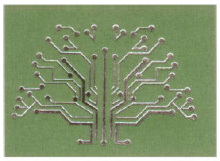

FREEMAN, CRAFT, MCGREGOR GROUP

Source Code Review

Dominion Democracy Suite 4.14-A.1 Voting System with Adjudication Version 2.4

Report Date: 2014-11-14

Version: 1.04

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

©2014 atsec information security corporation

Trademarks

atsec and the atsec logo is a registered trademark of atsec information security corporation.

The Freeman, Craft, McGregor Group (FCMG) logo is a trademark of FCMG.

Dominion Democracy, Democracy Suite Election Management System, Election Event Definition, Results Tally and Reporting, and ImageCast are trademarks of Dominion Voting Systems, Inc.

Microsoft, Microsoft SQL Server, Microsoft .NET, Windows 2007 and Visual C# are registered trademarks of Microsoft Corporation.

MITRE is a registered trademark of The MITRE Corporation.

IText is a registered trademark by iText Group NV.

Table of Contents

©2014 atsec information security corporation

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.1 Voting System with Adjudication Version 2.4.

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

The atsec team identified sixteen potential vulnerabilities in the system; all were categorized as having low levels of severity. All but two of the sixteen potential vulnerabilities involve non-conformances to one of the following coding or cryptographic standards:

- 2005 Voluntary Voting System Guidelines (specifically sections 5 and 7 of Volume I and section 5 of Volume II)
- StyleCop
- FIPS 140-2

The introduction to this report describes the basis of the source code review performed. In Chapter 3, we present the detailed results of the analysis.

©2014 atsec information security corporation

2 Introduction

The goal of this project was to provide test support services to assist the California Secretary of State (SOS) with the re-evaluation of Dominion Democracy Suite 4.14-A.1 Voting System, specifically the new Adjudication Version 2.4 component for its suitability for use in the State of California in accordance with Elections Code sections 19001 et seq.

This report has been prepared in support of a contract awarded to Freeman, Craft, McGregor Group, Inc. by atsec information security corporation as a result of the review of the security and integrity of the revised component, Adjudication System Version 2.4, a part of the Dominion Democracy Suite 4.14-A.1 Voting System. [1]

The source code review was performed by the following atsec consultants:

- Hedy Leung
- King Ables
- Lou Losee
- Swapneela Unkule

These individuals have prior experience of testing voting systems, and have not been involved in the federal level testing for the EAC of the subject voting system.

2.1 Scope and Basis

The Dominion Voting Systems Democracy Suite Version 4.14-A.1 Voting System with Adjudication Version 2.4 (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. #14S52049 with the State of California, which states that review includes evaluating the security of the Adjudication 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 given in section 2.3 below describes the basis of atsec's examination.

2.2 Inputs

The reviewers were provided with a set of documents associated with the system that were used to support the results described in this report. These documents are listed in the References chapter below.

These documents were examined during the source code review in order to understand the voting system's architecture and design and to support the identification of any discrepancies between the documentation and the source code.

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

- Adjudication Services (server) version 2.4.1.3201
- Adjudication (client) version 2.4.1.3201

[1] The system was previously examined by the EAC with certification ID: EAC Certification Number: DemSuite-4-14-A.1:
http://www.eac.gov/testing_and_certification/certified_voting_systems.aspx#Dominion_DemocracySuite414A.1Mod

- EMS Adjudication Services (Integration point with EMS) version 4.14.37

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 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:

- **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.

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

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 CVE database for potential vulnerabilities in the system. Although these lists may not have entries for the voting system itself, constituent software that the voting system uses may contain vulnerabilities. For the current scope of project, the review team identified that the Adjudication System is based on a C#/.NET environment and conducted searches for vulnerabilities related to these components. Searches for vulnerabilities for C#, .NET, xca, BouncyCastle, iTextSharp, and System.Security.Cryptography identified only one potential vulnerability that might pertain to the software under review.

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 represent vulnerabilities, it is a weakness and may lead to the introduction of vulnerabilities.

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

- StyleCop Coding Standard found at: https://stylecop.codeplex.com
- The CERT Oracle Secure Coding Standard for Java [CERTJ]

The Adjudication System does not contain Java source code, but C# has similarities to Java, so the Java coding standard still provides useful input into the analysis of code quality.

