

Software Review and Security Analysis of the Diebold Voting Machine Software

Premier TSx Version 4.7.1 Supplement

Kristine Amari Ryan Gardner Brian Hay
John Kerski Kara L. Nance Alec Yasinsac

with assistance from:

Connor Poske and Aaron Wilson

Security and Assurance in Information Technology Laboratory
Florida State University
Tallahassee, Florida
December 19, 2007

For the Florida Department of State

Software Review and Security Analysis of the Diebold Voting Machine Software Premier TSx Version 4.7.1 Supplement

Table of Contents

Section	Title	Page #
1	Executive Summary	3
2	Project Introduction and Background	4
3	Findings	6
4	Conclusions	12
5	Acknowledgments	14
6	References	14
Appendix A	Flaw List	15
Appendix B	TSx v. 4.7.0.3 Supplemental Report	18
Appendix C	Non-Pertinent Flaws – Coverity Errors	34
Appendix D	Other Non-Pertinent Flaws	42

Software Review and Security Analysis of the Diebold Voting Machine Software Premier TSx Version 4.7.1 Supplement

1 Executive Summary

On May 14th 2007, the Florida Department of State (FLDoS) commissioned an independent expert review of Diebold¹ Elections Systems touch screen (version 4.6.5) and optical scan voting system software by a team led by Florida State University's (FSU) Security and Assurance in Information Technology (SAIT) Laboratory [1]. The team issued a final report on July 27, 2007 [2] (hereafter referred to as the SAIT Diebold Report without citation). Shortly thereafter, the Florida Secretary of State (FLSoS) declined to certify the systems that employed the reviewed software [3].

Diebold Elections Systems submitted modifications to their touch screen software (version 4.7.0.3), which SAIT Laboratory reviewed and reported results in a supplemental report (see Appendix B) on September 28, 2007.

On November 6th, 2007 FLDoS asked SAIT Laboratory to conduct a second supplemental review for a third touch screen software revision (version 4.7.1). This report is the culmination of that second touch screen software supplemental review.

1.1 The Analysis' Scope

The scope of the investigation, as defined in the Statement of Work (SoW), is:

This review is for the purpose of yielding technological data to FLDoS to ensure voting system effectiveness and security in Florida elections by investigating for potential flaws in target software as documented in reported literature and other published studies [1].

The team constructed a flaw list from the SAIT Diebold Report and the California Top to Bottom Review Diebold report [4], and this list, given in Appendix A, drove the analysis. We did not conduct a comprehensive software review nor did we perform penetration testing, as each of these was outside the project scope. FLDoS provided the team a TSx device for this supplemental review. The Tallahassee Premier Election Solutions office provided ballot definition files and key cards that support voter, supervisor, and administrative actions.

We emphasize that our technical analysis reflects neither endorsement of, nor opposition to, certification. We present this technical data exclusively for consideration by FLDoS in their decision processes.

1.2 Systems Analyzed

We analyzed the AccuVote TSxTM touch screen firmware version 4.7.1. We considered flaws in previous firmware versions including those found in the firmware itself, the bootloader, and the AccuBasic interpreter.

1.3 Findings Summary

Our analysis identified software upgrades that included major improvements in the signature flaw, the bootloader process, and elimination of dangerous commands in the interpreter. The vulnerability associated with the unprotected "assure.ini" file is also largely mitigated. This version represents an incremental improvement relative to reported security flaws although many documented vulnerabilities remain. The system retains significant weaknesses in key

¹ Diebold Elections Systems subsequently changed its name to Premier Election Solutions.

management, removable media handling, communication procedures, and a myriad of fundamental, structural security weaknesses.

2 Project Introduction and Background

2.1 Report Organization

This document is the project report. It contains our pertinent findings and conclusions and the technical analysis that supports these conclusions. In accordance with the terms of the Statement of Work, we have avoided revealing proprietary information and we are careful to avoid revealing information that would describe how to attack an election. This report stands on its own and reflects the totality of our non-proprietary related findings regarding known flaws.

This document first gives background information about the known flaws that guided our analysis. We then describe our findings and conclusions.

2.2 The Software Review Team

Kristine Amari is a computer scientist with the Defense Information Systems Agency. She is an expert in information security with experience conducting red team analysis for critical information systems.

Ryan Gardner is a doctoral student in the Computer Science Department at Johns Hopkins University and is a member of the ACCURATE center.

Brian Hay is the Director of the Advanced Systems Security Education, Research, and Training (ASSERT) Lab at the University of Alaska Fairbanks.

John Kerski is a computer science graduate student and a member of SAIT Laboratory at Florida State University.

Kara Nance is a Chair of the Department of Computer Science at the University of Alaska Fairbanks and the Director of the ASSERT Center.

Connor Poske is an undergraduate computer science student at Florida State University and is a member of SAIT Laboratory.

Aaron Wilson is an undergraduate computer engineering student at Florida State University and is an intern in the Florida Division of Elections.

Alec Yasinsac is a co-Director of SAIT Laboratory, an Associate Professor of Computer Science at Florida State University, and is the lead Principal Investigator on this project.

2.3 The Investigative Process

As a supplemental review, the terms of the SAIT Diebold Report also apply to this document, including all definitions, disclaimers, and findings that are not specifically addressed herein. The target system is the Premier Election Solutions (formerly Diebold Election Systems) touch screen software that is presently submitted to the Florida Department of State Bureau of Voting Systems Certification. Specifically, we were provided the AccuVote TSx™ version 4.7.1 software and the earlier versions (4.6.5 and 4.7.0.3) for comparative purposes.

As the first objective, the team constructed a flaw list that is given in Appendix A. The items were comprised from the SAIT Diebold Report and the California Top to Bottom Review (TTBR) Diebold Analysis [4]. The resulting flaw list drove our analysis.

Our review proceeded in three parts, with the main component being examination of each issue identified in the SAIT Diebold Report. This review component leveraged code comparison

against the earlier version and we were guided and greatly assisted by the comments database from the original review and the first supplemental review.

Our second analysis emphasis involved new investigation of flaws identified in the California Top To Bottom Review [4] that we did not investigate in the original SAIT Diebold review. Finally, we conducted a broad analysis to identify any injected structural regression flaws. For this, we utilized an automated static code analysis tool.

We did *not* conduct penetration or red team testing for these systems, as that was outside our charter. We did construct exploits for identified flaws where we were able, as we did in the original SAIT Diebold review.

2.4 Limits of this Study

2.4.1 Laboratory Results

Unfortunately, there is no existing procedure for testing voting system security exploits in a non-laboratory setting. Voting systems cannot be subjected to dangerous security analysis while in operation so while laboratory analysis presents a clinical perspective, it gives the only reliable information regarding their operational security properties.

The SAIT Diebold Report combined software review with hands-on testing, exercising the hardware to verify results extracted from the software. The report listed operational attack prerequisites and mitigations based on this comprehensive analysis. This review leveraged previous analysis to extend its results and we refer to applicable mitigations and prerequisites where they still apply.

2.4.2 Completeness Limitations

It is well-known that testing is an inherently incomplete process and that no testing process can guarantee absence of flaws. Regarding even the isolated flaws that we evaluated, we cannot guarantee that the changes or repairs that removed, mitigated, or reduced these flaws did not cause regression faults. While we did look for regression faults, it is possible that undetected regression faults may be more dangerous than the flaws that were fixed.

2.4.3 Limitations of Scope

This project's purpose is to evaluate the target system's technical properties with respect to our list of previously-identified flaws. Our analysis is not comprehensive. We emphasize that our results do not extend beyond the scope of our investigation. We do not contend that these systems are correct or secure beyond the specific results given here. A significant amount of new code was introduced which we did not rigorously review for additional security flaws. As a supplemental review, our exclusive purpose is to provide a technical assessment regarding only those flaws identified in earlier reviews. We defer interpretation and application of these technical results to FLDoS.

