assessment	Categorization
001	Weak encryption algorithms and key derivation methods used.	Encryption, decryption, key generation, and key derivation algorithms not approved by NIST are used.	**Type:** Weakness **Severity:** Medium
002	Algorithms not approved by NIST are used.	Non-approved algorithms include Blowfish and CRC-16/32.	**Type:** Weakness **Severity:** Medium
003	Documentation is not up to date.	Documentation has not been updated consistently to match changes made in the product.	**Type:** Weakness **Severity:** Medium
004	Time Synchronization.	No instructions are given in regard to setting the time on the system. No mention of NTP services is found in the documentation. Being able to reset the date and time will open potential vulnerabilities in regard to time dependent functions, e.g. audit logs.	**Type:** Vulnerability **Severity:** Medium
005	Database password hard coded in Source code.	Found hardcoded database passwords and indications of their uses.	**Type:** Weakness **Severity:** Low

ID	Description	Assessment	Categorization
006	Not meeting Default Case requirements in Section 5 Volume II of the 2005 Voluntary Voting System Standards.	VVSG requirement mandates use of default choice in switch statement. The explicitly defined default choice is supposed to catch all else values not included in the case list. In some instances the requirement is trivially met because even though the 'default' choice is present; it is not present as the last choice. So it's not serving the purpose of catching all the else cases.	**Type:** VVSG Non-conformity **Severity:** Low
007	Not meeting line width requirements in Section 5 Volume II of the 2005 Voluntary Voting System Standards. (Note that the missing 014 is a duplicate of this issue as referred to in the Executive Summary.)	VVSG requires no line of code exceed 80 columns in length (including comments and tab expansions) without justification. This requirement seems dated as non-punch-card-based languages often contain longer lines to accommodate indentation to make the code more readable. The reviewer examined several of the files identified and finds no readability issues created by the longer lines. This item is listed only because it is technically a non-conformity with the VVSG standard.	**Type:** VVSG Non-conformity **Severity:** Low
008	The three keys in Triple-DES encryption are not checked for being different.	NIST requirements for Triple-DES encryption are now that all three Triple-DES keys should be different. This check is not performed by the code when performing Triple-DES encryption.	**Type:** Weakness **Severity:** Medium
009	Possible non encryption of Database back up file and hard coded password to protect media on which this files resides.	It is found that the procedure used to back up database in a file allows database back up in a file without any password if empty string is passed as input. The procedure has a password passed in as input parameter and allows empty string as input. In that case no password is applied to backup file.	**Type:** Weakness **Severity:** Low
010	Password present in the code	A component stores the admin password and a method was found where a pre-determined password might be allowed.	**Type:** Weakness **Severity:** Low

ID	Description	Assessment	Categorization
011	Many COMPUTE and PERFORM statements in COBOL source code contain non-trivial literal numeric values.	Single-digit values, such as counter increments and decrements, integer values of powers of 2, or year offsets, are acceptable as literal values. Literal values that might change in the future or whose meaning is not obvious create code management issues that could result in future errors.	**Type:** VVSG Non-conformity **Severity:** Low
012	Numerous COBOL source files contain more than 1000 lines of code.	The VVSG standard specifies limits for how much code a particular module should contain, but this requirement is intended for modern languages like C and C++. COBOL is a verbose language and organized differently than most other languages. The reviewer, therefore, does not expect COBOL code to adhere to those exact numbers of lines of code, but would recommend that overly large files (larger than the arbitrary limit of 1000 lines) be considered for reorganization in a future release.	**Type:** VVSG Non-conformity **Severity:** Low
013	Several FIPS 140-2 requirements are not implemented.	The following FIPS 140-2 implementation requirements are not met: - the power-on selftests and integrity tests (section 4.9 of [FIPS140-2]) - key zeroization (section 4.7.6 of [FIPS140-2]) - continuous test for random number generation (section 4.9.2 of [FIPS140-2]) - FIPS 186-4 RSA key generation (Appendix B.3 of [FIPS186-4] Those implementation requirements are not met by the code. However, whenever a cryptographic operation is invoked through by an external application we are unable to testify whether the FIPS requirements have been implemented or not.	**Type:** Weakness **Severity:** Medium

<table>
<tr><th>ID</th><th>Description</th><th>Assessment</th><th>Categorization</th></tr>
<tr><td>015</td><td>Some source code files contain more lines than recommended by VVSG.</td><td>VVSG requires half of source modules be 60 lines or less and allows a low percentage to be greater than 120 lines, but none over 240 lines.

