Documentation for Additional Review of Proposed Design Flaw on Dominion ImageCast Evolution (ICE) Voting System

Added note:
This 76-page package was prepared by the operations staff of the New York State Board of Elections in preparation for the Board meeting of April 29, 2019.

April 16, 2019

As requested at the March 19, 2019 meeting, the Election Operations Unit engaged NYSTEC, SLI Compliance and Dominion Voting Systems to undertake an additional review of the ImageCast Evolution (ICE) system, the security review performed by SLI as part of the original certification, threat register documentation as provided in the Technical Data Package (TDP) for the system and any other documentation deemed relevant to the "design flaw" as identified originally by Professor Andrew Appel and subsequently raised by Commissioner Kellner in his March 7th memo.

After reviewing the initial security review performed by SLI as part of the original certification process *(see SLI Source Code Review Version 1.0)*, NYSTEC requested additional information regarding SLI's approach and additional artifacts from their testing process. SLI responded with a second version *(see SLI Source Code Review Version 2.0),* which prompted additional requests from NYSTEC, resulting in a third version *(see SLI Source Code Review Version 3.0)*. After reviewing all of the information provided by SLI, a conference call was held with all parties to discuss if any additional review should be conducted. It was decided that SLI would move forward with an additional review of Dominion's code related specifically to the calling of certain print-based functions. The results of that review can be found in the *SLI Source Code Review Addendum*.

NYSTEC then prepared a second response to the State Board *(see NYSTEC Response #2)*, in which they summarized the process, their findings and potential additional mitigations to be considered. In their response, it is stated that:

> *NYSTEC believes that SLI security testing of the Dominion source code provided reasonable assurance that malicious code that could be triggered to enable the machine to print additional marks on an already marked ballot, is not present in the version tested.*

Regarding the potential implementation of any of the detective or preventative controls mentioned in their response, additional information should be considered. There was discussion before the last board meeting about the ICE system having a hardware-based counter which logged each time the printer was engaged. Although NYSTEC has proposed using a Commercial off-the-shelf (COTS) device for extracting information from the counter, accessing the counter hardware would require removing the main cover of the system, potentially voiding the warranty. An alternative proposal for accessing the information on the counter would be for Dominion to submit a modification to their software which would allow a report to be generated from the ICE unit that would show the information contained on the hardware-based counter. To ensure accuracy and integrity of the information extracted from this hardware component through the use of software, a County Board could verify the software installed on a given machine or re-install the trusted build before generating the counter report.

As for the preventative controls discussed, the most easily implementable among them would be to leave the printer access panel open. Any attempt to route a ballot through the printer path would result in the ballot coming out of the rear of the machine, preventing it from being scanned for tabulation and providing an obvious indication to the poll worker that there was some malfunction of the unit. If a poll worker needed to initiate an Accessible Voting Session (AVS) for a voter, they would simply need to close the panel and fasten the two hand screws which secure it in place. If a poll worker were to forget to close the panel when initiating an AVS, the machine would provide a warning that poll worker intervention was necessary to properly complete the AVS. Standard voting would be unaffected by the panel door remaining open.

Removing the printer cartridge by default and requiring its installation for an AVS to be properly executed would indeed negate the ability of a machine to place a mark on a ballot. However, the process of installing a cartridge would require the possession and use of a Technician Key, credentials to access the Technician menu and 3 to 5 minutes to switch between the poll

worker and technician profiles in order to initiate a "Change Printer Cartridge" function and take the appropriate action with the print cartridge. Any attempt to initiate an AVS when a printer cartridge is not installed would result in an error to the poll worker and an AVS would not be allowed to continue until a cartridge was installed. While standard voting would be unaffected by the absence of a print cartridge, most poll sites are not currently equipped with access to a Technician Key and the credentials necessary to take the necessary actions to install/remove the cartridge.

The last preventative control discussed in NYSTEC's second response would be to insert a foam block, or other obstruction, in order to physically prevent the printer making any marks on a ballot. The obstruction would need to be removed before an AVS could be initiated. As this physically control is not part of the voting system, there is no way for the machine to warn a poll worker of its presence if they were to initiate an AVS and forget to remove the obstruction. This could cause a potential malfunction or damage to the unit.

Lastly, there was discussion, both at the last board meeting as well as in NYSTEC's second response, as to having Dominion look at the Threat Register provided in the certified system's TDP for the possible addition of any applicable threats and mitigation strategies not already enumerated. Dominion did submit a revised version of its *Democracy Suite System Security Specification* document this week *(see Revised Threat Register from Dominion)*, in which they added a section for "Tampering with installed software" that seeks to address the threat of a bad actor's ability to gain access to the system to install fraudulent software.

The following contains various documentation discussed above and/or otherwise relevant to the review that was undertaken.

Table of Contents

SLI Original Response

SLI COMPLIANCE℠
4720 INDEPENDENCE ST • WHEAT RIDGE, COLORADO 80033 • 844-754-8683 • 303-422-1566
slicompliance.com SLI Compliance, a Division of GLI®

March 14, 2019

Brendan Lovullo
Deputy Director of Election Operations
New York State Board of Elections
40 North Pearl Street
Albany, New York 12207

Re: ICE Machine Design Flaw

Dear Mr. Lovullo,

SLI Compliance is responding to an inquiry from the New York State Board of Elections (NYSBOE) in regard to a blog written on October 16, 2018 by the Center for Information Technology Policy at Princeton University, about a perceived design flaw in the Dominion ImageCast Evolution voting machine. The questions asked by NYSBOE are as follows:

- Is/Was SLI aware of this issue?
- Was its threat potential checked in the SLI review?
- Is it a credible threat?
- Do you have any documents that would have addressed such?

After reading the blog and conducting internal research, SLI Compliance feels that the term design flaw is subjective to the author of the blog. The functionality in question has been confirmed via vendor documentation to be a part of the system programing and is valid functionality for the device (Please note that the documentation researched by SLI Compliance may not be Dominion's most up to date documentation). While the ability to tabulate and mark ballots on the same device might be perceived as a design flaw, that in and of itself doesn't constitute a full-blown security vulnerability or security flaw. The Ballot Marking and Remarking capability is part of the Accessible Voting functionality of the system and as such can be turned on and off. At no point was the machine observed making unauthorized or un-audited changes or additions to the ballots prior to or after being cast; this was confirmed via regular security

testing. While SLI Compliance did not specifically test this exact scenario, we did test the ability to modify or change the voting software/firmware of the device, as well as attempting to modify the results during and after the process of ballots being cast. Modifying the device software/firmware for a nefarious purpose was prevented, and all results of cast ballots were verified.

These are a few of the protections that were observed either through testing or documentation reviewed:

1. The AVS functionality of the voting system can be turned on and off.
2. The ballot marking functionality requires poll worker intervention, including utilization of the security token to perform the functions.
3. All ballot marking functionality is reviewed either by audio or visual confirmation prior to casting the ballot by the voter.
4. All ballot remarking functionality can be turned on and off.
5. All ballot remarking capability is decided by the voter if he/she wishes to auto-mark previous selections or revote the ballot. Auto remarked ballots are subject to the same audio/visual confirmation as other ballots. It should be noted ballot remarking is only instituted if there is an error condition that prevents the ballot from being cast.

There are also security protocols, processes and procedures in place to limit the amount of time that someone has access to the device. So, if the jurisdictions are doing their due diligence then this particular scenario is something that would qualify as a limited risk. The main focus of this article seems to be that hybrid machines are a bad idea, and that if they are ever compromised, a device could be used to affect votes completed on the ICE if it has the BMD features turned on or configured, either maliciously or by design.

A few answers to the questions at hand:

Q: Is it impossible to hack the ICE to do what the author states?

A: Given enough time and unfettered access to voting equipment, anything is possible.

A2: If the jurisdiction is following the vendor recommended security best practices, then any changes to the software should be noted immediately, not to mention there are built in protections (physical, electronic, procedural) that should catch the issues. Unfettered access to the devices in most cases is not a feasible reality.

Q: Is this a potentially credible concern/vulnerability?

A: While every potential threat needs to be taken seriously, we do not believe that this scenario represents a credible concern/vulnerability.

A2: There are processes and procedures in place to limit the amount of time that the machines are interacted with by the general public. Also, there are physical security barriers in place to prevent access to ports, so the ability to re-write the functionality of the device by inserting a USB device is greatly exaggerated. Therefore, in the context of this blog, SLI feels this is not a credible concern/vulnerability. While the author's perceived design flaw may be a vulnerability, there are mitigations in place to prevent, detect and deter these types of attacks.

In addition, there is a separate printer to print the ballots, so SLI Compliance feels that the article is not based on a realistic election scenario. There is no way to automatically mark and vote a ballot. See below from the Dominion ImageCast Evolution documentation.

2.3 Vote Selection and Casting of the Ballot

Audio-Visual voting is provided by allowing a voter to listen and/or view the vote options and select their choices using a combination of an Audio Tactile Interface (ATI) and the touch-screen monitor. Once a voter has finished making selections, the ballot printer produces a ballot that is inserted into the tabulator and handled like all other paper ballots.

SLI Compliance does not see this as an issue if the system is being used as intended.

If there are further questions for SLI Compliance, feel free to let me know.

Sincerely,

Traci Mapps
Director, SLI Compliance

NYSTEC Response #1

Reported Design Flaw in Dominion ImageCast Evolution (ICE) Voting Machine
NYSTEC Response

Background on Reported Design Flaw

In a memorandum to the NYS Board of Elections on March 7th, 2019 Commissioner Douglas A. Kellner identified a recently discovered report by two professors of computer science, that suggests that the Dominion ImageCast Evolution voting machine has a "design flaw"[1] that could enable the ICE voting machine to print more votes on the ballot after it was marked and reviewed by the voter.[2]

At the request of the NYS Board of Elections, NYSTEC:

- Read the relevant reports from the Freedom to Tinker site
- Reviewed the Dominion ICE TDP to understand the paper flow path and threats considered by the vendor
- Reviewed our August 29, 2018 report on the NYSBOE testing of the Dominion ImageCast Evolution (ICE) 4.14.25 upgrade
- Reviewed the response provided to NYSBOE by SLI
- Participated in a call with Dominion on 3/14/19 where all agreed an attack is possible when a machine is running compromised firmware.

Based on our independent research and above efforts, NYSTEC agrees with the basic premise of the Freedom to Tinker report that, if an attacker can find a way to defeat the internal technical controls and the County Board of Election's procedural controls that protect the firmware, this attack is theoretically possible and could result in additional votes being added to a ballot after a voter has inspected that ballot.

Background on NYSTEC role in the NYSBOE certification of the ICE Voting Machine

At the request of the New York State Board of Elections (SBOE), the New York State Technology Enterprise Corporation (NYSTEC) provided a review of the NYSBOE testing of the Dominion ImageCast Evolution (ICE) 4.14.25 upgrade. NYSTEC was asked to do a review of the Dominion ICE upgrade documentation and a review of the NYSBOE testing effort which included a review of test scripts, observations of a sample selection of test script executions and a review of test script results and artifacts. NYSTEC's scope did not include the review of any security testing conducted by SLI or during the Federal certification process.

[1] https://www.cs.princeton.edu/~appel/

2 https://freedom-to-tinker.com/2018/10/16/design-flaw-in-dominion-imagecastevolution-voting-machine/

NYSTEC provided a final report to SBOE on August 29, 2018. The NYSTEC report did identify all requirements whose results were accepted from the federal certification and these included NYS Election Law as well as 6209 and 6210 regulations tested by SLI, however review of that testing was out of scope. NYSTEC is not aware if the SLI testing included a source code review that focused on whether any malicious code, capable of this type of attack was present in the firmware.

NYSTEC Analysis of the Issue

NYSTEC, NYS State Board of Elections and computer science experts have long agreed that when an adversary has the ability to modify or replace the software/firmware that controls a voting machine then significant and damaging impacts to an election are possible. What makes this type of attack different however is that the voted paper ballots from a compromised combination BMD/scanner machine could not be easily used to audit the scanner results because they have been compromised. If the software/firmware was compromised to alter election results, on a regular scanner (without BMD capabilities) one still has the voted ballots to ensure the election can be properly decided. This would not be the case with the BMD/scanner attack and if such an attack were to occur, then a forensic analysis would be needed on all ballots in question to determine if a human or machine made the mark. Such a process is unlikely to be trusted by the public.

Threat Mitigation

It is NYSTEC's understanding that currently no Dominion ICE machines are in use. However, when they are deployed, we would expect that all of the controls that are in place for precinct scanners would be employed to mitigate this risk for the Dominion ICE machine. These would include but not be limited to:

- Cryptographic hash checking of the machines' code to detect the presence of unauthorized software/firmware.
- Physical controls during storage and voting which restricts unauthorized access to the system.
- Chain of custody procedures for the voting machine whenever it is not in a secure CBOE location.
- Tamper resistant enclosures and seals that protect against and detect unauthorized access to the machine when it is in public place.
- Use of credentialed access, including 2FA that restricts access to the voting machine

Other factors that make this threat scenario a low risk including level of sophistication needed to create an effective exploit that would be undetectable and successful in altering an election

Overall Risk

While NYSTEC has not conducted a full NIST 800-30 risk assessment of this particular threat we suspect that the overall risk rating would be low because the existing controls would be in place and because of the complexity of creating such firmware. However, if the attack were to occur, the impact would be very high. As per the NIST standard we use below this would result in an overall risk rating of low.

TABLE I-2: ASSESSMENT SCALE – LEVEL OF RISK (COMBINATION OF LIKELIHOOD AND IMPACT)

Likelihood (Threat Event Occurs and Results in Adverse Impact)	Level of Impact				
	Very Low	Low	Moderate	High	Very High
Very High	Very Low	Low	Moderate	High	Very High
High	Very Low	Low	Moderate	High	Very High
Moderate	Very Low	Low	Moderate	Moderate	High
Low	Very Low	Low	Low	Low	Moderate
Very Low	Very Low	Very Low	Very Low	Low	Low

Recommendations

To further mitigate the risks, we have described NYSTEC recommends consideration of the following risk mitigation steps.