2.5 Diebold Software Architecture

2.5.1 Code Structure

We reiterate earlier findings that the AccuVote TSx™ code was not developed for high assurance. The code style and readability vary significantly across the code base. There is substantial complicated code throughout the system and there is poor internal program documentation, such as missing, misleading, and weak comments (#23). There are evident deviations from EAC/FEC standards including unwieldy modules, modules with multiple

entry/exit points, modules with misleading comments, etc. These, and other observed deviations from traditional software engineering practice, result in code systems that are difficult to analyze. In particular, the lack of high assurance directed structure in the large, monolithic code base significantly impairs analysts' ability to confidently make positive statements regarding the system's security.

The addition of internal documentation that meets the specifications for automated documentation generation has significantly improved the readability of the individual modules. However, detailed design documentation was not available in the basic review or in this supplemental effort so the compliance of the documentation with FEC design standards could not be determined.

2.5.2 Off-The-Shelf Software

As with most modern applications, there are several third-party components to the reviewed voting systems. We did not independently investigate the third-party operating system, associated database system, or driver software.

3 Findings

We group our findings to parallel the SAIT Diebold Report structure. We re-reviewed many items reported as fixed (e.g., #18, 50) or benign (e.g., #37, 38) in the SAIT Diebold Report to determine if regression faults may have been injected in this version. We note the two such faults that we detected in section 3.3.10 and section 3.3.14 below. In addition, we have reconsidered some of the items based on additional information obtained through access to the physical machine (e.g., #33).

Original Flaw Identification	No Change from May	Improved/ Attempted	Fixed	Regression Errors
SAIT Diebold Report	7, 8, 9, 10, 11, 13, 14, 15, 17, 18, 20, 21, 22, 24, 26, 27, 28, 30, 31, 37, 38, 39, 55	16, 17, 23, 32	1, 2, 3, 4, 5, 6, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50	12, 16, 19, 25, 34, 35
SAIT Diebold Review		52	51	
California TTBR Report [REF1]	60	53, 58	56, 57, 61	

Table 1: Flaw Status by Flaw Number (See Appendix A)

3.1 Overview

Though there are many changes and some fundamental upgrades in the present software version, it represents only a slight security improvement overall. The interpreter code (section 3.5.1) in this version is cleared of known dangerous commands (e.g., `strcpy`), which constitutes an important safety improvement. We also note that the cryptographic use is expanding in the code base and accepted standards are beginning to be applied. Major improvements are evident in the bootloader code (see section 3.4.1), where revised verification processes and the repaired signature algorithm solidify software startup procedures.

Nonetheless, the code base is highly unstructured, apparently resulting from the institutional development procedure. Many cryptographic practices remain immature with naïve key management, and other weak applied-encryption practices permeate the application. Moreover, we identified a number of previously unreported buffer overflow flaws in the bootloader code that place the application at risk.

We reiterate that the purpose of this report is to evaluate the *technical properties* of the current systems with respect to previously-identified flaws. Its purpose is neither to condemn nor endorse these systems; rather the findings herein should be carefully considered in the context of the overall election process.

3.2 The Signature Flaw (#51)

One of the most important flaws identified in the SAIT Diebold Report concerned the code's faulty RSA signature verification implementation. The hand-coded RSA signature verification failed to examine the high order bits of the decoded signature allowing signatures to be forged as we implemented in the SAIT Diebold Review.

In the revision, the vendor corrected the signature verification by modifying the code to examine the entire decoded signature at verification. The signatures also now incorporate padding in RSASSA-PKCS1-v1-5 format as per RFC 3447 into the signature process which the code appropriately verifies. This prevents forgeries that we previously identified.

3.3 AV-TSx Firmware Faults

3.3.1 Cryptographic Key Management (#15)

The SAIT Diebold Report identified issues with the foundational symmetric keys utilized to protect TSx data. Two 128-bit AES keys, the data encryption key and the system key, are used for all authentication and encryption, including encrypting and generating the message authentication codes for the stored electronic ballots, generating the message authentication codes for the election database file, and keying the hash that stores the supervisor PIN.

The present version does not address these issues. Thus these weaknesses and the attack prerequisites and mitigations identified in the SAIT Diebold Report paragraph 3.7.1.1 still apply.

3.3.2 Memory Card Update File is Unprotected (#32, 52)

The SAIT Diebold Report noted that the “`assure.ini`” file onboard the TSx memory card was unencrypted and unauthenticated and was subject to malicious manipulation. An attacker who could remove the card from the voting terminal could modify the file to set the machine into pre-election mode and could create valid voter cards while the terminal was in this mode.

The present version does not encrypt the `assure.ini` file. However the file contains a Message Authentication Code (MAC) computed from the HMAC algorithm using the data encryption key

and with SHA-1 as the underlying hash. The MAC slightly mitigates the attack vector for this vulnerability. However, the authentication still has several weaknesses.

The assure.ini file is subject to replay attack if an adversary can copy files from a PCMCIA card during pre-election mode. Specifically, the adversary could replace the contents of the file with an older version from pre-election mode, and thus put the voting machine into pre-election mode. As stated in the SAIT Diebold Report, an attacker can create valid voter cards while the terminal is in this mode. The review team was able to successfully exploit this vulnerability on a single machine.

Furthermore, the authentication provided by the MAC relies on the secrecy of the data encryption key. Thus its benefit is largely obviated by the key management limitations described in section 3.3.1. SHA-1 is also a deprecated hash function as researchers have found collisions in it, and an attacker may be able to create two files that authenticate under the same MAC. As has been reiterated, using the same cryptographic key (the data encryption key) for more than one purpose is considered poor practice by cryptographers.

The present version has also added a synchronization function between files on the primary storage (the PCMCIA card) and the backup storage (located within the machine) when a PCMCIA card is inserted into the machine. If the files on the primary storage device are not equivalent to the same files on the secondary storage device, the files on the primary storage will be overwritten with the files from the secondary. If the files on the primary storage device are not found on the secondary storage, the files on the primary storage are trusted and copied to the secondary storage device.

The addition of this functionality creates a vulnerability. If the data encryption key is the same across more than one machine in the same voting area, there is the potential for ballot stuffing. By copying the contents of a PCMCIA card from one machine to another during an election, an attacker could cause some of the ballots stored on several voting machines to reflect copies from other machines.

Potential Mitigations: The replay attack and ballot copying vulnerabilities may be mitigated by providing a nonce or timestamp in the calculation of the MAC on the assure.ini file and maintaining expected, corresponding nonces internally. Election officials can also ensure that all machines use a different encryption key. Authenticating the file with a public key signature could avoid most of the key management issues. The code should also use a non-deprecated hash function such as SHA-256.

3.3.3 Smart Card Authentication Issues (#7, 39)

While the smart card passwords can be set and modified by election workers, the implemented authentication protocol is not secure and allows an attacker to potentially create counterfeit, validating smart cards, including voter cards. This flaw remains essentially unchanged from the SAIT Diebold Report paragraph 3.7.1.3 along with the prerequisites and mitigations noted there.

3.3.4 Supervisor PIN is Not Cryptographically Protected (#8, 26, 28)

This flaw remains essentially unchanged although the default password is now longer than the one described in the SAIT Diebold Report paragraph 3.7.1.4. The attack prerequisites and mitigations noted there remain valid in this code version.

3.3.5 Insecure Storage Mount (#9)

This flaw remains essentially as described in the SAIT Diebold Report paragraph 3.7.1.5. The attack prerequisites and mitigations noted there remain valid in this code version.

3.3.6 System Configuration Information is Unprotected (#10)

This flaw remains essentially unchanged from the SAIT Diebold Report paragraph 3.7.1.6. None of the examples identified there were changed and many other instances not specifically noted were visible as well. The attack prerequisites and mitigations remain valid in this code version.

3.3.7 Protective Counter Stored in a Mutable File (#11)

This flaw remains essentially unchanged from the SAIT Diebold Report paragraph 3.7.1.7. The attack prerequisites and mitigations noted there remain valid in this code version.

3.3.8 Ballot Definition File is Unprotected (#12)

The code has changed with respect to the authentication of the ballot definition file (SAIT Diebold Report paragraph 3.7.1.8) and introduces a regression flaw.

