FREEMAN, CRAFT, MCGREGOR GROUP

# Source Code Review

## Dominion Democracy Suite 4.14-A Voting System

Report Date: 2014-07-14

Version: 1.6

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

## 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 Dominion Democracy Suite 4.14-A Voting System. It 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.

The most significant vulnerability found was that the voting system generates weak keys used for a variety of purposes. A knowledgeable attacker could crack these keys in a matter of hours with commodity resources.

Moderate vulnerabilities found include:

- complex control flow
- use of cryptographic algorithms and modes considered insecure by NIST
- implementation and design are inconsistent with respect to cryptography
- iButton functionality may be bypassed through developer debugging code
- privilege escalation issues
- hard-coded encryption keys
- no detection of overflows in vote counting arithmetic

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 Chapter 3 for all findings.

# 2 Introduction

This report was prepared by atsec information security corporation to review aspects of the security and integrity of the Dominion Democracy Suite 4.14-A 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 Dominion Democracy Suite 4.14-A (EAC Certification Number: DemSuite-4-14-A) 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:

- Jeremy Powell
- Hedy Leung
- King Ables
- 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 Dominion Voting Systems Democracy Suite Version 4.14-A Voting System (hereafter referred to as the “voting system” or simply as the “system”) is a paper ballot-based, optical scan voting system. The system hardware consists of four major components:

- The Election Management System (EMS)
- ImageCast Evolution (ICE) precinct scanner with optional ballot marking capabilities
- ImageCast Central (ICC) central count scanner
- Adjudication system

atsec performed the code review on the basis of the Statement of Work between Freeman, Craft, McGregor Group Inc. #13S52044 with the State of California, which states that review includes evaluating the security of the system as it is allowed to be configured for use by the State of California (hereafter referred to as “the California configuration”). The threat model below defines the scope of this examination.

## 2.2 Inputs

The reviewers were provided with 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 chapter.

The reviewers were also provided with the source code for the following components:

- The Election Management System (EMS)
  - EMSAuditModuleSourceCode_package_1.0.1
  - EMSAuditModule 1.0.1 Outputs
  - EMSSourceCode_package_4.14.23

    - EMS Outputs
- ImageCast Evolution (ICE)
    - ICE Outputs
    - ICE 4.14.10
- ImageCast Central (ICC)
    - ICC Outputs
    - ICC_4.14.4_130321
- Adjudication system
    - Adj_1.0.14.17603_Source_20130523
    - Adjudication Outputs

## 2.3 Threat Model

This assessment is centered on the threat model prescribed 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 used the following methodology for the source code review.

### 2.4.1 Published vulnerabilities

The reviewers searched the MITRE Common Vulnerability and Exposures (CVEs) list and the Common Weakness Enumeration (CWEs) list to identify vulnerabilities that affect the system. Although these lists may not have entries for the voting system itself, constituent software that the voting system uses may contain vulnerabilities. The review team identified software 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:

- 2005 Voluntary Voting System Guidelines [VVSG1], [VVSG2] and supplemental interpretation statements found at: http://www.eac.gov/testing_and_certification/request_for_interpretations1.aspx
- Dominion Democracy  C/C++ Coding Standard [DSCODE]

Note that [DSCODE] is not a “published, reviewed, and industry-accepted” standard, so is ineligible as an exemption for any requirement in [VVSG2] Section 5.

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#). The system does not appear to contain Java source code, but the standard still provided useful input into the analysis of the code quality.

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

### 2.4.3 Design

The source code review team used the usage and installation guidance, 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 hardest security mechanism to break directly, misuse of cryptographic primitives can render that protection weak or non-existent. The review team identified use sites of cryptography 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 with 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 may have poor code quality and accesses sensitive information such as authentication credentials might be a good place to hide a back door. The reviewers treated 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 Chapter 3. Each finding contains:

- 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 nonconformity in the code quality standards. Nonconformities 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.