- Conduct a full security source code review to provide assurances there is no malicious code within the Dominion ICE (assuming one hasn't already been completed). Note, there is no reason to suspect this is the case and SLI and SBOE testing saw no behaviors that would indicate an issue.
- Require SLI and Dominion to include the threat of modification of voting machine software/firmware in their threat scenarios and list all of the technical safeguards and recommended procedures preventing and detecting the threat, as per 6209.6 After this is completed, review the safeguards for effectiveness (perhaps with additional security testing to ensure the safeguards work as documented). Ensure all procedures written for the ICE contain the steps as recommended by Dominion.
- Ask Dominion to consider the use of additional controls for the ICE such as:
    - Modification whereby a poll worker can physically disable the use of the printer when the ICE machine is not in BMD mode
    - Modification whereby a poll worker can physically disable the ability for a ballot to pass over the printer when the ICE machine is not in BMD mode
    - Use of different ink that would display differently to aid a forensic examination or manual audit
    - Any modification that is not implemented in software/firmware to ensure that a voted paper ballot cannot ever pass by the printer

- Consider additional requirements for the ICE and other combination BMD/scanner machines such as:
    - Increased frequency of software/firmware hash code validations
    - Audit CBOE's who use the ICE to ensure all existing and new (for early voting) security procedures are in place
    - Add an additional step to the Post Election Audit (6210.18) whereby the persons doing the audit review ballots to verify that all marks appear to come from a pen or printer, but not both.

SLI Source Code Review
Version 1.0

SLI Compliance
4720 Independence St
Wheat Ridge, CO 80033

303.215.5853
www.slicompliance.com

NYSBOE Dominion Source Code Review Findings
ImageCast Evolution Only

1.0 Introduction

In previous certification efforts for NYSBOE, SLI verified all source code modules provided to the state were compliant with the best practices and coding requirements defined in the following standards:

- VVSG 2005, (Volume I and Volume II)
- 2014NYElectionLaw.pdf
- 6210Regulations09052008.pdf
- Ciber_COTSStandard.pdf
- NYS_voting_systems_standards-4-20(6209).pdf
- havalaw.pdf
- Table of known vulnerabilities
- Democracy Suite EMS Coding Standards.docx

This included a secure source code review to ensure protection against all known and identified vulnerabilities identified within prior ITA reports, voting system tests, or risk assessment final reports, and other comparable examinations performed by independent testing organizations.

All Source code submitted for certification was compiled and reviewed at the SLI Compliance Testing Facility by SLI staff. The build scripts, vendor source code, and any modified Commercial Off-the-Shelf (COTS) code were reviewed for format, structure, and functionality. All code delivered was also reviewed using both automated and manual methods for malicious code, Trojans, and viruses.

Since the initial certification effort, Dominion upgraded the Imagecast Evolution (ICE) which underwent source code review and revealed no discrepancies. Dominion recently submitted an upgrade to Imagecast Evolution (ICE). SLI performed a thorough examination of the ICE code including a review for conformance against the VVSG 2005 requirements as well as the best practices and coding requirements outlined.

2.0 Scope of Review and Methodology

The objective of the source code review was to analyze the ICE software, and to confirm that the code complies with the criteria set forth by the state of New York.

The ICE source code was subjected to a complete review against the criteria of the VVSG by NTS, Dominion's VSTL, prior to delivery to SLI. Upon receiving the source code SLI performed a preliminary review of approximately 10% of the changed source code lines in order to verify compliance with Federal requirements. Following the preliminary review of the 10% sample, SLI performed a review of the entirety of the source code to ascertain compliance with the criteria set forth by the state of New York.

List of Methodology Criteria

Review all voting system software and firmware for security access controls, and determine any vulnerabilities.

Determine the effectiveness of the security access controls for the voting system's software.

Review the voting system's software source code to determine the presence of vulnerabilities, or if the system can be executed outside the intended manner and outside of normal conditions.

Where non-catastrophic failure of a device may be related to programming, exception handling routines were examined for data handling logic employed.

Review voting system software source code to verify data validation routines.

Verify Telecommunications capabilities, which are not allowed in NY elections systems, are not employed.

Verify there are no exploitable vulnerabilities that could affect warnings, alerts, error messages or logging.

Verify the inclusion of logic related to real time audit logging as stated in the voting standards.

Any other direct violations of the aforementioned standards were noted in discrepancy reports. Discrepancy reports were provided to the system vendor for their review and remediation.

3.0 Overview of Findings

SLI's review of the 10% sample of the changed source code lines confirmed the compliance of the source code with the Federal standard. SLI's review of the entirety of

the source code against the criteria set forth by the state of New York also confirmed compliance.

4.0 Conclusion

Based on the reviews conducted against the submitted DVS ICE source code, SLI recommends the following source code versions for certification:

ICE 4.14.25

SLI Compliance
4720 Independence St
Wheat Ridge, CO 80033

303.215.5853
www.slicompliance.com

NYSBOE Dominion Security, Accessibility and TDP Review
ImageCast Evolution Only

1.0 Introduction

SLI Compliance conducted a technical review of the Dominion ImageCast Evolution (ICE) device, version 4.14.25E firmware to verify the product meets the proper security, accessibility and documentation requirements for qualification in the New York State Board of Elections (NYSBOE) Certification of Election systems. The goal of the review was to verify that the ICE device met all security and accessibility requirements specified in the VVSG 2005 requirements and the 2017 NY State Election Laws as required by the NYSBOE. In addition, the ICE documentation was reviewed to verify it meets all VVSG and NY state law requirements as part of the Technical Data Package submitted with the system.

2.0 Scope of Review and Testing

The technical review included an analysis of all published test plans and reports provided on the US Elections Assistance Commission (EAC) public website including 3rd party usability test reports provided by the manufacturer, to verify what level of testing was conducted on the Dominion ICE device and firmware. The review looked at both security and accessibility testing and results that were reported by the test labs that conducted the certification testing. The results were compiled in an effort to identify any gaps in the test results where the VVSG requirements and NY Election Laws were not sufficiently met.

All gaps in accessibility testing were identified and the analysis has been provided in section 3.0 of this document below. Where gaps in security were identified, SLI Compliance conducted independent security evaluation on the ICE device and firmware to verify the product sufficiently meets the VVSG and NY requirements that were identified as gaps.

The review of the ICE TDP documentation looked at Software Design Specifications, System Functionality Description, System Hardware Specifications, System Maintenance Procedures, and System Operations Procedures to verify the documentation contained the required information per the VVSG and NY requirements.

2.1 Security Testing

Security Testing was performed on the ICE Voting system device to verify all security focused VVSG and NY Election Law requirements were satisfied. The following security testing was executed against the Dominion ICE voting system device.

Access Control

Tests were performed on each user role within the ICE voting system device in order to verify that a user is allowed to perform all necessary tasks for their role, but are not allowed to perform any tasks not assigned to their role, nor are they able to modify a role to a higher level role activity.

Tests were performed to verify that the ICE voting system equipment prevents modification of related software/firmware by any means other than the documented procedure for software upgrades. Tests were performed to verify that tampering is not allowed.

Tests were performed on each user role within the voting system in order to verify that a user is allowed to perform all necessary tasks for their role and are appropriately identified by the system.

Tests were performed to verify that the administrator group or role is able to perform all the functions listed in the requirements above. Tests were performed that verify that users are authenticated properly prior to being allowed access, and that all private access data is secured properly.

Tests were performed on each user role within the voting system in order to verify that only authorized users, roles, or groups are allowed access to election data, as based on pertinent access control lists or policies.

Tests were performed to verify the documented procedures as well as attempts to defeat the implemented access control security on each system component.

Physical Security

Tests were performed to verify that unauthorized physical access leaves physical evidence of the intrusion. Tests were performed to verify that only ports and access points essential to voting operations, testing, and auditing are present. Tests were performed to verify that event log entries adequately identify affected devices. Tests were performed to verify ports disabled during the open polls period can only be re-enabled by an authorized

administrator. Tests were performed to verify that all access points and ballot boxes are secured and provide adequate tamper evidence as well as tamper resistant countermeasures.

Polling place security

Tests were performed to verify that the system documented measures provide adequate polling place security.

Software and Firmware Installation

Tests were performed to verify that if any software or firmware is installed, unless the documentation details how to protect it, it is inaccessible to activation or control by anything other than authorized means. Tests were performed to verify that no source code or compilers or assemblers are resident or accessible, after Election Day testing.

Protection against Malicious Software

Tests were performed to verify that COTS products are implemented to protect against malicious software, as described in voting system manufacturer documentation.

The ICE system is a proprietary system that utilizes firmware and compact flash cards to run, load, and store election based software. This system contains no AV protection.

Software Setup Validation

Tests were performed to verify that the installation process for each system component is robust and maintains the integrity of the voting system.

SLI was able to update the ICE without having to uninstall the older version. SLI wasn't able to install an older version on top of a newer version.

Tests were performed to verify appropriate encryption, receipt validation, and data integrity.

It was verified that election results processed on the ICE device and modified were not able to be imported into the EMS for official vote tally. The ICE also resisted modification and reinsertion of results media back into the ICE device.

Analysis of the results media indicated that there were two sets of results the RAW ballot images which were not encrypted but digitally signed to prevent modification. It was confirmed that the tabulated results files, were encrypted by performing entropy analysis on each of the files located on the results media. In addition, manual analysis using a hex editor to check for the presence of plaintext within the encrypted files was conducted.

3.0 Overview of Findings

3.1 Accessibility

A review of the Dominion Democracy Suite 4.14 Test Report Rev C Final with Certification Number.pdf document from the EAC Website shows that the system was tested for the following Usability & Accessibility requirements:

- VVSG Vol 1 3.2.4.a
- VVSG Vol 1 3.2.4.b

From section 4.6.3 of the test report the following information was cited:

"The requirements identified for this campaign were EAC 2005 VVSG Vol. I, Section 3.2.4a and b. The newly introduced ICE Plastic Ballot box was tested to ensure the applicable mobility requirements were met. "

The two requirements (3.2.4) are Mobility requirements and were shown to have tested and indicated as passed.

Alternative Languages (3.1.3), Perceptual Issues (3.1.5), Interaction Issues (3.1.6), Privacy (3.1.7), General (3.2.1), Partial Vision (3.2.2), Blindness (3.2.2.2), Dexterity (3.2.3), Hearing (3.2.5), Speech (3.2.6), English Proficiency (3.2.7) and Cognitive Issues (3.1.4, 3.2.8) requirements were not included in the Dominion Democracy Suite 4.14 Test Report Rev C Final Report.

SLI reviewed the ICE Usability Study.pdf v1.0.0: 36 July 13, 2012 Usability Study of Dominion Voting Systems and ImageCast Evolution Version 4.1.1.1 and 4.6.1.1.
The study focused on Effectiveness, Efficiency, User Satisfaction and Usability. The participants represented those with partial vision, blindness, mobility and hearing issues. The results of the study were favorable and would satisfy the VVSG requirements for certification referenced above. SLI recommends accepting the usability testing study performed on the ImageCast Evolution device.

3.2 Security

The Dominion ICE device with firmware version 4.14.25 was verified and found to be compliant with the VVSG standards and NY Election Law security requirements. All requirements were verified through testing and/or documentation. Accompanying this report is a detailed list of all security requirements verified. This can be found in the *NYSBOE DVS (ICE) Security Requirements Verified xlxs* document.

See accompanying documentation.

3.3 TDP Documentation

Software Design & Specification - No issues were found and the documentation meets VVSG and NY requirements.

System Functionality Description – One potential issue found: NY State Regulation 6209.2 (7) – *The system shall incorporate multiple memories, including resident vote tabulation, storage of results and ballot images in resident memory, serving as a redundant means of verifying or auditing election results or ballot images, and further, the system shall be required to alert the election day worker that memory capacity is about to be reached.*

Issue: SLI was unable to locate documentation describing how a system alert is triggered when memory capacity is about to be reached.

System Hardware Specification - No issues were found and the documentation meets VVSG and NY requirements.

System Maintenance Procedures - 1 issue found**:** NY State Regulation 6209.6 F (7) (viii) - *The vendor shall provide complete instructions for all methods of voting which voters may use to cast their vote, including instructions on entering and changing votes, write-in voting, verifying votes and accepting the cast votes. Written and audio instruction shall be provided in each language in which voting shall occur within the state.*

Issue: SLI was unable to locate specific instructions for voting on the ICE Device.

System Operations Procedures - No issues were found and the documentation meets VVSG and NY requirements.