The code shows signs of attempted improvements where the Message Authentication Code (MAC) is verified by the HMAC algorithm to verify the file in place of the previous ad-hoc approach. Furthermore, SHA-1 is used as the MAC's underlying hash function rather than MD5. The MAC is still keyed by the data encryption key. Despite these indications of improvement, the code now completely ignores the result of the attempted authentication of the file. Thus, an attacker who can write to the PCMCIA card could replace the ballot definition file with a custom version of her own, e.g. with two candidate names switched to effectively swap their votes.

If the code did not ignore the authentication result, it would remain subject to the significant limitations in the authentication also described in section 3.3.6. Namely, the function of the MAC would be significantly obviated by poor key management (section 3.3.1) since it inevitably relies on the secrecy of the key, and the use of the deprecated SHA-1 hash function may enable an adversary to create two ballot definition files that authenticate under the same MAC. The repeated use of the data encryption key for many purposes is also again considered poor practice.

Potential Mitigations: The file should be authenticated and verified with a proven public key signature algorithm.

3.3.9 Impersonate a TSx Terminal to GEMS (#14, 25, 35)

In the present version, use of SSL is still optional and determined by a value read in from the registry. Root certificates and root public keys are hard coded. The machine's private key is stored in a local file. As described in the SAIT Diebold Report section 3.10.3, the client does not examine received server certificates after verifying that they have been signed by the certificate authority. Specifically, the client does not verify that the owner of the received certificate is the expected server. It also seems likely, given the lack of certificate examination, that all clients use the same public/private key pair.

Based on this analysis, we believe that the risk and attack prerequisites and mitigations noted in paragraph 3.7.1.9 of the SAIT Diebold report apply to the present version as well.

3.3.10 No Integrity Protection of Stored Electronic Ballots (#17)

The authentication of the stored electronic ballots, previously described in SAIT Diebold report paragraph 3.7.1.10, has changed as in sections 3.3.6 and 3.3.8. Each ballot is appended with a Message Authentication Code (MAC) now using the HMAC algorithm with SHA-1 and keyed by the data encryption key. Although the change in MAC and hash algorithms represent some improvement, the effectiveness of the authentication is still limited by the key management (section 3.3.1) and continued use of deprecated hash functions as described in sections 3.3.6 and 3.3.8.

The attack prerequisites and mitigations noted in SAIT Diebold Report paragraph 3.7.1.10 remain valid in this code version.

3.3.11 Ballots are Stored Sequentially (#20, 21)

This flaw remains essentially unchanged from the SAIT Diebold Report paragraph 3.7.1.11. The attack prerequisites and mitigations noted there remain valid in this code version.

3.3.12 Candidate Information is Not Stored in the Results File (#13)

This flaw remains essentially unchanged from the SAIT Diebold Report paragraph 3.7.1.12. The attack prerequisites and mitigations noted there remain valid in this code version.

3.3.13 Audit Logs are Not Cryptographically Protected (#28)

This flaw remains essentially unchanged from the SAIT Diebold Report paragraph 3.7.1.13. The logs are encrypted and authenticated the same way as the electronic ballots, using the “system key”, which is insecure. Hence, they are also susceptible to viewing and modification by an adversary. The attack prerequisites and mitigations noted there remain valid in this code version.

3.3.14 Data is Neither Authenticated Nor Encrypted Over the Communication Link (#16, 19, 25, 34, 35)

The SAIT Diebold Report noted that the vendor mitigated this vulnerability by implementing SSL for this communication, but further noted that there were weaknesses in the vendor’s SSL implementation, largely relating to the entropy level for generated seeds. In the present version, the vendor introduced further adjustments to this code that include a possible regression fault.

The injected fault is that the changes omit error handling for the random number generation, allowing SSL to potentially execute with important values, such as encryption or signature keys, that could be predicted. No notification or retry would occur in such a case. While we are unable to precisely determine the impact without a full test environment, this practice allows unpredictable results at best, and possibly injects unnecessary security vulnerability.

The use of random numbers without sufficient entropy also seems probable since the code uses an operating system call to obtain its entropy and system calls generally provide “randomness” by collecting information from external events. Because SSL is initialized very early after system initialization, the operating system may not have observed a sufficient number of such events to produce strong entropy. Rigorous testing is necessary to determine the impact of this potential flaw.

The use of the system call is, however, generally considered to be a good practice so long as sufficient error checking is conducted.

The certificate concerns discussed in section 3.3.9 above are also relevant here.

Potential Mitigation: The code should be modified to generate strong entropy or to fail safely with meaningful error reporting if operating conditions preclude proper operation.

3.3.15 Several TCP/UDP Ports Are Open (#29)

We did not have access to an appropriate PCMCIA Ethernet NIC during this test, and as such were unable to determine whether the TSx unit still exhibits this behavior.

3.3.16 A Local User Can Access the Main Menu/System Setup Menu Without a Smart Card or Key (#55)

This issue, identified in the California TTBR [4], remains an issue with the current software version.

3.3.17 A Malicious Election Resource File on the Memory Card Could Exploit Multiple Vulnerabilities to Run Arbitrary Code (#57, 58)

California TTBR [4] reported two vulnerabilities in the code that processes the election resource file. The code that reads the page number from the RTF format file now conducts length checks before the data is stored it into a buffer, preventing the noted vulnerability.

However, when the resource data for displaying instructions is in bitmap format, the code trusts the read file size when exercising copy operations. The resource data is read directly from the election definition on the card. This vulnerability remains in the present firmware version.

3.3.18 Files on the Voting Machine are Not Securely Erased When They are Deleted (#60)

This issue, identified in the California TTBR [4], remains an issue with the current software version. Our analysis indicates that this weakness may allow an attacker to circumvent the authentication process implemented to protect the integrity of the assure.ini file with a replay attack (see section 3.3.2). Since files are not securely deleted, the attacker may capture previous versions of the assure.ini, database, and resource files, and then copy those files to a memory card. If the memory card is properly setup with the older files and inserted into the machine, the machine will reset the terminal to pre-election mode where voter cards can be created and initialized.

Potential Mitigation: This vulnerability may be mitigated by election procedure. All removable media should be securely wiped clean between elections. If older files are stored elsewhere, such as on an election center computer, officials should ensure that these files are protected from unauthorized access. Finally, elections officials should ensure that new encryption keys are used for every election. The vendor should modify the code to securely erase all data it deletes.

3.3.19 AV-TSx Startup Code Contains Blatant Errors (#61)

This issue, identified in the California TTBR [4], is not present in the current software version. The identified module no longer exists in the code base and the offending code snippet does not exist in the current version.

3.4 AV-TSx Bootloader Faults

3.4.1 Bootloader Issues (#1, 2, 3, 4, 5, 6, 53)

The bootloader has changed significantly in this version. The booting process now first attempts to start the machine from a locally stored bootable image prior to conducting any other actions. If the initial attempt to boot fails, the bootloader then attempts to use any .ins files found on the removable memory card to install a new operating system and voting software image to the machine's permanent storage. The code now executes a standard signature verification (see section 3.2) over the new image prior to installation to verify its authenticity. This is a significant improvement, which will largely prevent the spread of voting machine viruses that propagate by installing malicious software directly through the bootloader's installation feature.

Finally, the bootloader relies on several foundational libraries and we were not provided. Without a complete code base, we were unable to run comprehensive static or dynamic analysis tools, though we were only able to run limited static analysis tools on the bootloader code. Even

this limited automated scan produced approximately thirty reported buffer overflow flaws. As is required in effective automated static code analysis, we verified that the reported buffer overflow flaws and report our findings in the private Appendix C as required in the Statement of Work. We stopped short of constructing exploits for these confirmed flaws, as this was outside the scope of this review.

3.5 AccuBasic Interpreter Issues

3.5.1 Unchecked String Operations (#40, 42, 43, 44, 45, 46, 47, 48, 56)

All instances of `strcpy`, `strncpy`, `memcpy`, and `printf` functions have been removed from the AccuBasic interpreter. They have been replaced with safe functions such as `strcpy_s` and `strncpy_s` that not only prevent the copy from overflowing, but also null-terminate the referenced strings. This is a notable improvement that mitigates an important class of security flaws.