# 3 Findings

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 |
|---|---|---|---|
| 1 | Catchall catch-blocks in try-catch statements. | Catchall catch-blocks may not handle some exceptions appropriately. Maliciously crafted input may cause denial of service or otherwise undefined behavior. | **Type:** Weakness<br>**Severity:** Low |
| 2 | Try-catch statements do not handle all potential exceptions. | Uncaught exceptions are thrown to the calling function. Maliciously crafted input may cause a denial of service. | **Type:** Weakness<br>**Severity:** Low |
| 3 | Non-descriptive constant names. | Many constants in the source code have identifiers that describe the value of the constant rather than the concept it represents or its usage. No flaws were identified, but the style is not robust to implementation flaws. It also makes source code review difficult. | **Type:** Weakness; VVSG nonconformity<br>**Severity:** Low |
| 4 | Non-descriptive comments. | Comments intended to describe the code often re-state the code rather than explain what it is doing semantically. No flaws were identified, but the style is not robust to implementation flaws. It also makes source code review difficult. | **Type:** Weakness; VVSG nonconformity<br>**Severity:** Low |
| 5 | Inconsistent comments. | Comments were observed to be copy-and-pasted without further updates, leaving them inconsistent. No flaws were identified, but the style is not robust to implementation flaws. It also makes source code review difficult. | **Type:** Weakness; VVSG nonconformity<br>**Severity:** Low |

| ID | Description | Assessment | Categorization |
|---|---|---|---|
| 6 | Switch statements do not have default cases. | When no default cases exist, control may pass through the switch statement without proper processing. No flaws were identified, but the style is not robust to implementation flaws. | **Type:** Weakness; VVSG nonconformity<br>**Severity:** Low |
| 7 | Single-statement control structures do not enclose statement with braces. | Single-statement control structures are not robust to development regressions. No flaws were identified, but the style is not robust to implementation flaws.<br><br>The reviewers explicitly relate this weakness to the recent Apple GOTO fail flaw. See http://www.wired.com/threatlevel/2014/02/gotofail/ for further details. | **Type:** Weakness; VVSG nonconformity<br>**Severity:** Low |
| 8 | Code extends beyond 80-character width limit specified by VVSG. | The source code often exceeds the limit by only a few characters. In some more rare cases it extends further. The reviewers do not consider this a security related issue and did not find that it detracts from readability. | **Type:** VVSG nonconformity<br>**Severity:** Low |
| 9 | Member variables are not initialized in the constructor. | Some classes were identified that did not initialize member variables. | **Type:** Weakness; Dominion Style Guide nonconformity<br>**Severity:** Low |
| 10 | The claim of FIPS compliance is not confirmed. | ICE and ICC use the OpenSSL module for cryptographic algorithms. The version of OpenSSL used by ICE cannot be confirmed to be FIPS 140-2 validated by the CMVP. The ICC uses a validated OpenSSL module. | **Type:** Weakness<br>**Severity:** Low |

| ID | Description | Assessment | Categorization |
|---|---|---|---|
| 11 | No password guessing prevention. | The password verification mechanism in ICC does not slow the rate of password guesses nor does it prevent access after a certain number of attempts. | **Type:** Vulnerability<br>**Severity:** Low<br>Might be exploited by a poll worker or elections official insider. |
| 12 | Some modules do not include header information required by VVSG. | Modules are required to have header information including the purpose of the unit and how it works, other units called, input parameter description, output description, file references, global variables, and revision record. | **Type:** VVSG nonconformity<br>**Severity:** Low |
| 13 | Some variable declarations do not include explanatory comments as required by VVSG. | VVSG states that all variables shall have comments at the point of declaration clearly explaining their use. | **Type:** VVSG nonconformity<br>**Severity:** Low |
| 14 | Mixed-mode operations exist counter to VVSG requirement. | VVSG states mixed-mode operations should be avoided or at least clearly explained if necessary. Several instances were found that did not provide any rationale. | **Type:** VVSG nonconformity<br>**Severity:** Low |
| 15 | Complex branching structure. | The developers use an uncommonly complex branching style for handling errors. No flaws were identified, but the style is not robust to implementation flaws. It also makes source code review difficult. | **Type:** Weakness<br>**Severity:** Medium |