4.0 Conclusion

Based on the source code review, TDP & Accessibility documentation review, ECO documentation analysis and Security testing conducted against the submitted Dominion ICE device & firmware, SLI recommends the following for acceptance and inclusion in the NYSBOE certification program:

ICE 4.14.25

Source	Requirement	Area	Covered
VVSG Vol 1: 2.1.1a	a. Provide security access controls that limit or detect access to critical system components to guard against loss of system integrity, availability, confidentiality, and accountability	General	Yes
VVSG Vol 1: 2.1.1d	d. Provide safeguards in response to system failure to protect against tampering during system repair or interventions in system operations	Physical Access Controls	Yes
VVSG Vol 1: 2.1.1e	e. Provide security provisions that are compatible with the procedures and administrative tasks involved in equipment preparation, testing, and operation	Procedural Controls	Yes
VVSG Vol 1: 2.1.1f	f. Incorporate a means of implementing a capability if access to a system function is to be restricted or controlled	Logical Access Controls	Yes
VVSG Vol 1: 2.1.1g	g. Provide documentation of mandatory administrative procedures for effective system security	PCA Reference Review	Yes
VVSG Vol 1: 2.1.4a	a. Protect against a single point of failure that would prevent further voting at the polling place	Availability	Yes
VVSG Vol 1: 2.1.4b	b. Protect against the interruption of electrical power	Availability	Yes
VVSG Vol 1: 2.1.4e	e. Protect against the failure of any data input or storage device	Integrity	Yes
VVSG Vol 1: 2.1.4f	f. Protect against any attempt at improper data entry or retrieval	Integrity	Yes
VVSG Vol 1: 2.1.4g	g. Record and report the date and time of normal and abnormal events	Accountability	Yes
VVSG Vol 1: 2.1.4h	h. Maintain a permanent record of all original audit data that cannot be modified or overridden but may be augmented by designated authorized officials in order to adjust for errors or omissions (e.g., during the canvassing process)	Accountability	Yes
VVSG Vol 1: 2.1.4i	i. Detect and record every event, including the occurrence of an error condition that the system cannot overcome, and time-dependent or programmed events that occur without the intervention of the voter or a polling place operator	Accountability	Yes
VVSG Vol 1: 2.1.5.1av	v. The generation of audit record entries shall not be terminated or altered by program control, or by the intervention of any person. The physical security and integrity of the record shall be maintained at all times.	Accountability	Yes
VVSG Vol 1: 2.1.5.1avi	vi. Once the system has been activated for any function, the system shall preserve the contents of the audit record during any interruption of power to the system until processing and data reporting have been completed.	Accountability	Yes
VVSG Vol 1: 2.1.5.2a	a. First, authentication shall be configured on the local terminal (display screen and keyboard) and on all external connection devices (“network cards” and “ports”). This ensures that only authorized and identified users affect the system while election software is running.	Logical Access Controls	Yes
VVSG Vol 1: 2.1.5.2b	b. Second, operating system audit shall be enabled for all session openings and closings, for all connection openings and closings, for all process executions and terminations, and for the alteration or deletion of any memory or file object. This ensures the accuracy and completeness of election data stored on the system. It also ensures the existence of an audit record of any person or process altering or deleting system data or election data.	Accountability	Yes
VVSG Vol 1: 2.1.5.2c	c. Third, the system shall be configured to execute only intended and necessary processes during the execution of election software. The system shall also be configured to halt election software processes upon the termination of any critical system process (such as system audit) during the execution of election software.	OS Hardening	Yes
VVSG Vol 1: 2.1.7.1b	b. Accommodate device control functions performed by polling place officials and maintenance personnel	Logical Access Controls	Yes
VVSG Vol 1: 2.1.8d	d. Prevents or disables the resetting of the counter by any person other than authorized persons at authorized points	Logical Access Controls	Yes
VVSG Vol 1: 2.2.1.2f	f. Prevention of unauthorized modification of any ballot formats	Logical Access Controls	Yes
VVSG Vol 1: 2.2.1.2g	g. Modification by authorized persons of a previously defined ballot format for use in a subsequent election	Logical Access Controls	Yes
VVSG Vol 1: 2.2.5i	ii. Confirmation that the device is ready to be activated for accepting votes	Availability	Yes

Source	Requirement	Area	Covered
VVSG Vol 1: 2.2.5j	i. If a precinct count system includes equipment for the consolidation of polling place data at one or more central counting locations, it shall have means to verify the correct extraction of voting data from transportable memory devices, or to verify the transmission of secure data over secure communication links.	Integrity	Yes
VVSG Vol 1: 2.3.1.2a	a. A means of verifying that ballot marking devices are properly prepared and ready to use	Availability	
VVSG Vol 1: 2.3.1.3a	a. A security seal, a password, or a data code recognition capability to prevent the inadvertent or unauthorized actuation of the poll-opening function	Logical Access Controls	Yes
VVSG Vol 1: 2.3.3.1e	e. In the event of a failure of the main power supply external to the voting system, provide the capability for any voter who is voting at the time to complete casting a ballot, allow for the successful shutdown of the voting system without loss or degradation of the voting and audit data, and allow voters to resume voting once the voting system has reverted to back-up power	Availability	Yes
VVSG Vol 1: 2.4.1a	a. Preventing the further casting of ballots once the polls have closed	Logical Access Controls	Yes
VVSG Vol 1: 2.4.3g	g. Prevent data from being altered or destroyed by report generation, or by the transmission of results over telecommunications lines	Integrity	Yes
VVSG Vol 1: 2.4.3h	h. Prevent the printing of reports and the unauthorized extraction of data prior to the official close of the polls	Logical Access Controls	Yes
VVSG Vol 1: 3.1.7.1a	a. The ballot and any input controls shall be visible only to the voter during the voting session and ballot submission.	Confidentiality	Yes
VVSG Vol 1: 3.1.7.1b	b. The audio interface shall be audible only to the voter.	Confidentiality	Yes
VVSG Vol 1: 4.1.4.2dii	ii. Incorporate locks or seals, the specifications of which are described in the system documentation	Physical Access Controls	Yes
VVSG Vol 1: 4.1.4.2diii	iii. Provide specific points where ballots are inserted, with all other points on the box constructed in a manner that prevents ballot insertion	Physical Access Controls	Yes
VVSG Vol 1: 7.2.1	7.2.1 General Access Control Policy The vendor shall specify the general features and capabilities of the access control policy recommended to provide effective voting system security.	Logical Access Controls	Yes
VVSG Vol 1: 7.2.1.1a	a. Identify each person to whom access is granted, and the specific functions and data to which each person holds authorized access	Logical Access Controls	Yes
VVSG Vol 1: 7.2.1.1b	b. Specify whether an individual's authorization is limited to a specific time, time interval or phase of the voting or counting operations	Logical Access Controls	Yes
VVSG Vol 1: 7.2.1.1c	c. Permit the voter to cast a ballot expeditiously, but preclude voter access to all aspects of the vote counting processes	Logical Access Controls	Yes
VVSG Vol 1: 7.2.1.2a	Vendors shall provide a detailed description of all system access control measures designed to permit authorized access to the system and prevent unauthorized access. Examples of such measures include: a. Use of data and user authorization b. Program unit ownership and other regional boundaries c. One-end or two-end port protection devices d. Security kernels e. Computer-generated password keys f. Special protocols g. Message encryption h. Controlled access security	Logical Access Controls	Yes
VVSG Vol 1: 7.2.1.2b	Vendors also shall define and provide a detailed description of the methods used to prevent unauthorized access to the access control capabilities of the system itself.	Logical Access Controls	Yes
VVSG Vol 1: 7.2.1a	a. Software access controls	Logical Access Controls	Yes
VVSG Vol 1: 7.2.1b	b. Hardware access controls	Logical Access Controls	Yes
VVSG Vol 1: 7.2.1c	c. Communications	Logical Access Controls	Yes
VVSG Vol 1: 7.2.1d	d. Effective password management	Logical Access Controls	Yes
VVSG Vol 1: 7.2.1e	e. Protection abilities of a particular operating system	Logical Access Controls	Yes
VVSG Vol 1: 7.2.1f	f. General characteristics of supervisory access privileges	Logical Access Controls	Yes
VVSG Vol 1: 7.2.1g	g. Segregation of duties	Logical Access Controls	Yes

Source	Requirement	Area	Covered
VVSG Vol 1: 7.2.1h	h. Any additional relevant characteristics	Logical Access Controls	Yes
VVSG Vol 1: 7.3	7.3 Physical Security Measures A voting system's sensitivity to disruption or corruption of data depends, in part, on the physical location of equipment and data media, and on the establishment of secure telecommunications among various locations. Most often, the disruption of voting and vote counting results from a physical violation of one or more areas of the system thought to be protected. Therefore, security procedures shall address physical threats and the corresponding means to defeat them.	Physical Access Controls	Yes
VVSG Vol 1: 7.3.1i	For polling place operations, vendors shall develop and provide detailed documentation of measures to enable poll workers to physically protect and perform orderly shutdown of voting equipment to counteract vandalism, civil disobedience, and similar occurrences.	PCA Reference Review	Yes
VVSG Vol 1: 7.3.1ii	The measures shall allow the immediate detection of tampering with vote casting devices and precinct ballot counters. They also shall control physical access to a telecommunications link if such a link is used	Physical Access Controls	Yes
VVSG Vol 1: 7.3.2	Vendors shall develop and document in detail the measures to be taken in a central counting environment. These measures shall include physical and procedural controls related to the handling of ballot boxes, preparing of ballots for counting, counting operations and reporting data.	Procedural Controls	Yes
VVSG Vol 1: 7.4.1a	a. If software is resident in the system as firmware, the vendor shall require and state in the system documentation that every device is to be retested to validate each ROM prior to the start of elections operations.	Procedural Controls	Yes
VVSG Vol 1: 7.4.1b	b. To prevent alteration of executable code, no software shall be permanently installed or resident in the voting system unless the system documentation states that the jurisdiction must provide a secure physical and procedural environment for the storage, handling, preparation, and transportation of the system hardware.	Integrity	Yes
VVSG Vol 1: 7.4.1c	c. The voting system bootstrap, monitor, and device-controller software may be resident permanently as firmware, provided that this firmware has been shown to be inaccessible to activation or control by any means other than by the authorized initiation and execution of the vote counting program, and its associated exception handlers.	OS Hardening	Yes
VVSG Vol 1: 7.4.1d	d. The election-specific programming may be installed and resident as firmware, provided that such firmware is installed on a component (such as a computer chip) other than the component on which the operating system resides.	OS Hardening	Yes
VVSG Vol 1: 7.4.1e	e. After initiation of election day testing, no source code or compilers or assemblers shall be resident or accessible.	OS Hardening	Yes
VVSG Vol 1: 7.4.2a	a. Voting systems shall deploy protection against the many forms of threats to which they may be exposed such as file and macro viruses, worms, Trojan horses, and logic bombs.	OS Hardening	Yes
VVSG Vol 1: 7.4.2b	Vendors shall develop and document the procedures to be followed to ensure that such protection is maintained in a current status.	Procedural Controls	Yes
VVSG Vol 1: 7.4.4a	a. The vendor shall document all software including voting system software, third party software (such as operating systems and drivers) to be installed on the certified voting system, and installation programs.	PCA Reference Review	Yes
VVSG Vol 1: 7.4.4ai	i. The documentation shall have a unique identifier (such as a serial number or part number) for the following set of information: documentation, software vendor name, product name, version, the certification application number of the voting system, file names and paths or other location information (such as storage addresses) of the software.	PCA Reference Review	Yes
VVSG Vol 1: 7.4.4aii	ii. The documentation shall designate all software files as static, semi-static or dynamic.	PCA Reference Review	Yes

Source	Requirement	Area	Covered
VVSG Vol 1: 7.4.4e	e. The voting system equipment shall be designed to allow the voting system administrator to verify that the software is the certified software by comparing it to reference information produced by the NSRL or other designated repository.	Procedural Controls	Yes
VVSG Vol 1: 7.4.6a	a. Setup validation methods shall verify that no unauthorized software is present on the voting equipment.	OS Hardening	Yes
VVSG Vol 1: 7.4.6b	b. The vendor shall have a process to verify that the correct software is loaded, that there is no unauthorized software, and that voting system software on voting equipment has not been modified, using the reference information from the NSRL or from a State designated repository.	Procedural Controls	Yes
VVSG Vol 1: 7.4.6bi	i. The process used to verify software should be possible to perform without using software installed on the voting system.	Procedural Controls	Yes
VVSG Vol 1: 7.4.6bii	ii. The vendor shall document the process used to verify software on voting equipment.	Procedural Controls	Yes
VVSG Vol 1: 7.4.6biii	iii. The process shall not modify the voting system software on the voting system during the verification process.	Procedural Controls	Yes
VVSG Vol 1: 7.4.6c	c. The vendor shall provide a method to comprehensively list all software files that are installed on voting systems.	Procedural Controls	Yes
VVSG Vol 1: 7.4.6d	d. The verification process should be able to be performed using COTS software and hardware available from sources other than the voting system vendor.	Procedural Controls	Yes
VVSG Vol 1: 7.4.6di	i. If the process uses hashes or digital signatures, then the verification software shall use a FIPS 140-2 level 1 or higher validated cryptographic module.	Procedural Controls	Yes
VVSG Vol 1: 7.4.6dii	ii. The verification process shall either (a) use reference information on unalterable storage media received from the repository or (b) verify the digital signature of the reference information on any other media.	Procedural Controls	Yes
VVSG Vol 1: 7.4.6e	e. Voting system equipment shall provide a means to ensure that the system software can be verified through a trusted external interface, such as a read-only external interface, or by other means.	Logical Access Controls	Yes
VVSG Vol 1: 7.4.6ei	i. The external interface shall be protected using tamper evident techniques	Physical Access Controls	Yes
VVSG Vol 1: 7.4.6eii	ii. The external interface shall have a physical indicator showing when the interface is enabled and disabled	Physical Access Controls	Yes
VVSG Vol 1: 7.4.6eiii	iii. The external interface shall be disabled during voting	Logical Access Controls	yes
VVSG Vol 1: 7.4.6eiv	iv. The external interface shall provide a direct read-only access to the location of the voting system software without the use of installed software	Logical Access Controls	Yes
VVSG Vol 1: 7.4.6fi	i. The vendor should provide a method to query the voting system to determine the values of all static and dynamic registers and variables including the values that jurisdictions are required to modify to conduct a specific election.	Logical Access Controls	Yes
VVSG Vol 1: 7.4.6fii	ii. The vendor shall document the values of all static registers and variables, and the initial starting values of all dynamic registers and variables listed for voting system software, except for the values set to conduct a specific election.	PCA Reference Review	Yes
NYS Law: 7-202.1f	(f) be provided with a “protective counter” which records the number of times the machine or system has been operated since it was built and a “public counter” which records the number of persons who have voted on the machine at each separate election	Integrity	Yes