3.6 Other Non-pertinent Flaws

During our review, we uncovered several flaws that were previously unreported, or were previously unknown to us. We documented these flaws in confidential Appendix D as is required in the Statement of Work.

4 Conclusions

Electronic voting systems are mission critical, testing-limited systems that demand high assurance engineering. This report documents the TSx software supplemental review supporting the Florida Department of State voting system certification effort for this software version. Our analysis focused on issues identified in the SAIT Diebold Report, but also considered issues identified during the SAIT Diebold Supplementary Review and the recent California Top To Bottom [voting system] Review.

Our analysis identified software upgrades that included major improvements in the signature flaw, the bootloader process, and elimination of dangerous commands in the interpreter. The vulnerability associated with the unprotected “`assure.ini`” file is also partially mitigated.

We conclude with the following summarizing issues.

4.1 Cryptographic Key Management

Cryptographic tools can provide many strong security properties that are demanded by electronic voting systems, but only when founded on strong key management practices. Subtle issues such as use of low entropy, weak key generation, or mild key reuse may seem harmless even to astute observers, but such issues can dramatically weaken the mechanisms they support or even render them useless.

Many of the critical cryptographic flaws that we identified in the original report result from naïve cryptographic misuse or seemingly mild deviations from standards. For critical applications, precisely following established cryptographic standards is paramount. Any customizations, variations or optimizations, no matter how mild, have potential to negate all security properties that the fully implemented standard provides.

When unusual situations not covered by standard operations arise, rigorous analysis by skilled cryptography experts is necessary. Secure applications demand rigor well beyond non-sensitive applications without exception.

4.2 Removable Media

The voting process is protected, not only by security measures in the machines, but also by the

formal process developed to govern the voting process. It is important that processes to minimize risk impact are in place and rigidly enforced. The machines and all removable media associated with electronic voting systems are highly sensitive and must be protected year-round with the same status and priority as voted ballots. Unsupervised access to an item as simple as a voter card may allow an individual to create her own valid cards. In the wrong hands, these cards can cause significant damage to a county's vote count validity.

Moreover, software associated with data processed from removable media must follow rigorous defense in depth principles since data stored on them is the most vulnerable to malicious tampering. Developers must rigorously follow safe practices for storage, computation, and garbage collection for these perimeter issues.

4.3 Issues to Improve Voting System Security Evaluation

As SAIT Laboratory's fifth voting system code review, several helpful rules of thumb have emerged and been validated through the review process. We offer these insights for future review consideration.

4.3.1 Precise Review Objectives

It is a well known cliché that “If you don't know where you are going, any road will take you there”. Without a precise objective, it is impossible to establish an efficient process or to measure analysis completeness. While open objective reviews to date have provided valuable contributions, their main result has been to identify disqualifying flaws.

Studies that result in system certification demand establishing and rigidly following precise requirements that can provide elections officials with actionable information that allows them to balance competing election system demands to conduct safe, accurate elections.

4.3.2 Team Spans Diverse Skills

Effective voting system security review requires broad and deep knowledge and skills. While mainstream computer scientists offer a firm foundation for these reviews, the analysis quality will benefit from members with specialty software engineering, testing, systems administration, cryptography, red teaming, and project management skills.

4.3.3 Structured Note Taking

Again, as voting system software security analysis becomes the norm, iteration is inevitable as we see significant overlap among the SAIT Reports, the California Top To Bottom Review, and the Ohio EVEREST review, for example. Rigorous documentation processes provide tremendous benefit in both coalescing immediate results and for regression analysis. The two SAIT Diebold supplemental reviews proved the value of rigorous, structured documentation. Teams should establish a sound documentation plan and relentlessly stick to it. It is essential that the notes database be retained by the convening authority as work product and provide this database for subsequent review efforts.

4.3.4 Complete Development Environment

Software review and proof of concept testing are resource intensive processes that demand domain-appropriate skills. Conversely, software development environments can vary greatly. In order to reduce both spin up and proof of concept construction time in the review process, states should require vendors to file complete development environments with their systems. The environment delivered should enable the state and any review teams to rebuild the certified binaries. During a review, these should be delivered with the source code to the review team. If

the environments are not available through the state, supporting tools and libraries should be provided to the team by the vendor.

4.3.5 Complete Design Operating Documentation

Similar to having access to complete development environments, software review can be simplified if accurate, detailed design and operating documentation is available. States should require such documentation from the vendor and make it available to any software analysis effort. If such documentation is not available from the state, the vendor should supply it directly to the review team.

5 Acknowledgments

As part of our work we used the automated source code analysis tool Coverity Prevent SQS to assist with the code review process. We note that Coverity is a SAIT Laboratory partner.

6 References

- 1 “Software Review and Security Analysis for Diebold Voting Machine Software.”, Joint Florida Department of State and Florida State University Statement of Work, May 14, 2007
- 2 Ryan Gardner, Alec Yasinsac, Matt Bishop, Tadayoshi, Kohno, Zachary Hartley, John Kerski, David Gainey, Ryan Walega, Evan Hollander, and Michael Gerke, “Software Review and Security Analysis of the Diebold Voting Machine Software”, Final Report For the Florida Department of State, July 27, 2007, <http://election.dos.state.fl.us/pdf/SAITreport.pdf>
- 3 Kurt Browning, Florida Secretary of State Letter to Mr. David Byrd, Diebold Election Systems, dated July 31, 2007, <http://election.dos.state.fl.us/pdf/SAITbrowningLetter.pdf>
- 4 Joseph A. Calandrino, Ariel J. Feldman, J. Alex Halderman, David Wagner, Harlan Yu, William P. Zeller, "Source Code Review of the Diebold Voting System", July 20, 2007, For the California Secretary of State “Top-to-Bottom” Electronic Voting System Review

Appendix A Flaw List

ID	Old ID	Description	Source	Page #
1	3	fboot.nb0 can be used to load malicious software	[2]	18
2	4	fboot.nb0 can be overwritten without authentication or integrity checks	[2]	18
3	5	Denial of Service with PowerOffSystem() API	[2]	16
4	6	Bootloader replaces itself with eboot.nb0 or fboot.nb0 if found on memory card on boot, no authentication	[2]	18
5	7	Bootloader replaces OS with nk.bin (or some other nk.xxx's) if found on memory card, no authentication	[2]	19
6	8	Bootloader "runs" any .ins files in memory card on boot after prompting user, no authentication	[2]	19
7	13	Lack of smartcard authentication	[2]	14
8	14	PIN sent from smartcard to terminal in cleartext	[2]	14
9	15	Insecure file mount, does not ensure writing to removable media	[2]	15
10	16	Unprotected system configuration file	[2]	15
11	17	Protective counter stored in mutable file	[2]	16
12	18	Ballot definition unprotected and unauthenticated	[2]	16
13	19	Candidate information is not stored in the results file	[2]	18
14	20	Impersonate a legitimate voting terminal, no authentication and data can be gathered from ballot definition files	[2]	17
15	21	Cryptographic key management	[2]	12
16	22	DES CBC encryption uses constant 0 for IV	[2]	17
17	23	No integrity protection of stored vote counts and data	[2]	17
18	24	No sequence numbers stored with votes (to help detect record deletion, may be problematic with respect to privacy)	[2]	15
19	25	No integrity checks, authentication, or encryption on votes transferred to the backend server	[2]	18
20	26	Votes are written serially	[2]	17
21	27	Bad RNG for randomizing vote order	[2]	17
22	28	Audit log problems: no consistency for what is logged, does not verify that printer is attached before writing to it	[2]	18
23	29	Complicated, undocumented code segment	[2]	18
24	34	DES encryption key is hard-coded	[2]	26