Using lines in a file as a measure, the reviewer found the following line counts in non-COBOL source files:

<table><tr><th>Line Count</th><th>VVSG Limit</th><th>Actual</th></tr><tr><td>60</td><td>50%</td><td>91%</td></tr><tr><td>120</td><td>5%</td><td>75%</td></tr><tr><td>240</td><td>0%</td><td>52%</td></tr></table>
In order to conform to the VVSG recommendation, the reviewer recommends a future release include substantial reorganization of the code.</td><td>Type: VVSG Non-conformity
Severity: Low</td></tr>
<tr><td>016</td><td>Leap year calculated incorrectly.</td><td>In one location, Leap Year is calculated only by checking that the year is evenly divisible by 4. This omits the exceptions for years evenly divisible by 100 and 400. To identify a leap year it must be evenly divisible by 4; however, if the year can be evenly divided by 100, it is not a leap year, unless it is also evenly divisible by 400. The system does not accurately check for the latter instances; it only checks that the year is divisible by four, which is not accurate. The flaw will have no effect until 1-Mar-2100.</td><td>Type: Vulnerability
Severity: Low</td></tr>
<tr><td>017</td><td>Some source files contain auto-generated code which is not clearly or consistently marked.</td><td>Auto-generated code contained in numerous source files is not always clearly marked in the source, and the notation varies between source files. No standard artifact from a code generation process was found. A somewhat standardized set of comments was found.</td><td>Type: Vulnerability
Severity: Low</td></tr>
</table>

ID	Description	Assessment	Categorization
018	Source code contains some instances of GoTo statements.	VVSG states acceptable constructs are Sequence, If-Then-Else, Do-While, Do-Until, Case, and the General Loop (including the special case for loop). Do-While (False) constructs and intentional exceptions (used as GoTos) are prohibited. The reviewers found uses of GoTos. Uses such as “On Error GoTo Failure” may be justifiable, however, still represent a technical violation of the VVSG requirement.	**Type:** VVSG Non-conformity **Severity:** Low
019	Source code contains some instances where single alphabet variable names are used.	The VVSG requires that object, function, procedure, and variable names shall be chosen to enhance the readability and intelligibility of the program. The reviewers found cases where small variable names are used which do not enhance the readability but the inline comments provide help in such cases.	**Type:** VVSG Non-conformity **Severity:** Low
020	Source code contains an instance where input value is not checked for null or empty.	The VVSG requires that the code use uniform calling sequences and that all parameters be validated for type and range on entry into each unit, or the unit comments shall explicitly identify the type and range for the reference of the programmer and tester. This is true in many cases, but the reviewer found instances where input strings were used without validation.	**Type:** VVSG Non-conformity **Severity:** Low
021	No code to check for counter overflow found.	The VVSG requires that the code provide controls to prevent any vote counter from overflowing. The source code does not appear to check for arithmetic overflow when using values used to maintain ballot counts.	**Type:** VVSG Non-conformity **Severity:** Low
022	Source code uses “a = b ? c : d” syntax.	The VVSG requires minimal use of the “a = b ? c : d” syntax. Multiple instances of this syntax were found by the reviewers.	**Type:** VVSG Non-conformity **Severity:** Low
023	Source code contains assert() statements.	The VVSG requires assert() statements be removed from production code. The reviewer found multiple instances of these statements in the source code.	**Type:** VVSG Non-conformity **Severity:** Low