Source	Requirement	Area	Covered
NYS Law: 7-202.1g	(g) be provided with a lock or locks, or other device or devices, the use of which, immediately after the polls are closed or the operation of the machine or system for such election is completed, will absolutely secure the voting or registering mechanism and prevent the recording of additional votes;	Physical Access Controls	Yes
NYS Law: 7-202.1rii	(ii) providing a means by which a malfunctioning voting machine or system shall secure any votes already cast on such machine or system;	Integrity	Yes
NYS Law: 7-202.1t	(t) not include any device or functionality potentially capable of externally transmitting or receiving data via the internet or via radio waves or via other wireless means	Telecom	Yes
NYS Law: 9-102.1a	(a) lock the voting machine against voting;	Logical Access Controls	Yes
6209.2.A.5	(5) Provide a battery power source in the event that the electric supply used to make the voting system equipment function, is disrupted. The battery power source shall operate the system and allow for the casting of votes for a period not less than 2 hours, to ensure that the system can shut down and preserve the integrity of votes cast prior to the power failure, and can resume functionality when power is provided or restored without significant or intrusive power-up procedures. Such batteries must be rechargeable and have minimum five-year life when used under normal conditions. In the event of a power failure, the equipment shall perform a normal shut-down not less than one hour before battery power is depleted, and shall notify the election inspector that the system will do so.	Availability	Yes
6209.2.F.10a	(a) All cryptographic software in the voting system shall have been approved by the U.S. Government's Crypto Module Validation Program (CMVP) as applicable.	Crypto	Yes
6209.2.F.12	(12) The voting system shall generate and store a digital signature for each electronic record.	Integrity	Yes
6209.2.F.13	(13) The electronic records shall be able to be exported for auditing or analysis on standards-based and/or information technology computing platforms.	Accountability	Yes
6209.2.F.13a	(a) The exported electronic records shall be in an open, non-proprietary format.	Accountability	Yes
6209.2.F.13b	(b) The voting system shall export the records accompanied by a digital signature of the collection of records, which shall be calculated on the entire set of electronic records and their associated digital signatures.	Integrity	Yes
6209.2.F.14	(14) The voting system printers shall be physically secure from tampering.	Physical Access Controls	Yes
6209.2.F.14a	(a) The voting system shall communicate with its printers over a standard, publicly documented printer port using a standard communication protocol.	General	Yes
6209.2.F.14c	(c) The printer shall not be permitted to communicate with any other system or machine other than the single voting system to which it is connected.	Logical Access Controls	Yes
6209.2.F.14d	(d) The printer shall only be able to function as a printer: it cannot store information or contain or provide any services that are not essential to system function, (e.g., provide copier or fax functions) or have network capability.	General	Yes
6209.2.F.14e	(e) Printer access to replace consumables such as ink or paper shall only be granted if it does not compromise the sealed printer paper path.	Physical Access Controls	Yes
6209.2.F.15	(15) The voting system's printers shall be highly reliable and easily maintained.	Availability	Yes
6209.2.F.15a	(a) The voting system should include a printer port to which a commercial off-the-shelf printer which complies with sub-section F(14) above, could be attached for the purposes of printing paper records and any additional records.	General	Yes
6209.2.F.15b	(b) The voting system shall detect errors and malfunctions such as paper jams or low supplies of consumables such as paper and ink that may prevent paper records from being correctly displayed, printed and stored.	Availability	Yes

Source	Requirement	Area	Covered
6209.2.F.15c	(c) If an error or malfunction occurs, the voting equipment attached to the defective printer shall suspend voting operations and shall present a clear indication to the voter and election workers of the error or malfunction.	Availability	Yes
6209.2.F.15di	(i) Printing devices should contain paper and ink of sufficient capacity so as not to require reloading or opening equipment covers or enclosures and circumvention of security features, or reloading shall be able to be accomplished with minimal disruption to voting and without circumvention of security features such as seals.	Physical Access Controls	Yes
6209.2.F.8	(8) Vendor documentation shall include procedures for returning a voting system to correct operation after a voter has used it incompletely or incorrectly; this procedure shall not cause discrepancies between the tallies of the electronic and paper records.	Procedural Controls	Yes
6209.2.G	G. Any submitted voting system's software shall not contain any code, procedures or other material which may disable, disarm or otherwise affect in any manner, the proper operation of the voting system, or which may damage the voting system, any hardware, or any computer system or other property of the State Board or county board, including but not limited to 'viruses', 'worms', 'time bombs', and 'drop dead' devices that may cause the voting system to cease functioning properly at a future time.	N/A	Yes
6209.2.H	H. Any submitted voting system shall provide methods through security seals or device locks to physically secure against attempts to interfere with correct system operations. Such physical security shall guard access to machine panels, doors, switches, slots, ports, peripheral devices, firmware, and software.	Physical Access Controls	Yes
6209.6.F.3n1	(n) Security 1. Security requirements and security provisions of the system's software shall be identified for each system function and operating mode. The voting system must be secure against attempts to interfere with correct system operation. The vendor shall identify each potential point of attack.	General	Yes
6209.6.F.3n2	2. For each potential point of attack, the vendor shall identify the technical safeguards embodied in the voting system to defend against attack, and the procedural safeguards that the vendor has recommended be followed by the election administrators to further defend against that attack. Each defense shall be classified as preventative, if it prevents the attack in the first place; detective if it allows detection of an attack; or corrective if it allows correction of the damage done by an attack.	PCA Reference Review	Yes
6209.6.F.3n3	3. Security requirements and provisions shall include the ability of the system to detect, prevent, log and recover from the broad range of security risks identified.	General	Yes
6209.6.F.3n4	4. These procedures shall also examine system capabilities and safeguards claimed by the vendor to prevent interference with correct system operations	General	Yes

SLI Source Code Review
Version 2.0

SLI Compliance
4720 Independence St
Wheat Ridge, CO 80033

303.215.5853
www.slicompliance.com

NYSBOE Dominion Source Code Review Findings
ImageCast Evolution

1.0 Introduction

SLI Compliance (SLI) certification efforts for the New York State Board of Elections (NYSBOE), consist of verifying all source code modules provided to the state for compliance with the best practices and coding requirements defined in the following standards:

- Voluntary Voting System Guidelines (VVSG) 1.0 2005 (Volume I and Volume II)
    - Democracy Suite EMS Coding Standards.docx
- 2017NYElectionLaw.pdf
    - 6210Regulations09052008.pdf
    - NYS_voting_systems_standards-4-20(6209).pdf
- Ciber_COTSStandard.pdf
- havalaw.pdf
- Table of known vulnerabilities

This includes a secure source code review to ensure protection against all known and identified vulnerabilities reported in prior ITA reports, voting system tests, or risk assessment final reports, and other comparable examinations performed by independent testing organizations.

All source code submitted for certification is compiled and reviewed at the SLI Compliance test laboratory by SLI qualified test engineers. The build scripts, vendor source code, and all modified Commercial Off-the-Shelf (COTS) code is reviewed for format, structure and functionality. Source code is also reviewed using both automated and manual methods for malicious code, Trojans, and viruses.

The Dominion Voting Systems (DVS) Imagecast Evolution (ICE) is an addition to the initial DVS configuration initially tested by SLI and certified by the NYSBOE. The ICE 4.14.25 source code was delivered to SLI from Pro V&V on January 12, 2018. SLI performed an examination of the ICE code including a review for conformance against the VVSG 1.0 2005 as well as the best practices and coding requirements outlined in the abovementioned standards.

2.0 Scope of Review and Methodology

The objective of the source code review was to analyze the ICE 4.14.25 software and to confirm that the code complies with the criteria set forth by the state of New York.

The ICE source code was subjected to a complete review, by Pro V&V for EAC certification, against the source code criteria of the VVSG 1.0 prior to delivery to SLI. SLI performed a review of approximately 10% of the source code lines in order to verify compliance with the VVSG 1.0 2005. In addition to the 10% sample, SLI performed a manual review of the entire source code base to ascertain compliance with the additional criteria set forth by the state of New York. An automated review was also performed utilizing the Understand tool to assist in finding data issues not easily visible. Understand enables static code analysis through various visuals and report and assists with code comprehension. The review was performed as detailed in the List of Methodology Criteria below.

List of Methodology Criteria

1. 10% review of the source code against the VVSG 1.0 2005 or the declared standard as allowed by the VVSG 1.0 2005.
2. Review of source and COTS delivered to SLI to determine if any COTS has been modified by the Vendor. All COTS products found to have been modified are added to the Vendor created source and subject to full review against the NY standards. This portion of the review utilizes the Ciber_COTSStandard.pdf criteria for determining if COTS products have been vendor modified.
3. Full review of the source code and any vendor modified COTS against the NY standards. For this portion of the review the following criteria are used.
    a. 2017NYElectionLaw.pdf
        i. 6209.2(g)
        ii. 6209.2(10)(i)

    SLI source code review focuses on the following areas of risk to ensure coverage of the standards:

    - Dynamically Loaded Libraries – this has potential to inadvertently deliver malicious code to an operational machine.
    - Embedded malicious SQL/HTML commands – developers often use embedded commands to help programs run efficiently. Certain commands, such as the SQL DELETE, should not be used.

- Password Management – passwords should not be stored in plaintext, hard coded, initialized as null or “empty”, stored in a configuration file, or poorly encoded.

- System Integrity – This is performed by utilizing the Understand tool along with the table of known vulnerabilities to check for the following requirements:
    - Sections of code should not be "commented out".
    - Objects and Functions should not be defined in Header Files.
    - (Required) Floating-point expressions shall not be directly or indirectly tested for equality or inequality.
    - The "goto" statement shall not be used.
    - Source will not contain Unreachable Code.
    - Every defined function shall be called at least once.
    - Find Local Variables that are defined but not used.
    - Find Static Global Variables that are defined but not used.
    - Warn about assigning non-{0,1} values to Boolean variables.
    - Check for logical errors for function calls and Objective-C message expressions (e.g., uninitialized arguments, null function pointers, and pointer to undefined variables).
    - Check when casting a malloc'ed type T, whether the size is a multiple of the size of T.
    - Check for cast from non-struct pointer to struct pointer.
    - Check for cases where the dynamic and the static type of an object are unrelated.
    - Check for assignment of a fixed address to a pointer.
    - Check for pointer arithmetic on locations other than array elements.
    - Check for pointer subtractions on two pointers pointing to different memory chunks.
    - Check for division by variable that is later compared against 0. Either the comparison is useless or there is division by zero.
    - Check virtual function calls during construction or destruction
    - Check unreachable code.
    - Warn about buffer.
    - Check for overflows in the arguments to malloc().
    - Check for an out-of-bound pointer being returned to callers.
    - Check improper use of chroot.

- Check for misuses of stream APIs.
- Check stream handling functions.
- Check for overlap in two buffer arguments.
- Check for arguments which are not null-terminating strings.
- Check for out-of-bounds access in string functions.
- Check for logical errors for function calls and Objective-C message expressions (e.g., uninitialized arguments, null function pointers).
- Check for division by zero.
- Check for null pointers passed as arguments to a function whose arguments are references or marked with the 'nonnull' attribute.
- Check for dereferences of null pointers.
- Check that addresses to stack memory do not escape the function.
- Check for undefined results of binary operators.
- Check for declarations of VLA of undefined or zero size.
- Evaluate "panic" functions that are known to not return to the caller.
- Check for uninitialized values used as array subscripts.
- Check for assigning uninitialized values.
- Check for uninitialized values used as branch conditions.
- Check for blocks that capture uninitialized values.
- Check for uninitialized values being returned to the caller.
- Check for double-free and use-after-free problems. Traces memory managed by new/delete.
- Check for memory leaks. Traces memory managed by new/delete.
- Check for values stored to variables that are never read afterwards.
- Warn when a null pointer is passed to a pointer which has a _Nonnull type.
- Warn when a null pointer is returned from a function that has _Nonnull return type.
- Warn when a nullable pointer is dereferenced.
- Warn when a nullable pointer is passed to a pointer which has a _Nonnull type.
- Warn when a nullable pointer is passed to a pointer which has a _Nonnull type.
- Check MPI code.

- Check that NSLocalizedString macros include a comment for context.
- Warn about uses of non-localized NSStrings passed to UI methods expecting localized NSStrings.
- Check for excessively padded structs.
- Warn on using a floating point value as a loop counter (CERT: FLP30-C, FLP30-CPP).
- Warn on uses of functions whose return values must be always checked.
- Warn on uses of the 'getpw' function.
- Warn on uses of the 'gets' function.
- Warn when 'mkstemp' is passed fewer than 6 X's in the format string.
- Warn on uses of the 'mktemp' function.
- Warn on uses of the 'rand', 'random', and related functions.
- Warn on uses of the 'strcpy' and 'strcat' functions.
- Warn on uses of the 'vfork' function.
- Check calls to various UNIX/Posix functions.
- Check for memory leaks, double free, and use-after-free problems. Traces memory managed by malloc()/free().
- Check for dubious malloc arguments involving sizeof.
- Check for mismatched deallocators.
- Check for proper usage of vfork.
- Check the size argument passed into C string functions for common erroneous patterns.
- Check for null pointers being passed as arguments to C string functions.

All results from the Automated tool are reviewed manually to determine if the finding is a false positive or actual issue. All findings, from both manual and automated reviews, determined to be an actual issue are recorded in detail in the 3.0 Overview of Findings section.

3.0 Overview of Findings

Items found to be non-compliant during this review are described in the table below. For this review, there were no issues found.