25	35	Data is not encrypted when transmitted over data link	[2]	18
26	36	The supervisor's password is hard-coded	[2]	26
27	37	Not applicable to this analysis	[2]	26
28	38	PIN for all smart cards is the same and the factory default	[2]	26
29	39	Several TCP/UDP ports are open		
30	40	Can make counterfeit voter and supervisor cards	[2]	14
31	41	Can vote multiple times with a card writer	[2]	14
32	42	PCMCIA cards are not encrypted	[2]	13
33	43	Not Applicable to this analysis	[2]	24
34	45	Digital certificates only exist at the servers and these are neither signed nor authenticated by the AccuVote terminals	[2]	18
35	46	No GEMs authentication by the AccuVote-TS terminals	[2]	18
36	63	Not applicable for this analysis.		
37	67	Error checking is inadequate for identifying and recovering from failures in a damaged or disfunctional environment	[2]	23
38	68	Error handling makes it difficult to differentiate between malicious activity and failure	[2]	23
39	69	The contents of the smartcards are neither encrypted nor digitally signed	[2]	23
40	70	W1 Array bounds violation: Overwrite any memory address with a 4-byte value that the adversary has partial control over. Allows attacker to inject malicious code	[2]	26
41	71	W3 Input validation error: Choose any memory location and begin executing it as .abo code; could be used to conceal malicious .abo code in unexpected locations, or to crash the machine	[2]	26
42	72	W6 Array bounds violation: Overwrite any memory location with any desired value. Allows attacker to inject malicious code	[2]	26
43	73	W7 Buffer overrun: Corrupt memory, crash the machine	[2]	26
44	74	W8 Buffer overrun, integer conversion bug: Corrupt memory until the machine crashes	[2]	26
45	75	W10 Buffer overrun: Overwrite return address on the stack. Allows attacker to inject malicious code and take complete control of the machine	[2]	26
46	76	W11 Array bounds violation: Information disclosure: read from potentially any memory address. Crash the machine	[2]	26
47	77	W12 Array bounds violation: Writes any 4-byte value to any address. Allows attacker to inject malicious code	[2]	26
48	78	W13 Array bounds violation: Information disclosure: read a 4-byte value from any address	[2]	26
49	79	W14 Pointer arithmetic error may crash machine. Could begin interpreting random memory locations as though they were .abo code	[2]	26

50	80	Three types of tokens used to potentially read and modify data in global memory. (no specifics given)	[2]	26
Flaws Identified During the SAIT Diebold Review				
51		Signature Flaw	[2]	10
52		Memory card file Assure.ini is not protected	[2]	13
Flaws Identified During the California Top To Bottom Review				
53		Multiple buffer overflows in .ins file handling allow arbitrary code execution on startup.	[4]	41
54		Duplicate of #7	[4]	42
55		A local user can get to the Main Menu/System Setup menu without a smart card or key.	[4]	46
56		Multiple vulnerabilities in the AccuBasic interpreter allow arbitrary code execution.	[4]	47
57		A malicious election resource file on the memory card could exploit multiple vulnerabilities to run arbitrary code	[4]	48
58		Malicious election database files can cause arbitrary code execution on the AV-TSx when uploading elections to GEMS.	[4]	48
59		Duplicate of #20	[4]	49
60		Files on the voting machine are not securely erased when they are deleted.	[4]	51
61		AV-TSx startup code contains blatant errors.	[4]	51

Appendix B TSx Supplementary Report of Sept 28, 2007

Software Review and Security Analysis of the Diebold Voting Machine Software

TSX Supplement

Alec Yasinsac John Kerski David Gainey
Michael Gerke Kristine Amari Donnie Newell

Security and Assurance in Information Technology Laboratory

Florida State University

Tallahassee, Florida

September 28, 2007

Final Report
For the Florida Department of State

Software Review and Security Analysis of the Diebold Voting Machine Software, TSX Supplement

Table of Contents

Section	Title	Page #
1	Executive Summary	3
2	Project Introduction and Background	4
3	Findings	6
4	Conclusions	10
5	Acknowledgments	11
6	References	12
Appendix A	Flaw List	13

Software Review and Security Analysis of the Diebold Voting Machine Software TSX Supplement

1 Executive Summary

On May 14th 2007, the Florida Department of State (FLDoS) commissioned an independent expert review of Diebold¹ Elections Systems touch screen and optical scan voting system software by a team led by Florida State University's (FSU) Security and Assurance in Information Technology (SAIT) Laboratory [1]. The team issued its final report on July 27, 2007 [2] (hereafter referred to as the SAIT Diebold Report without citation). Shortly thereafter, the Florida Secretary of State (FLSoS) declined to certify the systems that employed the reviewed software [3]. Subsequently, Diebold Elections Systems submitted modifications to their optical scan software, which SAIT Laboratory reviewed and reported results in a supplemental report [4] on August 10, 2007.

On August 29th, FLDoS asked SAIT Laboratory to conduct a second supplemental review, this time for the touch screen software revision that the vendor submitted subsequent to the FLSoS letter [3]. This report is the culmination of the touch screen software supplemental review.

1.1 The Analysis' Scope

The scope of the investigation, as defined in the Statement of Work (SoW), is:

This review is for the purpose of yielding technological data to FLDoS to ensure voting system effectiveness and security in Florida elections by investigating for potential flaws in target software as documented in reported literature and other published studies [1].

The team constructed a flaw list from the SAIT Diebold Report and the California Top To Bottom Review Diebold report [5] and this list, given in Appendix A, drove the analysis. We did not conduct a comprehensive software review nor penetration testing, as each of these was outside the project scope. Equipment was not provided to the team for this supplemental review.

We emphasize that our technical analysis reflects neither endorsement of, nor opposition to, certification. We present this technical data for consideration by FLDoS in their decision processes.

1.2 Systems Analyzed

We analyzed the AccuVote TSxTM touch screen firmware version 4.7.0.3. We considered flaws in previous firmware versions including those found in the firmware itself, the bootloader, and the AccuBasic interpreter.

1.3 Findings Summary

Our analysis identified software upgrades that included major improvements in the signature flaw, the bootloader process, and elimination of dangerous commands in the interpreter. The vulnerability associated with the unprotected "assure.ini" file is also largely mitigated. This version represents a clear, though incremental improvement relative to common security flaws.

Still, many documented vulnerabilities remain. The system retains significant weaknesses in key management, removable media handling, and communication procedures and a myriad of fundamental, structural security weaknesses.

¹ Diebold Elections Systems subsequently changed its name to Premier Election Solutions.

2 Project Introduction and Background

2.1 Report Organization

This document is the project report. It contains our pertinent findings and conclusions and the technical analysis that supports these conclusions. In accordance with the terms of the Statement of Work, we have avoided revealing proprietary information and we are careful to avoid revealing information that would describe how to attack an election. This report stands on its own and reflects the totality of our findings regarding known flaws.

This document first gives background information about the known flaws that guided our analysis. We then describe our findings and conclusions.

2.2 The Software Review Team

Alec Yasinsac is an Associate Professor of Computer Science at Florida State University, a co-Director of SAIT Laboratory, and is the lead Principal Investigator on this project.

Kristine Amari is a Computer Scientist with the Defense Information Systems Agency. She is an expert in information security with experience conducting red team analysis for critical information systems.

David Gainey is a computer science graduate student, a member of SAIT Laboratory, and is a member of the technical staff at the Florida State University Office of Technology Integration.

Ryan Gardner is a doctoral student in the computer science department at Johns Hopkins University and a member of the ACCURATE center for voting systems research.

Michael Gerke is a computer science graduate student and a member of SAIT Laboratory at Florida State University and recently served as a technology specialist intern at the Florida Department of State, Division of Elections.

John Kerski is a computer science graduate student and a member of SAIT Laboratory at Florida State University.

Donnie Newell is an undergraduate computer science student at Florida State University and is a member of SAIT Laboratory.

2.3 The Investigative Process

As a supplemental review, the basic terms of the SAIT Diebold Report also apply to this document, including all definitions, disclaimers, and findings that are not specifically addressed herein. The target system is the Premier Election System (formerly Diebold Election Systems) touch screen software that is presently submitted to the Florida Department of State Bureau of Voting Systems Certification. Specifically, we were provided the AccuVote TSX™ version 4.7.0.3 software and the earlier version (4.6.5) for comparative purposes.

Unlike our earlier supplemental review of the Diebold optical scan software [4], the presently submitted TSx software revision introduced extensive additions, modifications, and deletions. As the first objective, the team constructed a flaw list that is given in Appendix A. The items were comprised from the SAIT Diebold Report and the California Top To Bottom Review (TTBR) Diebold Analysis [5]. The resulting flaw list drove our analysis.