ID	Description	Assessment	Categorization
024	Use of weak random number generator (RNG) may lead to weak encryption keys.	A component uses the RNG to perform random number generation and key generation. This function is not recommended for cryptographic random-number generation and seeding with the system clock is not recommended. rand() is also not approved by NIST nor provided by a FIPS-validated module.	**Type:** VVSG Non-conformity **Severity:** Low
025	A component appears to not use cryptographic operations to protect data.	The source code analysis for a component that does not perform any cryptographic operations. It is always recommended to use cryptographic operations when storing confidential data on any portable device. The reviewer found that the firmware supports a simple checksum calculation and e-code for the data integrity protection. Checksums are good for accidental error checking, but insufficient for real data integrity protection. Malicious alteration of data can be done in a way to preserve a checksum value. NIST-approved hashing algorithms FIPS 180-4 SHS and FIPS 202 SHA-3 are recommended for data integrity protection.	**Type:** VVSG Non-conformity **Severity:** Low

ID	Description	Assessment	Categorization
026	A componenet uses weak digital signature generation method and a weak random number generator.	A component uses cryptographic algorithms provided by the FIPS validated module RSA BSAFE Crypto-C Micro Edition. The reviewer found that ECDSA is used for signature generation and verification, but SHA-1 is used when hashing the data in the digital signature. The reviewer found that the FIPS 186-2 General Purpose PRNG is used for random number generation (with SHA-1). This PRNG is not approved by NIST after 2015 and has been removed from FIPS 140 Annex C. The RSA BSAFE(R) Crypto-C Micro Edition V3.0 cryptographic library is an older version that supports cryptographic algorithms no longer approved by NIST. It is recommended this library be updated and only NIST and/or FIPS-approved algorithms be used.	**Type:** Weakness **Severity:** Low
027	Possibly inappropriate error handling.	The reviewer found many cases where try-catch is implemented with no arguments provided in the catch block. When this occurs, the catch block handles all potential errors with the same code. Using this type of catch block cannot find the exact type of error that has been thrown in the try block. Catching all exceptions can lead to a situation where the code handles an exception but does not handle the source of the problem.	**Type:** Vulnerability **Severity:** Low

ID	Description	Assessment	Categorization
028	A component uses a weak digital signature generation method.	The cryptographic algorithms are provided by the FIPS validated module RSA BSAFE Crypto-C Micro Edition. The reviewer found ECDSA uses SHA-1 for digital signature generation. SHA-1 is no longer approved by NIST for this purpose. The reviewer found that the SHA-256 is used for key generation and key derivation. While using SHA-256 for Password-Based Key Derivation Function (PBKDF) is a NIST-approved method, using SHA-256 for key generation is not an approved FIPS 140-2 Key Generation method. Weak keys can cause the NIST-approved algorithm to be weakened. FIPS 140-2 requires the key to be generated using an approved random number generation method. Using a hash value from a SHA-256 as the AES key to perform AES cryptographic operations weakens the security strength of the AES algorithm. The RSA BSAFE(R) Crypto-C Micro Edition V3.0 cryptographic library is an older version that supports cryptographic algorithms no longer approved by NIST. It is recommended this library be updated and only NIST and/or FIPS-approved algorithms be used.	**Type:** Weakness **Severity:** Low

Table 2: Summary of issues discovered during the static code analysis

Glossary

AES	Advanced Encryption Standard
AIMS	AutoMARK Information Management System
AM	Audit Manager
API	Application Programming Interface
ATI	Audio Tactile Interface
AVS	Accessible Voting Station
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CMVP	Cryptographic Module Validation Program
COTS	Commercial Off-The-Shelf
CRC	Cyclic Redundancy Check
CSP	Critical Security Parameter
CVE	Common Vulnerability and Exposures
CWE	Common Weakness Enumeration
DCF	Device Configuration File
DCM	Data Center Manager
ECDSA	Elliptic Curve Digital Signature Algorithm
EDM	Election Data Manager
EED	Election Event Designer
EMS	Election Management System
ESSIM	ES&S Ballot Image Manager
FIPS	Federal Information Processing Standard
HMAC	Hash Message Authentication Code
HPM	Hardware Programming Manager
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
IP	Internet Protocol
IV	Initialization Vector
LAN	Local Area Network
LDF	Log Data File
MCF	Machine Context File
NAS	Network Attached Storage
OMR	Optical Mark Recognition
OS	Operating System
PC	Personal Computer

PEB	Personal Electronic Ballot
PRNG	Pseudorandom Number Generator
RNG	Random Number Generator
RRH	Result Receiver Host
RSA	Rivest, Shamir, and Adelman
RTM	Result Transfer Manager
RTR	Results Tally and Reporting
SHA	Secure Hash Algorithm
SHS	Secure Hash Standards
TCP	Transmission Control Protocol
USB	Universal Serial Bus
VIF	Voter Information File
VVSG	Voluntary Voting System Guidelines

References

Documentation provided for the source code review included ES&S Unity product documentation, reports produced by Wyle Laboratories, and other publically available standards documents. The atsec source code review team also consulted other publically available documents listed in the last group.