File Name	Line number	Description
N/A	N/A	N/A

4.0 Conclusion

SLI found the DVS ICE 4.14.25 source code to meet all the requirements detailed in this document.

SLI Source Code Review
Version 3.0

SLI Compliance
4720 Independence St
Wheat Ridge, CO 80033

303.215.5853
www.slicompliance.com

NYSBOE Dominion Source Code Review Findings
ImageCast Evolution

1.0 Introduction

SLI Compliance (SLI) certification efforts for the New York State Board of Elections (NYSBOE), consist of verifying all source code modules provided to the state for compliance with the best practices and coding requirements defined in the following standards:

- Voluntary Voting System Guidelines (VVSG) 1.0 2005 (Volume I and Volume II)
    - C C++ Coding Standard.pdf
- 2017NYElectionLaw.pdf
    - NYS_voting_systems_standards-4-20(6209).pdf
- Ciber_COTSStandard.pdf
- Table of known vulnerabilities

This evaluation included a secure source code review to ensure protection against all known and identified vulnerabilities reported in prior ITA reports, voting system tests, or risk assessment final reports, and other comparable examinations performed by independent testing organizations.

All source code submitted for certification is compiled and reviewed at the SLI Compliance test laboratory by SLI qualified test engineers. The build scripts, vendor source code, and all modified Commercial Off-the-Shelf (COTS) code is reviewed for format, structure and functionality. Source code is also reviewed using both automated and manual methods for malicious code, Trojans, and viruses.

The Dominion Voting Systems (DVS) ImageCast Evolution (ICE) is an addition to the current DVS configuration certified by NYSBOE. The ICE 4.14.25 source code was delivered to SLI from Pro V&V on January 12, 2018. SLI performed an examination of the ICE code including a review for conformance against the best practices and coding requirements outlined in the abovementioned standards, noting that the “C/C++ Coding Standard declared by Dominion, superseded a subset of VVSG requirements.

2.0 Scope of Review and Methodology

The objective of the source code review was to analyze the ICE 4.14.25 software and to confirm that the code complies with the criteria set forth by the state of New York.

The ICE source code was subjected to a complete review, by Pro V&V for EAC certification, against the source code criteria of the C C++ Coding Standards.pdf as allowed by the VVSG 2005 prior to delivery to SLI. SLI performed a review of approximately 10% of the modified source code lines in order to verify compliance with the declared standard. In addition to the 10% sample, SLI performed a review of the entire source code base to ascertain compliance with the additional criteria set forth by the state of New York. The Understand code analysis tool was utilized to assist in finding data issues not easily visible through manual review. Understand enables static code analysis through various visuals and reports and assists with code comprehension. The review was performed as detailed in the List of Methodology Criteria below.

List of Methodology Criteria

A 10% review of the source code was conducted against the VVSG 1.0 2005 using the Dominion C C++ Coding Standard.pdf as allowed by the VVSG 1.0 2005. The examination was followed by a full review of source and all vendor modified COTS delivered to SLI to determine if any malicious code was present or had been introduced to the system. The full review utilized the Understand tool for code analysis. All results from the automated tool were reviewed manually to determine if a finding was a false positive or actual issue. All findings were recorded in detail in the 3.0 Overview of Findings section.

The review criteria is listed in the following tables

Criterion	VVSG 2005	Exempted by Accepted Standard	Covered by Understand Review and Results
Self-modifying code	v.1: 5.2.2		Yes
Specific function	v.1: 5.2.3.a		Yes
Module has unique name	v.1: 5.2.3.b		Yes
Module has header	v.1: 5.2.3.b, 5.2.7 (a, a.1-a.6)		Yes
Required resources	v.1: 5.2.3.c		Yes

Criterion	VVSG 2005	Exempted by Accepted Standard	Covered by Understand Review and Results
File's functions' line count	v.1: 5.2.3.d, v.2: 5.4.2.i	Yes	Yes
Single Entry Point	v.1: 5.2.3.e		Yes
Single Exit Point	v.1: 5.2.3.e		Yes
Control structures	v.1: 5.2.3.f		Yes
Acceptable Constructs	v.1: 5.2.4.a, v2: 5.4.1		Yes
Vendor Defined Constructs with Justification	v.1:5.2.4.a.i		Yes
Execution through Control Constructs	v.1:5.2.4.a.ii		Yes
Program re-direction	v.1:5.2.4.a.iii		Yes
Class, function and variable names	v.1:5.2.5.b, v.1:5.2.5.c	Yes	
Keyword	v.1: 5.2.5.d	Yes	
Variables	v.1: 5.2.7.b	Yes	Yes
In-Line Comments	v.1: 5.2.7.c	Yes	Partial
Assembly code	v.1: 5.2.7.d	Yes	Yes
Comments in uniform format	v.1: 5.2.7.e	Yes	Yes
Uniform calling sequences	v.2: 5.4.2.a	Yes	Yes
Parameters type and range validation	v.2: 5.4.2.a	Yes	Yes
Explicit return values	v.2: 5.4.2.b	Yes	Yes
Macros	v.2: 5.4.2.c	Yes	Yes
Unbound arrays	v.2: 5.4.2.d	Yes	Yes
Pointers	v.2: 5.4.2.e	Yes	Yes
Case statements	v.2: 5.4.2.f	Yes	Yes
Vote counter overflowing	v.2: 5.4.2.g	Yes	Yes
Indentation	v.2: 5.4.2.h	Yes	Yes
Code generator	v.2: 5.4.2.j	Yes	Yes

Criterion	VVSG 2005	Exempted by Accepted Standard	Covered by Understand Review and Results
Line length	v.2: 5.4.2.k	Yes	Yes
Executable statement	v.2: 5.4.2.l	Yes	Yes
Embedded executable statement	v.2: 5.4.2.m	Yes	Yes
Mixed-mode operations	v.2: 5.4.2.n	Yes	Partial
Exit() message	v.2: 5.4.2.o	Yes	
Format of messages	v.2: 5.4.2.p	Yes	Yes
References variables	v.2: 5.4.2.q	Yes	Yes
Levels of indented scope	v.2: 5.4.2.r	Yes	Yes
Variable initialization	v.2: 5.4.2.s	Yes	Yes
Constant Definitions	v.2: 5.4.2.t	Yes	Yes
Ternary Operator	v.2: 5.4.2.u	Yes	
Assert() statement	v.2: 5.4.2.v	Yes	

Table of Known Vulnerabilities Reviewed	
Buffer Overflow	Writing outside the bounds of allocated memory can corrupt data, crash the program, or cause the execution of an attack payload.
Command Injection	Executing commands from an entrusted source or in an entrusted environment can cause an application to execute malicious commands on behalf of an attacker.
Buffer Over Flow: Format String	Allowing an attacker to control a function's format string may result in a buffer overflow.
Illegal Pointer Value	This function can return a pointer to memory outside of the buffer to be searched. Subsequent operations on the pointer may have unintended consequences.
Integer Overflow	Not accounting for integer overflow can result in logic errors or buffer overflows.
Log Forging	Writing invalidated user input into log files can allow an attacker to forge log entries or inject malicious content into logs.
Path Manipulation	Allowing user input to control paths used by the application may enable an attacker to access otherwise protected files.

Process Control	Executing commands or loading libraries from an entrusted source or in an entrusted environment can cause an application to execute malicious commands (and payloads) on behalf of an attacker.
Resource Injection	Allowing user input to control resource identifiers may enable an attacker to access or modify otherwise protected system resources.
Setting Manipulation	Allowing external control of system settings can disrupt service or cause an application to behave in unexpected ways.
SQL Injection	Constructing a dynamic SQL statement with user input may allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.
String Termination Error	Relying on proper string termination may result in a buffer overflow.
Struts: Duplicate Validation Forms.	Multiple validation forms with the same name indicate that validation logic is not up-to-date.
Struts: Erroneous validate() Method.	The validator form defines a validate() method but fails to call super.validate().
Struts: Form Bean Does Not Extend Validation Class.	All Struts forms should extend a Validator class.
Struts: Form Field Without Validator	Every field in a form should be validated in the corresponding validation form.
Struts: Plug-in Framework Not In Use	Use the Struts Validator to prevent vulnerabilities that result from unchecked input.
Struts: Unused Validation Form	An unused validation form indicates that validation logic is not up-to-date.
Struts: Unvalidated Action Form	Every Action Form must have a corresponding validation form.
Struts: Validator Turned Off	This Action Form mapping disables the forms validate() method.
Struts: Validator Without Form Field	Validation fields that do not appear in forms they are associated with indicate that the validation logic is out of date.
Unsafe JNI	Improper use of the Java Native Interface (JNI) can render Java applications vulnerable to security bugs in other languages. Language-based encapsulation is broken.
Unsafe Reflection	An attacker may be able to create unexpected control flow paths through the application, potentially bypassing security checks.
XML Validation	Failure to enable validation when parsing XML gives an attacker the opportunity to supply malicious input.
Dangerous Function	Functions that cannot be used safely should never be used.

Directory Restriction	Improper use of the chroot() system call may allow attackers to escape a chroot jail.
Heap Inspection	Do not use realloc() to resize buffers that store sensitive information.
J2EE Bad Practices: getConnection()	The J2EE standard forbids the direct management of connections.
J2EE Bad Practices: Sockets	Socket-based communication in web applications is prone to error.
Often Misused: Authentication	Do not rely on the name getlogin() family of functions returns because it is easy to spoof.
Often Misused: Exception Handling	A dangerous function can throw an exception, potentially causing the program to crash.
Often Misused: File System	Passing an inadequately- sized output buffer to a path manipulation function can result in a buffer overflow.
Often Misused: Privilege Management	Failure to adhere to the principle of least privilege amplifies the risk posed by other vulnerabilities.
Often Misused: Strings	Functions that manipulate strings encourage buffer overflows.
Unchecked Return Value	Ignoring a method's return value can cause the program to overlook unexpected states and conditions.
Insecure Randomness	Standard pseudo-random number generators cannot withstand cryptographic attacks.
Least Privilege Violation	The elevated privilege level required to perform operations such as chroot() should be dropped immediately after the operation is performed.
Missing Access Control	The program does not perform access control checks in a consistent manner across all potential execution paths.
Password Management	Storing a password in plaintext may result in a system compromise.
Password Management: Empty Password in Config File	Using an empty string as a password is insecure.
Password Management: Hard-Coded Password	Hard coded passwords may compromise system security in a way that cannot be easily remedied.
Password Management: Password in Config File	Storing a password in a configuration file may result in system compromise.

Password Management: Weak Cryptography	Obscuring a password with a trivial encoding does not protect the password.
Privacy Violation	Mishandling private information, such as customer passwords or social security numbers, can compromise user privacy and is often illegal.

New York Election Law	
NYS 2007 Election Law 7-202.1f	A voting machine or system to be approved by the state board of elections shall: f. be provided with a "protective counter" which records the number of times the machine or system has been operated since it was built and a "public counter" which records the number of persons who have voted on the machine at each separate election;
NYS 2007 Election Law 7-202.1ri	A voting machine or system to be approved by the state board of elections shall: r. ensure the integrity and security of the voting machine or system by: (i) being capable of conducting both pre-election and post-election testing of the logic and accuracy of the machine or system that demonstrates an accurate tally when a known quantity of votes is entered into each machine
NYS 2007 Election Law 7-202.1rii	A voting machine or system to be approved by the state board of elections shall: r. ensure the integrity and security of the voting machine or system by: (ii) providing a means by which a malfunctioning voting machine or system shall secure any votes already cast on such machine or system
NYS 2007 Election Law 7-202.1t	A voting machine or system to be approved by the state board of elections shall: t. not include any device or functionality potentially capable of externally transmitting or receiving data via the internet or via radio waves or via other wireless means.

NYS Regulation 6209.2.F.1a	(1) The voting system shall print and display a paper record of the voter's ballot choices prior to the voter making the ballot choices final. In the case of a paper-based voting system, the ballot marked by the voter shall constitute the paper record referred to in this Section F. (a) The paper record shall constitute a complete record of ballot choices that can be used in audits of the accuracy of the voting systems electronic records, in audits of the election results, and in full recounts.
NYS Regulation 6209.2.F.4b	(4) The voting system shall allow the voter to approve or reject the paper record, in the case of DRE systems, marking the ballot as such in the presence of the voter. (b) Prior to reaching the maximum number of ballots allowed pursuant to statute, any DRE voting system shall display a warning message to the voter indicating the voter may reject only one more ballot, and that the third ballot shall become the ballot of record.
NYS Regulation 6209.2.F.10a	(10) The voting system's ballot records shall be structured and contain information so as to support highly precise audits of their accuracy. (a) All cryptographic software in the voting system shall have been approved by the U.S. Government's Crypto Module Validation Program (CMVP) as applicable.
NYS Regulation 6209.2.F.12	(12) The voting system shall generate and store a digital signature for each electronic record.
NYS Regulation 6209.2.F.13	(13) The electronic records shall be able to be exported for auditing or analysis on standards-based and/or information technology computing platforms.
NYS Regulation 6209.2.F.13b	(13) The electronic records shall be able to be exported for auditing or analysis on standards-based and/or information technology computing platforms. (b) The voting system shall export the records accompanied by a digital signature of the collection of records, which shall be calculated on the entire set of electronic records and their associated digital signatures.