| ID | Description | Assessment | Categorization |
|---|---|---|---|
| 16 | Incorrect HMAC invocation. | Although the documentation describes the use of an HMAC for integrity protection, SHA-256 is used instead. Inconsistency with documentation is a low severity, however since it relates to cryptography, it has a higher potential impact on the security of the system. | **Type:** Weakness<br>**Severity:** Medium |
| 17 | User is allowed to select weak RSA keys. | The system allows the user to select the size of RSA keys, including 1024 bits. NIST no longer considers 1024-bit RSA keys to be secure. | **Type:** Weakness<br>**Severity:** Medium<br>Might be used even accidentally by an elections official insider and then exploited by a poll worker or elections official insider. |
| 18 | Use of SHA-1 in RSA signature generation | NIST no longer considers SHA-1 as an appropriate cryptographic hash for digital signature generation after 2014-01-01. | **Type:** Weakness<br>**Severity:** Medium |
| 19 | Use of MD5 | NIST no longer considers MD5 as an appropriate cryptographic hash in FIPS 140-2. | **Type:** Weakness<br>**Severity:** Medium |
| 20 | Use of RSA encryption and decryption for use other than key distribution. | The system protects some files using RSA. Encryption and decryption with RSA is not considered appropriate by NIST unless used for key distribution. | **Type:** Weakness<br>**Severity:** Medium |

| ID | Description | Assessment | Categorization |
|---|---|---|---|
| 21 | Confidentiality and integrity protection implemented by code is inconsistent with documentation. | The description of the use of cryptography in the documentation is inconsistent with the implementation. For example, documentation states audio files are signed but code indicates they are not. Numerous locations in documentation refer to an AES key of 256 bits but the code shows it to be 128 bits. Inconsistency with documentation is a low severity, however since it relates to cryptography, it has a higher potential impact on the security of the system. | **Type:** Weakness<br>**Severity:** Medium |
| 22 | Unconventional authentication scheme. | The iButton uses an unconventional authentication scheme for authenticating users. It is unclear if this scheme is cryptographically sound. | **Type:** Potential vulnerability<br>**Severity:** Medium<br>Might be exploited by a poll worker or elections official insider. |
| 23 | iButton functionality may be bypassed. | Bypassing the iButton functionality requires write access to the configuration files of the ICC. This is a debugging development tool, but is active in the production code. This would allow an attacker to read or alter anything protected by the principle encryption and integrity keys. | **Type:** Vulnerability<br>**Severity:** Medium<br>Might be exploited by a poll worker, elections official insider, or vendor insider. |
| 24 | Privilege escalation from Administrator to Technician roles. | One instance of possible privilege escalation was identified. | **Type:** Vulnerability<br>**Severity:** Medium<br>Might be exploited by a poll worker or elections official insider. |

| ID | Description | Assessment | Categorization |
|---|---|---|---|
| 25 | Hard-coded encryption keys. | Hard coded encryption keys expose the system to insider threat agents with expert knowledge or attackers who decompile and retrieve the keys from the executable. | **Type:** Vulnerability<br>**Severity:** Medium<br>Might be exploited by a poll worker, elections official insider, or vendor insider. |
| 26 | No detection of overflows in arithmetic performed on vote counters. | A 32-bit signed integer requires over 2 billion votes to overflow a counter. While statistically unlikely[6], if an overflow were to occur, election results could be affected in unpredictable ways. | **Type:** Weakness, VVSG nonconformity<br>**Severity:** Medium |
| 27 | Sensitive keys are stored on disk unencrypted. | These keys are used to protect keys used to encrypt and integrity protect the election definition and results. The keys are stored on the disk without their own protection. Gaining access to these files will grant logical access to the election definitions and results. Access controls on the underlying file system mitigate this threat. | **Type:** Vulnerability<br>**Severity:** Medium |
| 28 | Hard-coded key used to encrypt sensitive election information in the adjudication system. | Hard-coded encryption keys expose the system to insider threat agents with expert knowledge or attackers who decompile and retrieve the keys from the executable. | **Type:** Vulnerability<br>**Severity:** Medium |

[6] Page 85 of [VVSG2] explicitly states that assuming the counter size is large enough that the overflow value will never be reached is not adequate.

| ID | Description | Assessment | Categorization |
|---|---|---|---|
| 29 | Poor quality keys | Keys generated by the voting system have a relatively low amount of entropy making them easier to crack. It would take an insider with expert knowledge or someone who can decompile the executable binaries to exploit this vulnerability. The exploit could then be easily transferred to threat agents with no expertise. Considering the fact that almost all cryptographic functions of the system hinges on the strength of the keys, this is considered a high severity vulnerability. | **Type:** Vulnerability<br>**Severity:** High<br><br>Might be exploited by any individual, voter or non-voter, who gains access to a compact flash memory card containing election data. |