Our review proceeded in three parts, with the main component being examination of each issue identified in the SAIT Diebold Report. This review component leveraged code comparison against the earlier version and we were guided and greatly assisted by the comments database from the original review. Our second analysis emphasis involved new investigation of flaws

identified in the California Top To Bottom Review [5] that we did not investigate in the original SAIT Diebold review. Finally, we conducted a broad analysis to identify any injected structural regression flaws.

We did *not* conduct penetration or red team testing for these systems, as that was outside our charter. No equipment was provided so we were unable to construct exploits for identified flaws as we did in the original SAIT review.

2.4 Limits of this Study

2.4.1 Laboratory Results

Unfortunately, there is no existing procedure for testing voting system security exploits in a non-laboratory setting. Voting systems cannot be subjected to dangerous security analysis while in operation so while laboratory analysis presents a clinical perspective, it gives the only reliable information regarding their operational security properties.

The SAIT Diebold Report combined software review with hands on testing, exercising the hardware to verify results extracted from the software. The report listed operational attack prerequisites and mitigations based on this comprehensive analysis. This review leveraged that comprehensive analysis to extend its results and we refer to those mitigations and prerequisites where they still apply.

2.4.2 Completeness Limitations

It is well-known that testing is an inherently incomplete process and that no testing process can guarantee absence of flaws. Regarding even the isolated flaws that we evaluated, we cannot guarantee that the changes or repairs that removed, mitigated, or reduced these flaws did not cause regression faults. While we did look for regression faults, it is possible that undetected regression faults may be more dangerous than the flaws that were fixed.

2.4.3 Scope

This project's purpose is to evaluate the target system's technical properties with respect to our list of previously-identified flaws. Our analysis is not comprehensive in any sense. We further emphasize that our results do not extend beyond the scope of our investigation. We do not contend that these systems are correct or secure beyond the specific results given here. A significant amount of new code was introduced which we did not rigorously review for additional security flaws. As a supplemental review, our exclusive purpose is to provide a technical assessment only regarding flaws identified in earlier reviews. We defer interpretation and application of these technical results to FLDoS.

2.5 Diebold Software Architecture

2.5.1 Code Structure

We reiterate earlier findings that the AccuVote TSx™ code was not developed for high assurance. The code style and readability vary significantly across the code base. There is substantial complicated code throughout the system and there is poor internal program documentation, such as missing and weak comments (#23). Detailed design documentation was not available in the basic review, or in this supplemental effort.

2.5.2 Off-The-Shelf Software

As with most modern applications, there are several third-party components to the reviewed voting systems. We did not independently investigate the third-party operating system, associated database system, or driver software.

3 Findings

We group our findings to parallel the SAIT Diebold Report structure. We re-reviewed many items reported as fixed (e.g. #18, 50) or benign (#37, 38) in the SAIT Diebold Report to determine if regression faults may have been injected in this version. We note the only such fault that we detected in section 3.3.14 below.

3.1 Overview

Though there are many changes and some fundamental upgrades in the present software version, it represents only a slight security improvement overall. The interpreter code (section 3.5.1) in this version is cleared of known dangerous commands (e.g. `strcpy`), which constitutes an important safety improvement. We also note that the cryptographic applications continue to expand and accepted standards are beginning to be applied. Major improvements are evident in the bootloader code (see section 3.4.1), where revised verification processes and the repaired signature algorithm solidify software startup procedures.

Nonetheless, the code base is highly unstructured, apparently resulting from institutional development procedure. Many cryptographic practices remain immature with naïve key management and other weak applied-encryption practices permeate the application.

We reiterate that the purpose of this report is to evaluate the *technical properties* of the current systems with respect to previously-identified flaws. Its purpose is neither to condemn nor endorse these systems; rather the findings herein should be carefully considered in the context of the overall election process.

3.2 The Signature Flaw (#51)

One of the most important flaws identified in the SAIT Diebold Report concerned the code's faulty RSA signature verification implementation. The hand-coded RSA signature verification failed to examine the high order bits of the decoded signature allowing signatures to be forged as we implemented in the SAIT Diebold Review.

In the revision, the vendor corrected the signature verification by modifying the code to examine the entire decoded signature at verification. The signatures also now incorporate padding in RSASSA-PKCS1-v1-5 format as per RFC 3447 into the signature process which the code appropriately verifies. This prevents forgeries that we previously identified.

3.3 AV-TSX Firmware Faults

3.3.1 Cryptographic Key Management (#15)

The SAIT Diebold Report identified issues with the foundational symmetric keys utilized to protect TSx data. Two 128-bit AES keys, the data encryption key and the system key, are used for all authentication and encryption, including encrypting and generating the message authentication codes for the stored electronic ballots, generating the message authentication codes for the election database file, and keying the hash that stores the supervisor PIN.

The present version does not address these issues. Thus these weaknesses and the attack prerequisites and mitigations identified in the SAIT Diebold Report paragraph 3.7.1.1 still apply.

3.3.2 Memory Card Update File is Unprotected (#32, 52)

The SAIT Diebold Report noted that the “assure.ini” file onboard the TSx memory card was unencrypted and unauthenticated and was subject to malicious manipulation. An attacker who could remove the card from the voting terminal could modify the file to set the machine into pre-election mode and could create valid voter cards while the terminal was in this mode.

The present version encrypts the assure.ini file, which mitigates this vulnerability, though subject to the key management limitations described in section 3.3.1 of this report and the failure to securely delete files documented in section 3.3.18.

3.3.3 Smart Card Authentication Issues (#7, 39)

While the smart card passwords can be set by election workers, the implemented authentication protocol is not secure, allowing an attacker to potentially create counterfeit, validating smart cards, including voter cards. This flaw remains essentially unchanged from the SAIT Diebold Report paragraph 3.7.1.3 along with the prerequisites and mitigations noted there.

3.3.4 Supervisor PIN is Not Cryptographically Protected (#8, 26, 28)

This flaw remains essentially unchanged from the SAIT Diebold Report paragraph 3.7.1.4. The attack prerequisites and mitigations noted there remain valid in this code version.

3.3.5 Insecure Storage Mount (#9)

This flaw remains essentially as described in the SAIT Diebold Report paragraph 3.7.1.5. The attack prerequisites and mitigations noted there remain valid in this code version.

3.3.6 System Configuration Information is Unprotected (#10)

This flaw remains essentially unchanged from the SAIT Diebold Report paragraph 3.7.1.6. None of the examples identified there were changed and many other instances not specifically noted were visible as well. The attack prerequisites and mitigations remain valid in this code version.

3.3.7 Protective Counter Stored in a Mutable File (#11)

This flaw remains essentially unchanged from the SAIT Diebold Report paragraph 3.7.1.7. The attack prerequisites and mitigations noted there remain valid in this code version.

3.3.8 Ballot Definition File is Unprotected (#12)

This flaw remains essentially unchanged from the SAIT Diebold Report paragraph 3.7.1.8. The attack prerequisites and mitigations noted there remain valid in this code version.

3.3.9 Impersonate a TSX Terminal to GEMS (#14, 25, 35)

In the present version, use of SSL is still optional and is decided by a value read in from the registry. Root certificates and root public keys are hard coded. The machine's private key is stored in a local file. As described in the SAIT Diebold Report section 3.10.3, the client does not examine received server certificates after verifying that they have been signed by the certificate authority. Specifically, the client does not verify that the owner of the received certificate is the expected server. It also seems likely, given the lack of certificate examination, that all clients use the same public/private key pair.

Based on this analysis, we believe that the risk and attack prerequisites and mitigations noted in paragraph 3.7.1.9 apply to the present version.

3.3.10 No Integrity Protection of Stored Electronic Ballots (#17)

This flaw remains essentially unchanged from the SAIT Diebold Report paragraph 3.7.1.10. The attack prerequisites and mitigations noted there remain valid in this code version.