NYS Regulation 6209.2.G	Any submitted voting system's software shall not contain any code, procedures or other material which may disable, disarm or otherwise affect in any manner, the proper operation of the voting system, or which may damage the voting system, any hardware, or any computer system or other property of the State Board or county board, including but not limited to 'viruses', 'worms', 'time bombs', and 'drop dead' devices that may cause the voting system to cease functioning properly at a future time.
NYS Regulation 6209.6F3	(3) Software Specification The Software Specification shall contain and describe the vendor's design standards and conventions, environment and interface specifications, functional specifications, programming architecture specifications, and test and verification specifications. Vendor must also provide document identification, an abstract of the specification, configuration control status and a table of contents. The body of the specification shall contain the following material:
NYS Regulation 6209.6F3n2	(3) Software Specification: The Software Specification shall contain and describe the vendor's design standards and conventions, environment and interface specifications, functional specifications, programming architecture specifications, and test and verification specifications. Vendor must also provide document identification, an abstract of the specification, configuration control status and a table of contents. The body of the specification shall contain the following material. (n) Security 2. For each potential point of attack, the vendor shall identify the technical safeguards embodied in the voting system to defend against attack, and the procedural safeguards that the vendor has recommended be followed by the election administrators to further defend against that attack. Each defense shall be classified as preventative, if it prevents the attack in the first place; detective if it allows detection of an attack; or corrective if it allows correction of the damage done by an attack.

NYS Regulation 6209.6F3n3	(3) Software Specification: The Software Specification shall contain and describe the vendor's design standards and conventions, environment and interface specifications, functional specifications, programming architecture specifications, and test and verification specifications. Vendor must also provide document identification, an abstract of the specification, configuration control status and a table of contents. The body of the specification shall contain the following material. (n) Security 3. Security requirements and provisions shall include the ability of the system to detect, prevent, log and recover from the broad range of security risks identified.
NYS Regulation 6209.6F3o	(3) Software Specification: The Software Specification shall contain and describe the vendor's design standards and conventions, environment and interface specifications, functional specifications, programming architecture specifications, and test and verification specifications. Vendor must also provide document identification, an abstract of the specification, configuration control status and a table of contents. The body of the specification shall contain the following material. (o) Programming Specifications The vendor shall provide an overview of the software design, structure and implementation algorithms. Whereas the Functional Specification of the preceding section provides a description of what functions the software performs and the various modes in which it operates, this section should be prepared so as to facilitate understanding of the internal functioning of the individual software modules. Implementation of functions shall be described in terms of software architecture, algorithms and data structures and all procedures or procedure interfaces which are vulnerable to degradation in data quality or security penetration shall be identified.
BMD 2.3.3	Ballots shall include machine-readable code and also human-readable code, to identify different ballot styles.

BMD 2.6.3	The BMD shall contain software and hardware required to perform a diagnostic test of system status, to demonstrate that the system is fully operational and that all voting positions are operable.
BMD 2.7.1	a BMD shall meet the following provisions: Be constructed to allow for a voter to mark a paper ballot for all candidates who may be nominated and on all ballot proposals which may be submitted.
BMD 2.7.2	a BMD shall meet the following provisions: The BMD shall provide a method for a voter to mark a paper ballot indicating their selection for any person for any office, whether nominated as a candidate (write-in) by any party or independent body.
BMD 2.7.3	a BMD shall meet the following provisions: Be constructed so that a voter cannot mark a ballot for a candidate or for a ballot proposal for whom or on which he or she is not lawfully entitled to vote.
BMD 2.7.4	a BMD shall meet the following provisions: The BMD must prevent voters from over-voting and indicate to the voter specific contests or ballot issues for which no selection or an insufficient number of selections has been made and provide the voter with the opportunity to correct the ballot before the ballot is marked.
BMD 2.7.5	a BMD shall meet the following provisions: Provide an opportunity such that any voter, including voters who are blind or visually impaired, may privately and independently verify their selections and the ability to privately and independently change such selections or correct any error before the ballot is marked.
BMD 2.7.6	a BMD shall meet the following provisions: Provide a feature to permit a voter to independently verify their paper ballot after it has been marked, including voters who are blind or visually impaired.
BMD 3.6.1.3	3.6.1. BMD and other devices accessible to individuals with disabilities, must be approved for use by the NYSBOE prior to use. Each complete BMD, all documentation prescribed herein, must be submitted to the NSYBOE for testing purposes no later than 11 am Eastern Standard Time ten (10) business days after a bid opening. Deliveries must be completed as inside delivery and include the following: 3.6.1.3. The election management software.

The following is a full list of all documents and tools and the corresponding requirements used during the review.

Automated review using the Understand tool:

The following checks are performed by the Understand tool during the review process:

- Definitions in Header Files - RECOMMENDED_01
  Description: Objects and Functions should not be defined in Header Files.
- Floating-point expressions shall not be directly or indirectly tested for equality or inequality. - RECOMMENDED_03
  Description: (Required) Floating-point expressions shall not be directly or indirectly tested for equality or inequality.
- Functions Too Long - RECOMMENDED_04
  Description: Program units should not have more than the specified number of lines
- Goto Statements - RECOMMENDED_06
  Description: The "goto" statement shall not be used.
- Macros shall not be #define'd or #undef'd within a block - RECOMMENDED_07
  Description: Macros shall not be #define'd or #undef'd within a block
- Magic Numbers - RECOMMENDED_08
  Description: All fixed values will be defined constants.
- Overly Complex Functions - RECOMMENDED_10
  Description: Tes the McCabe Cyclomatic Complexity for program units.
- Unreachable Code - RECOMMENDED_12
  Description: Source will not contain Unreachable Code
- Variables should be commented - RECOMMENDED_16
  Description: Each variable declaration should have a comment
- BoolAssignment
  Description: Warn about assigning non-{0,1} values to Boolean variables
- CallAndMessageUnInitRefArg
  Description: Check for logical errors for function calls and Objective-C message expressions (e.g., uninitialized arguments, null function pointers, and pointer to undefined variables)
- CastSize
  Description: Check when casting a malloc'ed type T, whether the size is a multiple of the size of T
- CastToStruct
  Description: Check for cast from non-struct pointer to struct pointer

- DynamicTypeChecker
  Description: Check for cases where the dynamic and the static type of an object are unrelated.
- alpha\core\FixedAddr
  Description: Check for assignment of a fixed address to a pointer
- IdenticalExpr
  Description: Warn about unintended use of identical expressions in operators
- alpha\core\PointerArithm
  Description: Check for pointer arithmetic on locations other than array elements
- PointerSub
  Description: Check for pointer subtractions on two pointers pointing to different memory chunks
- SizeofPtr
  Description: Warn about unintended use of sizeof() on pointer expressions
- TestAfterDivZero
  Description: Check for division by variable that is later compared against 0. Either the comparison is useless or there is division by zero.
- VirtualCall
  Description: Check virtual function calls during construction or destruction
- UnreachableCode
  Description: Check unreachable code
- ArrayBoundV2
  Description: Warn about buffer overflows (newer checker)
- ArrayBound
  Description: Warn about buffer overflows (older checker)
- MallocOverflow
  Description: Check for overflows in the arguments to malloc()
- ReturnPtrRange
  Description: Check for an out-of-bound pointer being returned to callers
- TaintPropagation
  Description: Generate taint information used by other checkers
- Chroot
  Description: Check improper use of chroot
- PthreadLock
  Description: Simple lock -> unlock checker
- SimpleStream
  Description: Check for misuses of stream APIs
- Stream
  Description: Check stream handling functions

- BufferOverlap
  Description: Checks for overlap in two buffer arguments
- NotNullTerminated
  Description: Check for arguments which are not null-terminating strings
- OutOfBounds
  Description: Check for out-of-bounds access in string functions
- CallAndMessage
  Description: Check for logical errors for function calls and Objective-C message expressions (e.g., uninitialized arguments, null function pointers)
- DivideZero
  Description: Check for division by zero
- DynamicTypePropagation
  Description: Generate dynamic type information
- NonNullParamChecker
  Description: Check for null pointers passed as arguments to a function whose arguments are references or marked with the 'nonnull' attribute
- NullDereference
  Description: Check for dereferences of null pointers
- StackAddressEscape
  Description: Check that addresses to stack memory do not escape the function
- UndefinedBinaryOperatorResult
  Description: Check for undefined results of binary operators
- VLASize
  Description: Check for declarations of VLA of undefined or zero size
- BuiltinFunctions
  Description: Evaluate compiler builtin functions (e.g., alloca())
- NoReturnFunctions
  Description: Evaluate "panic" functions that are known to not return to the caller
- ArraySubscript
  Description: Check for uninitialized values used as array subscripts
- Assign
  Description: Check for assigning uninitialized values
- Branch
  Description: Check for uninitialized values used as branch conditions
- CapturedBlockVariable
  Description: Check for blocks that capture uninitialized values
- UndefReturn
  Description: Check for uninitialized values being returned to the caller

- NewDelete
  Description: Check for double-free and use-after-free problems. Traces memory managed by new/delete.
- NewDeleteLeaks
  Description: Check for memory leaks. Traces memory managed by new/delete.
- DeadStores
  Description: Check for values stored to variables that are never read afterwards
- NullPassedToNonnull
  Description: Warns when a null pointer is passed to a pointer which has a _Nonnull type.
- NullReturnedFromNonnull
  Description: Warns when a null pointer is returned from a function that has _Nonnull return type.
- NullableDereferenced
  Description: Warns when a nullable pointer is dereferenced.
- NullablePassedToNonnull
  Description: Warns when a nullable pointer is passed to a pointer which has a _Nonnull type.
- NullablePassedToNonnull
  Description: Warns when a nullable pointer is passed to a pointer which has a _Nonnull type.
- MPI-Checker
  Description: Checks MPI code
- EmptyLocalizationContextChecker
  Description: Check that NSLocalizedString macros include a comment for context
- NonLocalizedStringChecker
  Description: Warns about uses of non-localized NSStrings passed to UI methods expecting localized NSStrings
- Padding
  Description: Check for excessively padded structs.
- FloatLoopCounter
  Description: Warn on using a floating point value as a loop counter (CERT: FLP30-C, FLP30-CPP)
- UncheckedReturn
  Description: Warn on uses of functions whose return values must be always checked
- getpw
  Description: Warn on uses of the 'getpw' function
- gets
  Description: Warn on uses of the 'gets' function

- mkstemp
  Description: Warn when 'mkstemp' is passed fewer than 6 X's in the format string
- mktemp
  Description: Warn on uses of the 'mktemp' function
- rand
  Description: Warn on uses of the 'rand', 'random', and related functions
- strcpy
  Description: Warn on uses of the 'strcpy' and 'strcat' functions
- vfork
  Description: Warn on uses of the 'vfork' function
- API
  Description: Check calls to various UNIX/Posix functions
- Malloc
  Description: Check for memory leaks, double free, and use-after-free problems. Traces memory managed by malloc()/free().
- MallocSizeof
  Description: Check for dubious malloc arguments involving sizeof
- MismatchedDeallocator
  Description: Check for mismatched deallocators.
- Vfork
  Description: Check for proper usage of vfork
- unix\cstring\BadSizeArg
  Description: Check the size argument passed into C string functions for common erroneous patterns
- NullArg
  Description: Check for null pointers being passed as arguments to C string functions

For verification purposes, the .ini file used for the review is included in the Supplementals.zip file provided along with this report.

Manual Review:

A manual review was done utilizing key word searches to examine the code for instances of malicious code. This review was done utilizing the Table of Known Vulnerabilities to find areas of source code that may contain code commonly used by malicious software. During this review, the following standard was also employed in order to verify VVSG compliance of requirements not covered in the automated review.

C C++ Coding Standards.pdf

The following coding standards were utilized during the review:

2.0.1 Obvious Constraints

• No self-modifying code

• No 'exit' statements (except from the main routine)

• No 'go-to' statements, or 'while (0) {}' constructs

• Names used in code and in documentation must be consistent

• C/C++ language keywords shall not be used as names of functions, variables, structures, classes, etc.

2.0.2 Functions

• Functions must be clear and concise

• All functions must be preceded by a header - See section 2.0.11.4 for details.

2.0.2.1 Function Names

• Abbreviations within function names are permitted, but their meaning should be clear

2.0.2.2 Input Parameters

• Parameter ranges must be validated on entry, or indicated in the header comments. If the caller guarantees that the parameters are within range, a comment stating so may instead be provided in the header.

2.0.2.3 Returns

• All non-void functions must have, at most, a single explicit 'return' statement at the end. Midroutine returns should only be used for cases "...so severe that execution cannot be resumed".
• If a function detects an error and cannot complete, it may return a value indicating the error or throw an exception.

• Functions which return a pointer should return NULL (i.e. 0) to indicate failure.

2.0.2.4 Control Constructs

• Program flow should be based on simple combinations of the following constructs: – if-then-else – for-loop – do-while – do-until – switch/case

2.0.3 Variables

Keep variables in the smallest possible scope (i.e. try to use locals where possible and avoid using globals).

2.0.3.1 Variable Names

• Single character variable names should not be used, except for: – loop variables, or – mathematical or scientific standards (e.g. X and Y co-ordinates).

• Variable names should include at least one noun or verb. Any tense or form is allowed, and abbreviations are permitted.

• The component parts of a variable name may be formed from whole words, abbreviations of words, or numbers, if they are significant to the meaning of the variable. Names may also contain the underscore character (‘ ’), in order to separate the name into its component parts (see below).
• Please take care to ensure that any variable name does not form a word that is offensive in any way.

2.0.3.2 Abbreviations

Variable names may be composed of full words, or abbreviations of full words. Two possible methods of abbreviating words are:

• Eliminate vowels (e.g. prnt can replace “print”)

• Use the first 3 or more letters of the intended word (e.g. ball for Ballot)All abbreviations and acronyms used in variable names must be added to the DVS “Glossary of Abbreviated Terms”. This Glossary should be maintained in the code repository of each project, and should be kept up-to-date for reference by other programmers and code reviewers. A Sample Glossary is found in Appendix ‘A’ of this document.