# Glossary

| | |
|---|---|
| **AES** | Advanced Encryption Standard |
| **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 |
| **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 |
| **EED** | Election Event Designer |
| **EMS** | Electronic Management System |
| **FIPS** | Federal Information Processing Standard |
| **HMAC** | Hash Message Authentication Code |
| **HTTP** | Hyper Text Transfer Protocol |
| **HTTPS** | Hyper Text Transfer Protocol Secure |
| **ICC** | ImageCast Central |
| **ICE** | ImageCast Evolution |
| **ICP** | ImageCast Precinct |
| **IP** | Internet Protocol |
| **IV** | Initialization Vector |
| **LAN** | Local Area Network |
| **LDF** | Log Data File |
| **MCF** | Machine Context File |
| **NAS** | Network Attached Storage |
| **OS** | Operating System |
| **PC** | Personal Computer |
| **RRH** | Result Receiver Host |
| **RSA** | Rivest, Shamir, and Adelman |
| **RTM** | Result Transfer Manager |
| **RTR** | Results Tally and Reporting |

| | |
|---|---|
| **SHA** | Secure Hash Algorithm |
| **TCP** | Transmission Control Protocol |
| **USB** | Universal Serial Bus |
| **VIF** | Voter Information File |
| **VVSG** | Voluntary Voter System Guidelines |

# References

Documentation provided for the source code review included Dominion Democracy product documentation, reports produced by Wyle Laboratories, as well as other publically available standards documents. The atsec source code review team also consulted other publically available documents listed in the last group.

## Dominion Democracy General Documentation

| | |
|---|---|
| [DSOV] | 2.02 - Democracy Suite System Configuration Overview, Version: 1.2.0::274, September 20, 2013. |
| [DSSEC] | 2.06 - Democracy Suite System Security Specification, Version: 1.1.0::339, September 27, 2013. |
| [DSTR] | 2.10 - Democracy Suite Personnel Deployment and Training Requirements, Version: 1.1.0::53, April 9, 2013. |
| [DSCM] | 2.11 - Democracy Suite Configuration Management Process, Version: 1.2.0::177, April 9, 2013. |
| [DSQA] | 2.12 - Democracy Suite Quality Assurance Program, Version: 1.2.0::80, October 16, 2012. |
| [DSUSE] | Dominion Voting Systems' Democracy Suite Voting Systems Use Procedures, Version: 4.14::6, November 2, 2013. |
| [DSTEST] | Democracy Suite Readiness Test Procedure, Version: 1.2.0::96, August 23, 2013. |
| [DSPRINT] | Dominion Voting Systems ImageCast Printing Specification, Version: 1.0.0::18, May 16, 2013. |
| [DSCODE] | Dominion Democracy  C/C++ Coding Standard, Version: 1.0.0::8, July 27, 2012. |

## Dominion Democracy EMS Documentation

| | |
|---|---|
| [EMSFUN] | 2.03 - Democracy Suite EMS Functionality Description, Version: 1.1.0::238, September 20, 2013. |
| [EMSSDS] | 2.05 - EMS Software Design and Specification, Version: 1.0.0::209, September 20, 2013. |
| [EMSOP] | 2.08 - EMS System Operation Procedures, Version: 1.2.0::483, September 20, 2013. |
| [EMSMAINT] | 2.09 - EMS System Maintenance Manual, Version: 1.0.0::50, October 16, 2012. |
| [EMSASUG] | EMS Audio Studio User's Guide, Version: 1.0.0::32, September 5, 2013. |
| [EMSDTUG] | EMS Election Data Translator User's Guide, Version: 1.0.0::32, August 23, 2013. |
| [EMSEEDUG] | EMS Election Event Designer User's Guide, Version: 1.0.1::173, September 20, 2013. |
| [EMSRTRUG] | Results Tally and Reporting User's Guide, Version: 1.0.0::115, September 20, 2013. |
| [EMSDBSEC] | EMS SQL Server and Database Security Procedures, Version 1.0.0::2, November 1, 2013. |
| [EMSASUG2] | Democracy Suite EMS Audio Studio Users Guide, Version: 1.0.0::27, February 8, 2013. |
| [EMSBODUG] | Democracy Suite EMS Ballot On Demand Users Guide, Revision: 1.0.0, May 4, 2010. |
| [EMSEDTUG] | Democracy Suite EMS EDT Users Guide, Version: 1.0.0::23, April 22, 2013. |
| [EMSEDTUG2] | Democracy Suite EMS EED Users Guide, Version: 1.0.1::154, May 16, 2013. |
| [EMSRTM] | Democracy Suite EMS Results Transfer Manager, Version: 1.0.0::8, February 7, 2013. |
| [EMSRTRUG2] | Democracy Suite EMS RTR Users Guide, Version: 1.0.0::91, May 16, 2013. |

| | |
|---|---|
| [EMSCOMP] | Democracy Suite Election Management System Computer Awareness and Data Security Compliance Statement. |
| [EMSEULA] | END USER LICENSE AGREEMENT FOR DEMOCRACY SUITE ELECTION MANAGEMENT SYSTEM. |
| [EMSCADSCS] | Democracy Suite EMS CADSCS. |