3.3.11 Ballots are Stored Sequentially (#20, 21)

This flaw remains essentially unchanged from the SAIT Diebold Report paragraph 3.7.1.11. The attack prerequisites and mitigations noted there remain valid in this code version.

3.3.12 Candidate Information is Not Stored in the Results File (#13)

This flaw remains essentially unchanged from the SAIT Diebold Report paragraph 3.7.1.12. The attack prerequisites and mitigations noted there remain valid in this code version.

3.3.13 Audit Logs are Not Cryptographically Protected (#28)

This flaw remains essentially unchanged from the SAIT Diebold Report paragraph 3.7.1.13. The logs are encrypted and authenticated the same way as the electronic ballots, using the “system key”, which is insecure. Hence, they are also susceptible to viewing and modification by an adversary. The attack prerequisites and mitigations noted there remain valid in this code version.

3.3.14 Data is Neither Authenticated Nor Encrypted Over the Communication Link (#16, 19, 25, 34, 35)

The SAIT Diebold Report noted that the vendor mitigated this vulnerability by implementing SSL for this communication, but further noted that there were weaknesses in the vendor’s SSL implementation, largely relating to the entropy level for generated seeds. In the present version, the vendor introduced further adjustments to this code that include a possible regression fault.

The changes omit error handling for the random number generation, allowing SSL to potentially execute with important values, such as encryption or signature keys, that could be predicted more easily than if they were randomly selected. No notification or retry would occur in such a case. While we are unable to precisely determine the impact without a test environment, this practice allows unpredictable results at best, and possibly injects unnecessary security vulnerability.

The use of random numbers without sufficient entropy also seems probable since the code uses an operating system call to obtain its entropy and system calls generally provide “randomness” by collecting information from external events. Because SSL is initialized very early after boot, the operating system may not have observed a sufficient number of such events to produce strong entropy. The use of the system call is, however, generally considered to be a good practice so long as sufficient error checking is conducted.

The code should be modified to generate strong entropy or to fail safely with meaningful error reporting if operating conditions preclude proper operation.

The certificate concerns discussed in section 3.3.9 above are also relevant here.

3.3.15 Several TCP/UDP Ports Are Open (#29)

Because we did not have access to a machine, we were unable to test the presence of open ports. This mostly relies on the host operating system and its configuration, so without access to a machine with the proper firmware, we were unable to test this vulnerability with the target software version.

3.3.16 A Local User Can Access the Main Menu/System Setup Menu Without a Smart Card or Key (#55)

This issue, identified in the California TTBR [5], remains an issue with the current software version.

3.3.17 A Malicious Election Resource File on the Memory Card Could Exploit Multiple Vulnerabilities to Run Arbitrary Code (#57, 58)

California TTBR [5] reported two vulnerabilities in the code that processes the election resource file. The code that reads the page number from the RTF format file now conducts length checks before the data is stored it into a buffer, preventing the noted vulnerability.

However, when the resource data for displaying instructions is in bitmap format, they trust the file size to copy it into a buffer. The resource data is read directly from the election definition on the card. This vulnerability remains in the present firmware version.

3.3.18 Files on the Voting Machine are not Securely Erased When They are Deleted (#60)

This issue, identified in the California TTBR [5], remains an issue with the current software version. Our analysis indicates that this weakness may allow an attacker to circumvent the encryption process implemented to protect the assure.ini file with a replay attack (see section 3.3.2). Since encrypted files are not securely deleted, the attacker may capture previously encrypted versions of the assure.ini, database, and resource files, and then copy those files to a memory card. If the memory card is properly setup with the older files and inserted into the machine, the machine will reset the terminal to pre-election mode where voter cards can be created and initialized.

Mitigation: This vulnerability may be mitigated by election procedure. All removable media should be securely wiped between elections. If older files are stored elsewhere, such as on an election center computer, ensure that these files are protected from unauthorized access. Finally, elections officials should ensure that new encryption keys are used for every election.

3.3.19 AV-TSX Startup Code Contains Blatant Errors (#61)

This issue, identified in the California TTBR [5], is not present in the current software version. The identified module no longer exists in the code base and the offending code snippet does not exist in the current version.

3.4 AV-TSX Bootloader Faults

3.4.1 Bootloader Issues (#1, 2, 3, 4, 5, 6, 53)

The bootloader has changed significantly in this version. The booting process now first attempts to start the machine from a locally stored bootable image prior to conducting any other actions. If the initial attempt to boot fails, the bootloader then attempts to use any .ins files found on the removable memory card to install a new operating system and voting software image to the machine's permanent storage. The code now executes a standard signature verification (see section 3.2) over the new image prior to installation to verify its authenticity. This is a significant improvement, which will largely prevent the spread of voting machine viruses that propagate by installing malicious software directly through the bootloader's installation feature.

Finally, Because the bootloader relies on several foundational libraries and we were not provided with a complete code base necessary to run static or dynamic analysis tools, we were unable to rigorously check the bootloader for all buffer overflow vulnerabilities.. Since many such

weaknesses were noted in the California TTBR in this code, we emphasize that such a rigorous evaluation is necessary before this code can be considered to be safe.

3.5 AccuBasic Interpreter Issues

3.5.1 Unchecked String Operations (#40, 42, 43, 44, 45, 46, 47, 48, 56)

All instances of strcpy, strncpy, memcpy, and printf functions have been removed from the AccuBasic interpreter. They have been replaced with safe functions such as strcpy_s and strncpy_s that not only prevent the copy from overflowing, but also null-terminate the referenced strings. This is a notable improvement that mitigates an important class of security flaws.

4 Conclusions

Electronic voting systems are mission critical, testing limited systems that demand high assurance engineering. This report documents our TSx software supplemental review supporting the Florida Department of State voting system certification effort for this software version. Our analysis focused on issues identified in the SAIT Diebold Report, but also considered issues identified during the recent California Top To Bottom [voting system] Review.

Our analysis identified software upgrades that included major improvements in the signature flaw, the bootloader process, and elimination of dangerous commands in the interpreter. The vulnerability associated with the unprotected “assure.ini” file is also largely mitigated.

We conclude with the following summarizing issues.

4.1 Cryptographic Key Management

Cryptographic tools can provide many strong security properties that are demanded by electronic voting systems, but only when founded with strong key management practices. Subtle issues such as use of low entropy, weak key generation, or mild key reuse may seem harmless even to astute observers, but such issues can dramatically weaken the mechanisms they support or even render them useless.

Many of the critical cryptographic flaws that we identified result from naïve cryptographic misuse or seemingly mild deviations from standards. For critical applications, precisely following established cryptographic standards is paramount. Any customizations, variations or optimizations, no matter how mild, have potential to negate all security properties that the fully implemented standard provides.

When unusual situations not covered by standard operations arise, rigorous analysis by skilled cryptography experts is necessary. Secure applications demand rigor well beyond non-sensitive applications without exception.

4.2 Removable Media

All removable media associated with electronic voting systems is highly sensitive and must be protected year-round with the same status and priority as voted ballots. Unsupervised access to an item as simple as a supervisor card and its accompanying PIN may allow an individual to create valid voter cards. In the wrong hands, these cards can cause significant damage to a county’s vote count validity.

Moreover, software associated with data processed from removable media must follow rigorous defense in depth principles since data stored on them is the most vulnerable to malicious tampering. Developers must rigorously follow safe practices for storage, computation, and garbage collection for these perimeter issues.

4.3 Issues to Improve Voting System Security Evaluation

As SAIT Laboratory's fifth voting system code review, several helpful rules of thumb have emerged and been validated through the review process. We offer these insights for future review consideration.

4.3.1 Precise Review Objectives

It is a well known cliché that “If you don't know where you are going, any road will take you there”. Without a precise objective, it is impossible to establish an efficient process or to measure analysis completeness. While open objective reviews to date have provided valuable contributions, their main result has been to identify disqualifying flaws.

Studies that result in system certification demand establishing and rigidly following precise requirements that can provide elections officials with actionable information that allows them to balance competing election system demands to conduct safe, accurate elections.

4.3.2 Team Spans Diverse Skills

Effective voting system security review requires broad and deep knowledge and skills. While mainstream computer scientists offer a firm foundation for these reviews, the analysis quality will benefit from members with specialty software engineering, testing, systems administration, cryptography, red teaming, and project management skills.