2.4.3 Design

The source code review team utilized the provided usage and installation guidance, source code, and any other provided material as well as publicly available information in order to construct an understanding of the architecture and design of the voting system. This included discovering the external interfaces and their associated 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 provided design description also provided a mapping of the high-level features and interfaces of the product to the features and interfaces implementation.

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 by brute-force, misuse of cryptographic primitives or implementation errors 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. The cryptographic primitives used in the source code are AES, HMAC-SHA-256, SHA-1, MD5, and RSA key generation.

Note that in a code review, there is no effective way to test the correctness of implementation of any cryptographic algorithm. We recommend that all cryptographic algorithms be functionally tested, such as with the Cryptographic Algorithm Validation System (CAVS) test which is part of the NIST Cryptographic Algorithm Validation Program.

2.4.5 Backdoors

Those with malicious intent who also have access to the Adjudication System during development may be able to place backdoors into the source code so that they could gain unauthorized access to the Adjudication System during operation.

Backdoors are extremely hard to find because a seasoned programmer can obfuscate code to look benign. The atsec team would like to stress that, when facing a competent and sufficiently motivated malicious developer, it is extremely difficult to prove that all backdoors in a system have been identified. The famous Turing award lecture by Ken Thompson in 1984 entitled *Reflections on Trusting Trust* [TRUST] demonstrated how fundamentally easy it is to undermine all security mechanisms when the developers cannot be trusted. This voting system is no exception. The current scope of the project is the Adjudication System, which consists of total 617 C# source files.

A full backdoor analysis is also impractical in a short period of time because a deep understanding of the data structures and code design is required to be able to recognize functionality that is out of place. Penetration test on a running system and observation of data modification and movement is also helpful. For this report, the reviewers are only able to examine the source code and look for signs of obfuscation or strange functionality.

The review team marked the areas of poor code quality and use of cryptographic operations for further scrutiny. For example, a particular area of code that has poor code quality and accesses sensitive information such as authentication credentials is identified as a likely candidate for a hidden backdoor. The reviewers treated such areas by considering the threat of introduced backdoors in addition to unintentional implementation flaws.

2.4.6 Summary of the results

A summary of the results is listed in Chapter 3. Each result contains:

- A description of the identified potential vulnerability or weakness.
- An assessment of what threats are involved in the possible exploitation of the vulnerability or weakness.
- A categorization of the result, 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.
    Potential vulnerabilities cannot be fully verified, but one or more conditions for the vulnerability have been observed.
  - A vulnerability in the source code.
    The reviewers have either shown or referenced other parties who have asserted the identified vulnerability to be exploitable.
- A severity level assigned to the result, which can be one of:
  - Low severity.
    Low severity implies the impact to the product is low, is 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.
  - Medium severity.
    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.
  - High severity.
    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.

©2014 atsec information security corporation

3 Results

The following table summarizes the results that arose from the source code review team's assessment of the Adjudication component system. Potential exploitation of a weakness or vulnerability and type of attacker is noted where applicable.

Table 1: Summary of Potential Vulnerabilities in the Adjucation System

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 **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 **Severity:** Low
3	Switch statements do not have default cases.	When no default cases exist, control may pass through the switch statement without proper processing.	**Type:** Weakness; VVSG nonconformity **Severity:** Low
4	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 **Severity:** Low
5	Initialize every variable upon declaration, and comment its use.	Instances were found where variables are not initialized.	**Type:** VVSG nonconformity **Severity:** Low
6	Member variables of a class must be initialized in the class constructor(s), either directly or indirectly	Instances were found where the member variables are not initialized.	**Type:** VVSG nonconformity **Severity:** Low
7	Mixed-mode operations exist counter to VVSG requirement.	VVSG states mixed-mode operations should be avoided or at least clearly explained if necessary, instances of mixed-mode operations were found.	**Type:** VVSG nonconformity **Severity:** Low
8	No detection of overflows in arithmetic performed on vote counters.	The source code does not appear to check for arithmetic overflows anywhere within the source tree.	**Type:** Weakness, VVSG nonconformity **Severity:** Low