ES&S Unity Election Management System (EMS) General Documentation

[EMSEPG] Unity EMS Election Programming Guide, California Use Procedures, Software Version 3.4.1.0, Revision 1.0, November 11, 2015

[ESSPDTP] ES&S Standards and Procedures, Personnel Deployment and Training Program, Document Revision 1.0, December 18, 2012

[EMSCUP] Unity 3.4.1.0 Election Management System California Use Procedures, Revision 3.0, October 2015

Automark 3410 Documentation

[AUTOJG] ES&S AutoMARK Jurisdiction Guide, Revision 13, June 30, 2011

[AUTOOSDS] ES&S AutoMARK Operating Software Design Specifications, Revision 8, July 1, 2013

[AUTOPDTR] ES&S AutoMARK Personnel Deployment and Training Requirements, Revision 9, August 12, 2013

[AUTOPWG] ES&S AutoMARK Poll Worker's Guide, Revision 14, June 30, 2011

[AUTOSHS] ES&S AutoMARK System Hardware Specification, Revision 8, August 12, 2013

[AUTOSIMG] ES&S AutoMARK System Installation and Maintenance Guide, Revision 8, August 12, 2013

[AUTOVG] ES&S AutoMARK Voter's Guide, Revision 13, August 12, 2013

Automark Information Management System (AIMS) Documentation

[AIMSEOG] AutoMARK Information Management System, Election Official's Guide, Revision 24, August 5, 2013

[AIMSSDS] AutoMARK Information Management System, Software Design Specifications, Revision 9, August 12, 2013

[AIMSSHS] AutoMARK Information Management System, System Hardware Specifications, Revision 8, August 5, 2013

[AIMSSOP] AutoMARK Information Management System, System Operations Procedures, Revision 8, August 12, 2013

DS 200 Precinct Scanner and Tabulator Documentation

[DS200HS] DS200 Engineering Hardware Specification, Document Revision 3.0, Hardware Revision 1.3, November 15, 2013

[DS200SDS] ES&S Software Design Specifications, DS200, Unity v. 3.4.0.1, November 15, 2013

[DS200MG] ES&S DS200 Maintenance Guide, Document Version 3.2, Firmware Version 1.7, February 3, 2014

[DS200OG] ES&S DS200 Operator's Guide, Document Version 5.2, Firmware Version 1.7, February 3, 2014

DS 850 High-Speed Scanner and Tabulator Documentation

[DS850HS] DS850 Engineering Hardware Specification, Document Revision 1.0, Hardware Revision 1.0, July 24, 2013

[DS850SDS] ES&S Software Design Specifications, DS850, Unity 3.4.1.0, November 15, 2013.
[DS850MG] ES&S DS850 Maintenance Guide, Document Version 2.1, Firmware Version 2.9, January 29, 2014
[DS850OG] ES&S DS850 Operator Guide, Document Version 4.2, Firmware Version 2.9, January 29, 2014

ES&S Unity Audit Manager (AM) Documentation

[AMSDS] ES&S Software Design Specifications, Audit Manager, Unity v. 3.4.1.0, November 15, 2013
[AMUSER] ES&S Audit Manager User's Guide, Document Version 2.1, Firmware Version 7.8, January 29, 2014

ES&S Unity Election Data Manager (EDM) Documentation

[EDMSDS] ES&S Software Design Specifications, Election Data Manager (EDM), Unity v. 3.4.1.0, November 15, 2013
[EDMOP] ES&S Election Data Manager Operator's Guide, Document Version 2.3, Software Version 7.8, January 29, 2014