4.3.3 Use Structured Note Taking

Again, as voting system software security analysis becomes the norm, iteration is inevitable. Rigorous documentation processes provide tremendous benefit in both coalescing immediate results and for regression analysis. The two SAIT Diebold supplemental reviews proved the value of rigorous, structured documentation. Teams should establish a sound documentation plan and relentlessly stick to it.

4.3.4 Acquire a Complete Development Environment

Software review and proof of concept testing are resource intensive processes that demand domain appropriate skills. Conversely, software development environments can vary greatly. In order to reduce both spin up and proof of concept construction time in the review process, states should require vendors to file complete development environments with their systems. The environment delivered should enable the state, and any review teams, to rebuild the certified binaries. During a review, these should be delivered with the source code to the review team. If the environments are not available through the state, supporting tools and libraries should be provided to the team by the vendor.

4.3.5 Complete Design Documentation

Similar to having development environments, software review can be simplified if accurate, detailed design documentation is available. States should require such documentation from the vendor and make it available to any software analysis effort. If such documentation is not available from the state, the vendor should supply it directly to the review team.

5 Acknowledgments

As part of our work we used the automated source code analysis tool Coverity Prevent SQS to assist with the code review process. We note that Coverity is a SAIT Laboratory partner.

6 References

- 1 “Software Review and Security Analysis for Diebold Voting Machine Software.”, Joint Florida Department of State and Florida State University Statement of Work, May 14, 2007
- 2 Ryan Gardner, Alec Yasinsac, Matt Bishop, Tadayoshi, Kohno, Zachary Hartley, John Kerski, David Gainey, Ryan Walega, Evan Hollander, and Michael Gerke, “Software Review and Security Analysis of the Diebold Voting Machine Software”, Final Report For the Florida Department of State, July 27, 2007, <http://election.dos.state.fl.us/pdf/SAITreport.pdf>
- 3 Kurt Browning, Florida Secretary of State Letter to Mr. David Byrd, Diebold Election Systems, dated July 31, 2007, <http://election.dos.state.fl.us/pdf/SAITbrowningLetter.pdf>
- 4 "Software Review and Security Analysis of the Diebold Voting Machine Software, Supplemental Report", David Gainey, Michael Gerke, and Alec Yasinsac, Security and Assurance in Information Technology Laboratory, August 10, 2007, For the Florida Department of State, <http://election.dos.state.fl.us/pdf/DieboldSupplementalReportFinalSubmission.pdf>
- 5 Joseph A. Calandrino, Ariel J. Feldman, J. Alex Halderman, David Wagner, Harlan Yu, William P. Zeller, "Source Code Review of the Diebold Voting System", July 20, 2007, For the California Secretary of State “Top-to-Bottom” Electronic Voting System Review

Appendix A Flaw List

ID	Old ID	Description	Source	Page #
1	3	fboot.nb0 can be used to load malicious software	[2]	18
2	4	fboot.nb0 can be overwritten without authentication or integrity checks	[2]	18
3	5	Denial of Service with PowerOffSystem() API	[2]	16
4	6	Bootloader replaces itself with eboot.nb0 or fboot.nb0 if found on memory card on boot, no authentication	[2]	18
5	7	Bootloader replaces OS with nk.bin (or some other nk.xxx's) if found on memory card, no authentication	[2]	19
6	8	Bootloader "runs" any .ins files in memory card on boot after prompting user, no authentication	[2]	19
7	13	Lack of smartcard authentication	[2]	14
8	14	PIN sent from smartcard to terminal in cleartext	[2]	14
9	15	Insecure file mount, does not ensure writing to removable media	[2]	15
10	16	Unprotected system configuration file	[2]	15
11	17	Protective counter stored in mutable file	[2]	16
12	18	Ballot definition unprotected and unauthenticated	[2]	16
13	19	Candidate information is not stored in the results file	[2]	18
14	20	Impersonate a legitimate voting terminal, no authentication and data can be gathered from ballot definition files	[2]	17
15	21	Cryptographic key management	[2]	12
16	22	DES CBC encryption uses constant 0 for IV	[2]	17
17	23	No integrity protection of stored vote counts and data	[2]	17
18	24	No sequence numbers stored with votes (to help detect record deletion, may be problematic with respect to privacy)	[2]	15
19	25	No integrity checks, authentication, or encryption on votes transferred to the backend server	[2]	18
20	26	Votes are written serially	[2]	17
21	27	Bad RNG for randomizing vote order	[2]	17
22	28	Audit log problems: no consistency for what is logged, does not verify that printer is attached before writing to it	[2]	18
23	29	Complicated, undocumented code segment	[2]	18
24	34	DES encryption key is hard-coded	[2]	26

25	35	Data is not encrypted when transmitted over data link	[2]	18
26	36	The supervisor's password is hard-coded	[2]	26
27	37	Not applicable to this analysis	[2]	26
28	38	PIN for all smart cards is the same and the factory default	[2]	26
29	39	Several TCP/UDP ports are open		
30	40	Can make counterfeit voter and supervisor cards	[2]	14
31	41	Can vote multiple times with a card writer	[2]	14
32	42	PCMCIA cards are not encrypted	[2]	13
33	43	Not Applicable to this analysis	[2]	24
34	45	Digital certificates only exist at the servers and these are neither signed nor authenticated by the AccuVote terminals	[2]	18
35	46	No GEMs authentication by the AccuVote-TS terminals	[2]	18
36	63	Not applicable for this analysis.		
37	67	Error checking is inadequate for identifying and recovering from failures in a damaged or disfunctional environment	[2]	23
38	68	Error handling makes it difficult to differentiate between malicious activity and failure	[2]	23
39	69	The contents of the smartcards are neither encrypted nor digitally signed	[2]	23
40	70	W1 Array bounds violation: Overwrite any memory address with a 4-byte value that the adversary has partial control over. Allows attacker to inject malicious code	[2]	26
41	71	W3 Input validation error: Choose any memory location and begin executing it as .abo code; could be used to conceal malicious .abo code in unexpected locations, or to crash the machine	[2]	26
42	72	W6 Array bounds violation: Overwrite any memory location with any desired value. Allows attacker to inject malicious code	[2]	26
43	73	W7 Buffer overrun: Corrupt memory, crash the machine	[2]	26
44	74	W8 Buffer overrun, integer conversion bug: Corrupt memory until the machine crashes	[2]	26
45	75	W10 Buffer overrun: Overwrite return address on the stack. Allows attacker to inject malicious code and take complete control of the machine	[2]	26
46	76	W11 Array bounds violation: Information disclosure: read from potentially any memory address. Crash the machine	[2]	26
47	77	W12 Array bounds violation: Writes any 4-byte value to any address. Allows attacker to inject malicious code	[2]	26
48	78	W13 Array bounds violation: Information disclosure: read a 4-byte value from any address	[2]	26
49	79	W14 Pointer arithmetic error may crash machine. Could begin interpreting random memory locations as though they were .abo code	[2]	26

50	80	Three types of tokens used to potentially read and modify data in global memory. (no specifics given)	[2]	26
Flaws Identified During the SAIT Diebold Review				
51		Signature Flaw	[2]	10
52		Memory card file Assure.ini is not protected	[2]	13
Flaws Identified During the California Top To Bottom Review				
53		Multiple buffer overflows in .ins file handling allow arbitrary code execution on startup.	[5]	41
54		Duplicate of #7	[5]	42
55		A local user can get to the Main Menu/System Setup menu without a smart card or key.	[5]	46
56		Multiple vulnerabilities in the AccuBasic interpreter allow arbitrary code execution.	[5]	47
57		A malicious election resource file on the memory card could exploit multiple vulnerabilities to run arbitrary code	[5]	48
58		Malicious election database files can cause arbitrary code execution on the AV-TSX when uploading elections to GEMS.	[5]	48
59		Duplicate of #20	[5]	49
60		Files on the voting machine are not securely erased when they are deleted.	[5]	51
61		AV-TSX startup code contains blatant errors.	[5]	51