2.0.3.3 Compound Variable Names

As mentioned above, variable names may be built-up of concatenations of several component parts. Each component part may be either a full word, or an abbreviation. At least one of the component parts should be either a noun or a verb (or an abbreviation of such). In order to make the variable name meaning more obvious, highlight where each component part begins. Use the following techniques:

• Capitalize the first letter of each component (after the first), OR

• Separate component parts with an underscore character (‘ ’). Example: A variable containing the “Previous Rotation Association Found” could be either:

prevRotAssocFnd

OR

prev_rot_assoc_fnd

See section 2.0.3.4 below concerning comments for variable names made of three or more abbreviations.

2.0.3.4 Variable Declaration and Initialization

• Initialize every variable upon declaration, and comment its use. Variables which share the same meaning may share the same comment.

• Member variables of a class must be initialized in the class constructor(s), either directly or indirectly (i.e. by calling a routine specifically intended to perform initialization). They must be commented in the header file that defines the class.

• If a variable name contains three or more abbreviations, then the declaration comment must explain the full significance of each component abbreviation.

UINT32 prevRotAssocFnd=0; // Previous Rotation Association Found

2.0.4 Constants

• Constants other than 0 and 1 should be defined or enum'd (either in a header file, or the specific source file, if pertinent to that file only).

• Use of non-defined constants (other than 0 or 1) must be clearly commented.

• Names defined to replace constants should be intuitive.

2.0.5 Macros

• Macros cannot include a 'return' statement or pass control beyond the next statement.

2.0.6 Conditionals

2.0.6.1 Explicit Comparisons

• Explicit comparisons are recommended, but not required. Both of the following are permitted: if (bValidResponse == TRUE) // valid if (bValidResponse) // valid

2.0.6.2 Embedded Assignment Statements

• Embedding assignments within a conditional is permitted, but only one per line – i.e. the following requires two lines: if (((rc1 = myRoutine()) == SUCCESS) && ((rc2 = hisRoutine()) == FAILURE))

2.0.7 Switch Statements

• All switch statements must explicitly include the 'default' case.

2.0.8 Assert Statements

• Use ASSERT instead of ‘assert’, which permits disabling of all ‘asserts’ in production code.

2.0.9 Error Checking

The code must take specific precautions against unexpected faults and memory corruption in the following areas specifically:

2.0.9.1 Bounds Checking

• Ensure array subscripts and pointer variables are within range, so that arrays, strings and dynamically allocated memory accesses are all valid. If all calling routines ensure incoming parameters are within range, it is not necessary for the called routine to do so, but then the function header comments must indicate this.

• Specifically, all calls to ‘malloc’ must check for a NULL (0) returned value.

• Pointers may be initialized to NULL (0) or to the intended memory location.

2.0.9.2 Counter Overflow

• Specifically noted in the testing guidelines: Code must explicitly test to ensure that ‘vote counters’ do not overflow. “Assuming the counter size is large enough... is not adequate”.

2.0.10 Code Simplicity / Legibility

The code must be easily understood by the personnel at the lab doing the certification. These points are meant to make their job of understanding our code easier.

2.0.10.1 Line Count

A ‘line’ here is defined as an executable or control line plus its formatting and comment lines.

• No more than 50% of all functions should exceed 60 lines in length.

• No more than 5% of all functions should exceed 120 lines in length.

• No functions should exceed 240 lines in length. If necessary to exceed, justify in comments.

2.0.10.2 Indirection

• Do not use more than 4 levels of indirection Example: a -> b -> c -> d [e] // too complex

2.0.10.3 Nesting

• Do not use more than 5 levels of indented scope.

2.0.11 Formatting

Code must be formatted according to the following rules, and contains headers as described below:

2.0.11.1 Code Formatting

• All code lines must be less than or equal to 120 characters in width. Place comment lines ahead of statements if too wide.

• All module header line comments (for Files, Classes and Function headers) must be less than or equal to 120 characters in width.

2.0.11.2 File Headers

• State the purpose of the unit and how it works

• Include the date of creation and revision record • Use the following sample as a template for file headers:

/***
** ** THE FILE CONTAINS PROPRIETARY INFORMATION ** ** The material and information contained herein may not be reproduced, copied, ** printed or disclosed for any purpose to any person or organization, except ** as may be consistent with the terms and conditions of a license agreement or ** non-disclosure agreement by Dominion Voting. All copies must contain this ** trade secret warning. The material contained herein is the property of ** Dominion Voting and is considered confidential and proprietary information. ** ** Dominion Voting Systems Corporation ** Toronto, Ontario, Canada **
***/
/***
** ** File name: vscan_drv.cpp ** Module name: SCANNER (Layer2) ** Description: - All scanner transport controls ** - Acquire images from CSIs (front & back) ** ** Author: Shandon Wong ** ** Revision History ** yyyy.mm.dd Author Description ** ---- -- -- -------- ----------** 2006.12.14 JRB Extracted from CF200 code ** 2007.01.03 JRB Added return codes for all routines ** 2006.01.14 Shandon ICP-1035 - incorrect return code from scan_it **
***/

2.0.11.3 Doxygen complaint File Headers

• Proprietary information (mandatory)

• \file – name of the file (mandatory tag)

• \brief – brief description (mandatory tag)

• Author: – name of the person who performed last modification on file (mandatory)

• Revision: – svn version of last modification on file (mandatory)

• Date: – date and time of file last modification (mandatory)

• \verbatim – log of file changes (revision history) (mandatory tag)

/***
** ** THE FILE CONTAINS PROPRIETARY INFORMATION

** ** The material and information contained herein may not be reproduced, copied, ** printed or disclosed for any purpose to any person or organization, except ** as may be consistent with the terms and conditions of a license agreement or ** non-disclosure agreement by Dominion Voting. All copies must contain this ** trade secret warning. The material contained herein is the property of ** Dominion Voting and is considered confidential and proprietary information. ** ** Dominion Voting Systems Corporation ** Toronto, Ontario, Canada ** ***/
/*! * \file InitialVerifier.h * \brief Initial verifier declaration * $Author: peter $ * $Revision: 6 $ * $Date: 2011-06-22 10:58:40 -0400 (Wed, 22 Jun 2011) $ * * \verbatim * Revision History * yyyy.mm.dd Author Revision Description * ---- -- -- -------- -------- ----------* 2010.01.14 pirke 3692 Class InitialVerifier added * 2010.01.14 pirke 3700 MachineContext initialization moved to * InitialVerifier class * 2010.01.14 pirke 3729 ProgressList inherits from StringAsArgument class. * 2010.02.05 pirke 4180 Machine behavior settings persistence * \endverbatim */

2.0.11.4 Function Headers

Use the following as a template for all functions with greater than, or equal to, 10 lines of code. Note that if no Globals are used in the module, do not include the line “Globals Used”. Similarly, if no Future Enhancements are envisioned, or no File Access is used, do not include the lines indicating their non-existence. All other fields are mandatory.

/***
** ** Function: ScanDivert ** ** Purpose: Send ballot through the Divertor Slot ** ** Description: ** Advance ballot until tail end is beyond Paper Sensor 4, (i.e. the divertor slot). ** Then reverse, which should cause the ballot to back up through the slot. While ** reversing, monitor Paper Sensor 3, if it becomes active, retry the whole process. ** ** Future enhancement: Monitor Paper Sensor 5 to ensure ballot drops **

** Input: scan_data -> scan_data_t structure, for global scanning parameters ** ** Return: rc = VSCAN_OK = 0 => Success ** = VSCAN_DRV_PAPER_JAM_ERROR. This could be for 2 reasons: ** 1) Advanced further than expected without PS4 becoming False ** or 2) Advanced OK, but reversing kept passing PS3. ** = VSCAN_DRV_ERROR => Low level Motor routine error.(rc from ioctl) ** ** Globals used: gSecurity, gpSDManager ** ** File Access: The file specified to function scan_open is

opened and read. ** ** Call tree: ** ScanDivert ** | CDvsUARTDrv::CDvsUARTDrv ** | CDvsQSPI::CDvsQSPI ** | CDvsEdeviceDrv::CDvsEdeviceDrv ** | CDvsEdeviceDrv::Modem ** | CDvsEdeviceDrv::CreateBuffer ** | EDevice::SetDrv ** | EDevice::SetUartDrv ** | CDvsUARTDrv::~CDvsUARTDrv (Virtual) ** | CDvsQSPI::~CDvsQSPI (Virtual) ** | CDvsPrinterQSPI::~CDvsPrinterQSPI (Virtual) ** ** Revision History: ** yyyy.mm.dd Author Description ** ---- -- -- -------- ----------** 2010.09.14 JRB Initial Revision **
**********/

2.0.11.5 Functions containing less than 10 lines of code

For functions less than 10 lines of code, either the above, or the following abbreviated header may be used.

/**********
** ** Function: scan_divert ** ** Revision History: ** yyyy.mm.dd Author Description ** ---- -- -- -------- ----------** 2010.09.14 JRB Initial Revision **
**********/

2.0.11.6 Doxygen complaint function header

Doxygen complaint function header:

• \fn – function signature (mandatory tag)

• \brief – brief description (mandatory tag)

• \details – detail description (not mandatory tag)

• \param – description of function parameter (this tag repeats for each function parameter. If function has no parameters this tag is omitted)

• \return – description of function return value (mandatory tag; if function has no return value add '\return none.')

• \note Globals used: – list of used global variables in function (if no global variables are used this tag is omitted)

• \note File access: – list of files accessed in function and method of access (i.e., read, write, modify or append) (if no files is accessed this tag is omitted)

• \verbatim – function revision history and function call tree (mandatory tag) Use the following as a template for all functions with greater than or equal 10 lines of code:

/*! * \fn bool FileSerializer::readDomDocumentFromFile(QDomDocument& document, * const QString& fullPath) * * \brief Read DomDocument from file * * \details Function opens the file at

specified full path and set is as content to given * dom document * * \param[out] document * \param[in] fullPath File path * * \return Success flag * * \note Globals used: gFile * * \note File access: The file specified to function readDomDocumentFromFile is opened * and read. * * \verbatim * Revision History * yyyy.mm.dd Author Revision * ---- -- -- -------- ----------* 2010.02.10 pirke 4254 * 2010.04.29 drazha 5682 * 2010.05.25 jovica 6080 * * Call Sequence * - QFile::open * - logQStringMessageDebug * - QDomDocument::setContent * - arg

* - QFile::close * \endverbatim */

2.0.11.7 Doxygen compliant Function Header for functions containing less than 10 lines of code

For functions less than 10 lines of code, either the above, or the following abbreviated header may be used

• \fn – function signature (mandatory tag)

• \verbatim: – function revision history and function call tree (mandatory tag)

/*! * \fn bool FileSerializer::readDomDocumentFromFile(QDomDocument& document, * const QString& fullPath) * * \verbatim * Revision History * yyyy.mm.dd Author Revision * ---- -- -- ----------* 2010.02.10 pirke 4254 * 2010.04.29 drazha 5682 * 2010.05.25 jovica 6080 * * Call Sequence * - QFile::open * - logQStringMessageDebug * - QDomDocument::setContent * - arg * - QFile::close * \endverbatim */

2.0.11.8 Class Headers

Headers for classes should use the following format:

/**********
* ** ** Class: CDvsSecureIo ** ** Base Class: none ** ** Description: This class handles I/O to any file which is encrypted or signed ** It also handles raw files (ie. neither encrypted nor signed) ** ** It supports most standard DvsFile I/O functions, with notable ** exception that a file cannot be opened for Read & Write, nor

** can one Seek in a file opened for Writing. ** ** Revision History: ** yyyy.mm.dd Author Description ** ---- -- -- ----------** 2010.09.14 JRB Created ** **********/

2.0.11.9 Doxygen complaint Class Header

Headers for classes should use the following format:

• \class – class name (mandatory tag)

• \brief – brief description (mandatory tag)

• \details – detail description (not mandatory tag) • \verbatim – class revision history (mandatory tag)

/*! * \class M1Packageable * * \brief Base class for M1Package and M1Class. * * \details * Abstracts the fact that each M1Package element can be * packed into M1Package. * * \verbatim * Revision History * yyyy.mm.dd Author Revision * ---- -- -- -------- ----------* 2009.11.06 pirke 1711 * 2010.04.25 pirke 5501 * \endverbatim */

2.1 Assembler Code • Any Assembler routines must: – have a single entry point and return – follow 'C' like control paths (if-then-else, do-while, etc) – be commented to read like 'C' code

2.2 Classes with Code Fully or Partially Generated Using DVS Applifter plug-in

• File header is located in Preserve section.

• Whole code that should be intact after the next auto generating step is placed inside the Preserve section. Everything else is re-generated according to the model context.

• Preserve section of the Class' header contains description of class functionality.

• Constructor's Preserve section contains addition to the Constructor code.

• Class' Preserve sections (Public, Protected and Private) contain additional Functions declarations from the Class that are documented.

• Class' Preserve section (File Footer) contains additional Functions definitions from the Class that are documented.

• Data members that are auto generated are not commented in source code.

2.3 Checklist

Following is a checklist for C++ code so that it meets our Coding Guidelines. Rules listed under "New Code" are not absolute requirements for EAC certification, and so we need not modify existing code to comply with them. However any newly written code should include them. All Code

• File and Function headers

• 120 column width

• variable Names > 1 char long & differ by > 1 character

• Variable declarations: Assign initial values and comment variable's use

• #define or enum all constants (other than 0 and 1)

• Conditionals: max of 1 embedded assignment per statement

• ‘default’ for all ‘switch’ statements • Use ASSERT instead of assert

• Validate parameter ranges, esp. array subscripts and pointers (or comment in function header)

• Test for divide by zero • Test return values for errors

• No more than 4 levels of indirection

• No more than 5 levels of indent

3.0 Overview of Findings

The potential issues found are categorized and listed in the table below. For this review there were a total of 2,622 potential issues found. All potential issues are listed below.