ES&S Unity Election Reporting Manager (ERM) Documentation

[ERMSDS] ES&S Software Design Specifications, Election Reporting Manager (ERM), Unity v. 3.4.1.0, November 15, 2013
[ERMUSER] ES&S Election Reporting Manager User's Guide, Document Version 5.1, Software Version 7.9, January 29, 2014

ES&S Ballot Image Manager (ESSIM) Documentation

[ESSIMSDS] ES&S Software Design and Specification, ES&S Ballot Image Manager (ESSIM), Unity v. 3.4.1.0, November 15, 2013
[ESSIMUSER] ES&S Image Manager User's Guide, Document Version 3.1, Software Version 7.8, January 30, 2014

Hardware Programming Manager (HPM) Documentation

[HPMSDS] ES&S Software Design and Specification, Hardware Programming Manager (HPM), Unity v. 3.4.1.0, November 15, 2013
[HPMUSER] ES&S Hardware Programming Manager User's Guide, Document Version 5.1, Software Version 5.9, January 29, 2014

Log Monitor Service Documentation

[LMSDS] ES&S Software Design and Specifications, Log Monitor, Unity v. 3.4.1.0, July 8, 2013
[LMUSER] ES&S Log Monitor User's Guide, Document Version 2.1, Software Version 1.0, January 29, 2014

M 100 Precinct Scanner and Tabulator Documentation

[M100SHS] ES&S System Hardware Specification, Model 100, Hardware Revision 1.3, July 11, 2013
[M100SDS] ES&S Software Design Specifications, Model 100, Unity v.3.4.1.0, November 15, 2013
[M100MG] ES&S M100 Maintenance Guide, Document Version 2.1, Firmware Version 5.4, January 30, 2014

[M100OG] ES&S M100 Operator Guide, Document Version 2.1, Firmware Version 5.4.4.5, January 29, 2014

M650 Central Scanner and Tabulator Documentation

[M650SHS] ES&S System Hardware Specification, Model 650, Hardware Revision 1.2, July 11, 2013

[M650SDS] ES&S Software Design Specifications, Model 650, Unity v. 3.4.1.0, November 15, 2013

[M650SMM] ES&S Model 650 System Maintenance Manual, Document Version 2.1, Firmware Version 2.2, January 30, 2014

[M650SOG] ES&S Model 650 System Operators Guide, Document Version 2.1, Firmware Version 2.2, January 30, 2014

Wyle Laboratories Reports

[WTEST] National Certification Test Report for Certification Testing of the Election Systems & Software Unity 3.4.1.0 Voting System, Report Number T71220.01-01, Revision B, March 31, 2014

[ESSUSOC] Scope of Certification, Election Systems & Software Unity 3.4.1.0, Wyle Laboratories, April 4, 2014

California Application for Approval of a Voting System

[CAAVS] California Application for Approval of a Voting System, Unity 3.4.1.0, February 6, 2015

Public Documents

[CERTC] Seacord, Robert C., The CERT C Secure Coding Standard, Addison-Wesley, Upper Saddle River, NJ, 2009

[CERTJ] Long, et al., The CERT Oracle Secure Coding Standard for Java, Addison-Wesley, Upper Saddle River, NJ, 2012

[FIPS140IG] 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

[FIPS140-2] 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

[FIPS140-2A] 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

[FIPS180-4] 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

[FIPS186-4] 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

[FIPS197] National Institute of Standards and Technology, FIPS 197 Advanced Encryption Standard, November 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[FIPS198-1] 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

[NIST57] 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

[NIST90A] 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

[NIST131A] National Institute of Standards and Technology, NIST Special Publication 800-131A, Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths, January, 2011, http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf

[VVSG1] United States Election Assistance Commission, 2005 Voluntary Voter System Guidelines, Volume 1, Version 1.0, 2005 http://www.eac.gov/testing_and_certification/voluntary_voting_system_guidelines.aspx

[VVSG2] United States Election Assistance Commission, 2005 Voluntary Voter System Guidelines, Volume 2, Version 1.0, 2005 http://www.eac.gov/testing_and_certification/voluntary_voting_system_guidelines.aspx