©2014 atsec information security corporation

ID	Description	Assessment	Categorization
9	Restriction on code size by VVSG requirement stating: no more than 50% of all modules exceeding 60 lines in length, no more than 5% of all modules exceeding 120 lines in length, and no modules exceeding 240 lines in length.	For the current scope of code, the reviewer found that 435 files (70%) are greater than 60 lines (VVSG limit is 50%), 267 files (43%) are greater than 120 lines (VVSG limit is 5%), and 139 files (23%) are greater than 240 lines (VVSG limit is 0%).	**Type:** VVSG nonconformity **Severity:** Low
10	SA1501: A statement that is wrapped in opening and closing curly brackets must be written on a single line.	An instance was found where the statement and the curly braces are on the same line.	**Type:** StyleCop nonconformity **Severity:** Low
11	SA1124: Do not place a region anywhere within the code.	Code containing '#region' was found.	**Type:** StyleCop nonconformity **Severity:** Low
12	SA1120: When the code contains a C# comment it must contain text.	An instance was found where only comment '//' was present without any text.	**Type:** StyleCop nonconformity **Severity:** Low
13	SA1122: The code cannot contain empty strings.	Two instances were found where a variable was set to "".	**Type:** StyleCop nonconformity **Severity:** Low
14	The MD5 is not a FIPS Approved algorithm and should not be used.	One instance of use of MD5 was found.	**Type:** Potential vulnerability, FIPS nonconformity **Severity:** Low
15	The SHA-1 and MD5 are not FIPS Approved Key Generation methods and should not be used.	One instance of use of SHA-1 and MD5 for key generation was found.	**Type:** Potential vulnerability, FIPS nonconformity **Severity:** Low
16	The Rijndael implementation in Microsoft Cryptographic library is in non-conformance to FIPS-197, AES implementation should be used instead.	Several uses of the Rijndael algorithm were found.	**Type:** Weakness, FIPS nonconformity **Severity:** Low

©2014 atsec information security corporation

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
NIST	National Institute of Science and Technology
OS	Operating System
PC	Personal Computer
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
TCP	Transmission Control Protocol
USB	Universal Serial Bus
VIF	Voter Information File
VVSG	Voluntary Voter System Guidelines

©2014 atsec information security corporation

References

Documentation provided for the source code review included Dominion Democracy Suite product documentation, as well as other publically available standards documents. The atsec source team also consulted other publically available documents.

Dominion Democracy Suite General Documentation

[DEMC] Dem Suite EMS Coding Standard C#.docx Revision 0.0.98.

Dominion Democracy Suite EMS Documentation

[EMSFUN] 2.03 - EMS Functionality Description Version: 4.14.A-1::253 September 19, 2014

Dominion Democracy Suite Adjudication Documentation

[ADJSDS] 2.05 - Adjudication Software Design and Specification, Version: 4.14.D::25, Dominion Voting Systems, September 18, 2014.

[ASMM] 2.09 – Adjudication System Maintenance Manual, version 4.14.D::9, September 18, 2014.

[DSIAP] Democracy Suite ImageCast Adjudication Installation and Configuration Procedures, Version: 4.14.C::48, July 17, 2014

Public Documents

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

[FCAM] Xie, T., Liy, F., Feng, D.,: Fast Collision Attack on MD5. Cryptology ePrint Archive, Report 2013/170 (2013), https://eprint.iacr.org/2013/170.pdf.

[FIPS140-2C] "Annex C: Approved Random Number Generators for FIPS PUB 140-2, Security Requirements for Cryptographic Modules," Information Technology Laboratory, National Institute of Technology, February, 2012.

[NFR] SANS Institute InfoSec Reading Room, .NET Framework Rootkits: Backdoors inside your Framework, November, 2008, http://www.sans.org/reading-room/whitepapers/windowsnet/dotnet-framework-rootkits-backdoors-framework-32954.

[SP800131A] NIST Special Publication 800-131A, "Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths," U. S. Department of Commerce and National Institute of Standards and Technology, January, 2011.

[TRUST] Thompson, Ken, "Reflections on Trusting Trust," Communications of the ACM, Volume 27, Number 8, August, 1984, http://dl.acm.org/citation.cfm?id=358210.

[VVSG1] United States Election Assistance Commission, 2005 Voluntary Voting System Guidelines, Volume 1, Version 1.0, 2005.

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