### Dominion Democracy ICC Documentation

| | |
|---|---|
| [ICCFUN] | 2.03 - ImageCast Central Functionality Description, Version: 1.1.0::78, August 26, 2013. |
| [ICCSDS] | 2.05 - ImageCast Central Software Design and Specification, Version: 1.0.0::37, August 30, 2013. |
| [ICCOP] | 2.08 - ImageCast Central System Operation Procedures, Version: 1.1.0::111, August 2, 2013. |
| [ICCSCANUM] | Canon DRX10C User Manual. |
| [ICCSCANDR] | ImageCast Central - Canon DR-X10C Scanner Driver Installation, Version 1.2.7, August 14, 2013. |
| [ICCAPP] | ImageCast Central Application Installation, Version: 2.1.12::18, October 31, 2013. |
| [ICCUG] | ImageCast Central User's Guide, Version: 1.0.0::29, August 16, 2013. |
| [ICCSCANRG] | Canon DR-X10C Reference Guide. |

### Dominion Democracy ICE Documentation

| | |
|---|---|
| [ICEFUN] | 2.03 - Democracy Suite ImageCast Evolution Functionality Description, Version: 1.2.0::72, September 20, 2013. |
| [ICEHWS] | 2.04 - ImageCast Evolution Tabulator System Hardware Specification, Version: 1.2.0::291, September 20, 2013. |
| [ICEHWC] | 2.04.1 - ImageCast Evolution System Hardware Characteristics, Version: 1.2.0::84, October 16, 2012. |
| [ICESDS] | 2.05 - ImageCast Evolution Software Design And Specification, Version: 1.0.0::102, February 12, 2013. |
| [ICEOP] | 2.08 - ImageCast Evolution System Operation Procedures, Version: 1.0.0::134, September 20, 2013. |
| [ICEMAINT] | 2.09 - ImageCast Evolution System Maintenance Manual, Version: 1.1.0::115, October 16, 2012. |
| [ICEFWINST] | ImageCast Evolution Firmware Installation Procedure, Version: 1.0.0::30, December 26, 2012. |
| [ICESWINST] | ImageCast Evolution Software Installation, Update and Verification Procedures, Version: 1.0.0::31, October 23, 2013. |
| [ICEGUIDE] | ImageCast Evolution Technical Guide, Version: 1.0.0::66, July 18, 2013. |

### Dominion Democracy Adjudication Documentation

| | |
|---|---|
| [ADJFUN] | 2.03 - Democracy Suite Adjudication Functionality Description, Version: 1.0.0::16, November 1, 2013. |
| [ADJSDS] | 2.05 - Adjudication Software Design and Specification, Version: 1.0.0::19, November 1, 2013. |
| [ADJOP] | 2.08 - Adjudication System Operation Procedures, Version: 1.0.0::23, November 1, 2013. |
| [ADJMAINT] | 2.09 - Adjudication System Maintenance Manual, Version: 1.0.0::4, June 27, 2013. |

## Wyle Laboratories Reports

| | |
|---|---|
| [WPLAN] | 2013-2-12-Dominion-Democracy-Suite4_14-Test-Plan-Rev-A, Test Plan T70251.01-01, Rev. A, February 6, 2013. |
| [WDEFECT] | Democracy Suite 4.14 Defects Report - Final, November 8, 2013. |
| [WTEST] | Certification Testing of the Dominion Democracy Suite version 4.14 Voting System, Report No. T70251.01-01, July 9, 2013. |
| [WCODE] | EMS Audit Module CA SQL Scripts and Source Code Analysis, Report No. T58965.02, November 4,2013. |

## 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. |
| [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. |
| [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. |
| [VVSG2] | United States Election Assistance Commission, 2005 Voluntary Voter System Guidelines, Volume 2, Version 1.0, 2005. |