The reasoning used to determine whether an anomaly found was an issue, is noted in the table below.

Topic	# of Issues found	Reasoning
Security:		
Check virtual function calls during construction or destruction	5	Virtual calls are used for setting values for later use only.
Check unreachable code	15	Statements are pointer initialization.
Check for uninitialized values being returned to the caller	1	Return value initialized by default constructor.
Check for dereferences of null pointers	1	Does not violate the levels of indirection as stated in the C++ Coding Standards.pdf
Check for memory leaks. Traces memory managed by new/delete.	1	If not properly managed there is potential for memory leak to occur
Warn on uses of the 'strcpy' and 'strcat' functions	44	Calls to the ‘strcpy’ function are used for internal logging with fixed values.
VVSG 2005/Vendor Standard:		

Objects and Functions should not be defined in Header Files.	400	Potentially unsafe if an attacker were to gain access to the source. Not a direct violation of any requirements.
Floating-point expressions shall not be tested for equality or inequality	1	Tested variable is assigned a value directly before being tested and is tested against 0.
Program units should not have more than the specified number of lines	1	Accepted as not an issue for consistency with the EAC review.
Macros shall not be #define'd or #undef'd within a block	5	Accepted as not an issue for consistency with the EAC review
All fixed values other than 0 or 1 will be defined constants	1443	Numbers being used as variable initialization are not a violation.
Test the McCabe Cyclomatic Complexity for program units	33	While this may make the code more difficult to review/understand, it is not a direct violation of any requirements.
Source will not contain Unreachable Code	2	Unreachable code is defensive and intentionally used to guard against malfunction.
Each variable declaration should have a comment	665	Accepted as not an issue for consistency with the EAC review

Noncompliant Items

There were no items found to be noncompliant to the requirements identified in the scope above.

4.0 Conclusion

After conducting a manual and automated secure source code review of the Dominion Voting Systems (DVS) ImageCast Evolution (ICE), SLI Compliance found the DVS ICE 4.14.25 source code to meet the requirements detailed in this document.

Appendix A – Accompanying Documentation

For verification purposes, the documents referenced in this report have been provided in the Supplementals.zip file.

- 2017NYElectionLaw.pdf
  - New York's Election Law
- 6210Regulation09052008.pdf
  - Section 6210 from New York's Election Law
- C C++ Coding Standard.pdf
  - Dominion's declared coding standard
- Ciber_COTSStandard.pdf
  - Ciber and NYSTEC's interpretation of the VVSG 2005 requirements as applied to COTS products.
- ICE Codecheck Details.txt
  - Detailed output report from the Understand review
- ICE Codecheck Summary.txt
  - Summary report from the Understand review
- ICE Configuration .ini
  - Set of checks run for the Understand review. Same as above listed checks, able to be directly imported into Understand
- NYS_voting_systems_standards-4-20(6209).pdf
  - Section 6209 from the New York Election Law

SLI Source Code Review Addendum

SLI Compliance
4720 Independence St
Wheat Ridge, CO 80033

303.215.5853
www.slicompliance.com

NYSBOE Dominion – Additional Review
ImageCast Evolution

1.0 Introduction

SLI Compliance conducted an additional review of the Dominion ICE 4.14.25 source code to specifically search for specific printer calls for marking ballots in an effort to uncover any malicious code that may exist.

2.0 Review and Methodology

The following process was utilized by SLI for conducting the review against the ICE 4.14.25 source code. The processed included an automated and manual review of the code.

2.1 Tools Used Review

Understand is a Commercial Source Review tool with a fully integrated IDE. It provides various checks by language to support reviewing source code to commonly accepted standards. The Understand tool also contains a set of data security checks to help safeguard against data loss and corruption.

2.2 Review Process

1. Locate module used for sending information to printer/printing marks on ballot. The print file in question has been identified as Swath.cpp. Its purpose is to send a command to print a swath (one run of the inkjet from left to right or right to left) to the printer.

   The module in question is:

   void Swath::print(QRect leftMarker, QRect rightMarker).

2. Created a list of files that the Swath.cpp file depends on.
3. Created list of files that depend on Swath.cpp
4. Searched ICE Source for occurrences of “swath” and created a list of containing files.

5. Manually verified that all files from all lists did not contain any malicious code.
6. Performed a manual search for any conditions that would prevent an already marked ballot from being passed to the printer.

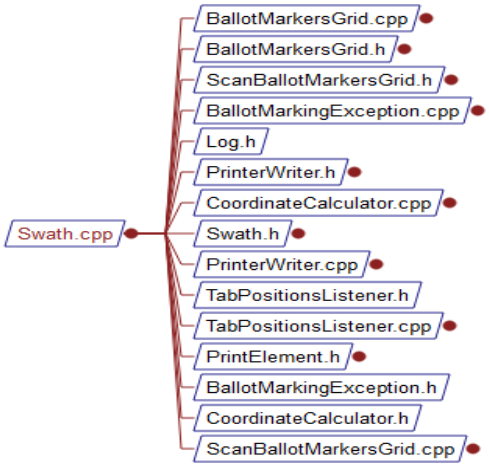
2.3 Module Dependencies Found

Files Containing "Swath"

- LandscapeWriteInSection.cpp
- LandscapeWriteInSection.h
- PrintPage.cpp
- PrintPage.h
- PrinterWriter.cpp
- PrinterWriter.h
- Swath.cpp
- Swath.h
- VotingMarkElement.cpp
- WriteInElement.cpp
- PrinterConfiguration.cpp

Files that Depend on Swath. cpp

Files that Swath.cpp Depends on



3.0 Overview of Findings

Upon completion of the review of the modules and functions identified, there was no malicious code found in the specific modules SLI reviewed that could cause the printer to print an already marked ballot. The search was then expanded to include files that included the key word “swath”, depended on the Swath.cpp file or was depended on by the Swath.cpp file. No malicious code was found in any of the source code modules reviewed as described above.

4.0 Conclusion

After conducting a manual and automated secure source code review of the Dominion Voting Systems (DVS) ImageCast Evolution (ICE), SLI Compliance found the DVS ICE 4.14.25 source code does not contain any malicious code in the specific modules reviewed.

NYSTEC Response #2

Reported Design Flaw in Dominion ImageCast Evolution (ICE) Voting Machine
NYSTEC Response #2

Background on Reported Design Flaw

During the March 19th, 2019 Commissioners' meeting, the NYS Elections board requested that NYSTEC consult with SLI and Dominion to determine the extent of the SLI security testing and code review and supplement the report (included below, as Appendix 1) previously provided to SBOE. The reported issue with the Dominion ImageCast Evolution is described in that earlier report. This update documents the additional analysis performed by NYSTEC, which considered the additional SLI code review and our updated risk ratings and mitigation control recommendations.

In this analysis, NYSTEC took the following steps:

- Met with Dominion on March 19th to observe the ICE system, understand several additional controls Dominion was recommending and discuss the issue with them. Observed how the system can prevent the issue through the possible addition of additional preventative controls including:
    - Removal of the ink cartridge
    - Leaving the printer access door open
    - Ability to use a foam block to prevent printing
- Participated in a call with SBOE, SLI and Dominion on March 28th to further determine what security testing was completed relevant to this specific issue and recommend additional testing to which SLI agreed.
- Reviewed the additional testing artifacts provided by SLI on March 27th and 29th, 2019.
- Provided SLI with additional questions to respond to, which they did.
- Reviewed the report "NYSBOE Dominion – Additional Review ImageCast Evolution" sent by SLI on 4/3/2019.
- Participated in a call with SLI, Dominion and SBOE on April 5th, 2019, where additional questions were asked and answered.
- Obtained input from Dominion on the implementation feasibility of the controls they recommended at the March 19th meeting.

Conclusion

NYSTEC's review was limited strictly to SLI code review efforts to verify that malicious code, specific to the ability to print voter marks on a cast ballot, was not present in the ICE source code. Our conclusion is based on our review of the materials provided, meetings with SLI and results of the additional testing performed as documented by SLI. The second round of automated and manual source code review completed by SLI verified that all calls to and dependent code for the module that can print votes (Swath.cpp) were valid and that none of

the related code contained malicious code. NYSTEC believes that SLI security testing of the Dominion source code provided reasonable assurance that malicious code that could be triggered to enable the machine to print additional marks on an already marked ballot, is not present in the version tested.

Threat Mitigation Recommended Steps

In addition to the recommendations from our prior report, we also recommend that following additional controls recommended by the vendor, be considered for adoption, where practical at the CBOE, for any combination BMD/Scanner. These would include but not be limited to:

Detective Controls

- Incorporate an audit of the hardware-based printer counter following each election through the use of a trusted COTS device that cannot be compromised by the attacker.
- Updated poll site procedures to instruct poll workers to be aware of the machine printer running when it should not be, as this takes 30-40 seconds during which time scanning cannot occur.

Preventative Controls

- Leave the printer access panel open as this will prevent an unauthorized ballot from being marked without detection.
- Remove the printer ink and only insert it when the system is being used in BMD mode.
- Insert a foam block inside the printer carriage, as this will prevent the system from ever printing on an already voted ballot.

Additional information provided by Dominion and verified by SBOE indicated that removal of the printer ink cartridge may not be feasible in all situations. This control requires the Technician Key and credentials, and feasibility of having this on hand would depend on the voting site. Also, the system would produce continual warnings that the printer is not ready. Dominion also expressed concerns over the use of a foam insert as it could cause damage should the print head ever attempt to move while the foam block is in place. These issues should be considered by CBOEs when implementing the respective preventative controls.

Dominion Updates to the TDP

NYSTEC reviewed the potential threats listed in the threat register identified in the Dominion TDP (2.06 – Democracy Suite System Security Specification 4.14E::436) and recommends that this list be updated to include a compromise of the system firmware/software, resulting in (amongst other possible outcomes) votes not being counted correctly or illicit marking of cast ballots. Documenting this threat scenario and the risk mitigating controls available would be valuable to NYS counties as well in response to the State Board.

Overall Risk

While NYSTEC has not conducted a full NIST 800-30 risk assessment of this particular threat we have updated our prior assessment to include the overall risk with new compensating controls in place.

Threat Scenario	Compensating Controls	Likelihood	Impact	Risk
Adversary succeeds in modifying ICE software/firmware with malicious code to alter marked ballots	Existing controls in place (not inclusive of any of the above listed)	Very Low	Very High	Low
Adversary succeeds in modifying ICE software/firmware malicious code to alter marked ballots	Existing controls and any of the above additional preventative controls	Very Low	Low*	Very Low

*In the second scenario, we considered the impact to be Low because there could still be a negative impact to public trust, as it is likely the event would be reported in the media, should the printer start, or a printer error be displayed during a voting session. We can also state that there is zero risk of altering the outcome of an election as the exploit cannot succeed when the access panel is open, print cartridge is removed, or the foam block is in place.

TABLE I-2: ASSESSMENT SCALE – LEVEL OF RISK (COMBINATION OF LIKELIHOOD AND IMPACT)

Likelihood (Threat Event Occurs and Results in Adverse Impact)	Level of Impact				
	Very Low	Low	Moderate	High	Very High
Very High	Very Low	Low	Moderate	High	Very High
High	Very Low	Low	Moderate	High	Very High
Moderate	Very Low	Low	Moderate	Moderate	High
Low	Very Low	Low	Low	Low	Moderate
Very Low	Very Low	Very Low	Very Low	Low	Low

Revised Threat Register from Dominion

- **Tampering with installed software**

  Description - The software installed on the PCOS devices is reviewed, built and tested by a Voting System Test Lab (VSTL). These Trusted Builds are installed on the PCOS devices and control their operation. A special set of credentials is required to install the software and integrity checks are performed during installation to ensure a valid build is being installed. Hash values are generated by the VSTL for both the installation files and the files on the PCOS device after installation. The hash values are recorded in a System ID Guide for jurisdictions to use to verify the integrity of the PCOS software.

  Threat - A malicious actor obtains unauthorized physical access to the PCOS devices after pre-election "logic and accuracy" testing but before Election Day, successfully defeating the physical controls that Election Administrators have in place. The installation software is counterfeited and fraudulent software is installed. The malicious actor also defeats the controls in place related to the hash codes which are verified on Election Day. Then, this malicious actor once again obtains unauthorized physical access to the PCOS devices after the Election, again defeating physical security practices in place, and installs the certified software after Election Day.

  Impact - By changing the software, the malicious actor makes the voting system inaccurate or inoperable.

  Impacted security pillars - Integrity and availability.

  Risk rating - Low.

  Mitigation - Implement proper processes (access control) for memory card handling and device storage. Verify the integrity of the installation software prior to and after installation. During points where the physical chain of custody of a device is unknown, verify the integrity of the installed software. Cryptographic and digital signing controls mitigate tampering with installation software. Tampering is evident to operators when verifying the software installed on the device. For more information, refer to Sections 4 and 5.5 of this document. Also, refer to the VSTL generated hash values.

3.2.3.3 Deterrent Security Controls - Risk Avoidance

The overall system architecture of the Democracy Suite platform is based on the fundamental principle that the deployed system is *exclusively* used for election project definition and management including pre-voting, voting and post-voting activities. Under this premise, the system is designed in such a way that it does not allow, nor require, any external processing or communication infrastructure in its base configuration (i.e. public telecommunication and data transmission technologies such as Internet or wireless networks are not allowed nor required). Using this approach, potential security threats are significantly reduced.

Nevertheless, internal threats remain and proper security controls and policies are imperative.

The Democracy Suite platform, and especially its EMS components, provide technological capabilities to implement proper security measures within the system. Some of the deterrent security controls are:

- Dedicated system infrastructure
- Clear security policies for all employees
- Physical security measures
- Premises monitoring measures

The Democracy Suite platform implements the following security controls as part of the deterrent security controls: See Sections 4 and 5 of this document.