Context Description: Posted Dec. 1, 2006

This draft report was prepared by NIST staff at the request of the Technical Guidelines Development Committee (TGDC) to serve as a point of discussion at the Dec. 4-5 meeting of the TGDC. Prepared in conjunction with members of a TGDC subcommittee, the report is a discussion draft and does not represent a consensus view or recommendation from either NIST or the TGDC. It reflects the conclusions of NIST research staff for purposes of discussion. The TGDC is an advisory group to the Election Assistance Commission, which produces voluntary voting system guidelines and was established by the Help America Vote Act. NIST serves as a technical advisor to the TGDC.

The NIST research and the draft report's conclusions are based on interviews and discussions with election officials, voting system vendors, computer scientists, and other experts in the field, as well as a literature search and the technical expertise of its authors. It is intended to help in developing guidelines for the next generation of electronic voting machine to ensure that these systems are as reliable, accurate, and secure as possible. Issues of certification or decertification of voting systems currently in place are outside the scope of this document and of the TGDC's deliberations.

OPEN ENDED VULNERABILITY TESTING OF VOTING MACHINES

November 20, 2006

Version 0.02

TABLE OF CONTENTS

1 Introduction

The ideas developed in this paper on Open Ended Vulnerability Testing (OEVT) is based on the assumption that the system is designed with security from the start and thoroughly analyzed and tested by a team of computer security experts with excellent analytical and hacking skills. Specifically, it is assumed that the following are true:

- A set of voting system security requirements have been developed that are based on pragmatic threat model.
- The voting system has been designed and developed with security in mind in general and with the expressed purpose of mitigated the identified threats and meeting the stated system security requirements.
- The voting system developer has sound software development and engineering process in place and uses these processes to develop the system and to produce qualify documentation that accurately and fully represents the voting system design.
- A team of computer security experts is assembled to understand, analyze, test and perform OEVT on the voting system.
- The team of experts reviews the system design documents to gain a thorough understanding of the system internals, i.e., how the voting system performs the voting functions securely.
- The team of experts analyzes the system for potential flaws in system architecture and design. If any flaws are found, the team works with the developer so that the flaws are rectified.
- The team performs thorough security functional testing as described in other documents and summarized in Section 1.5 of this paper. If any flaws are found, the team works with the developer so that the flaws are rectified.
- The team tests the system against known vulnerabilities as described in other documents and summarized in Section 1.5 of this paper. If any flaws are found, the team works with the developer so that the flaws are rectified.
- The team performs OEVT as described in Section 2 of this paper. If any flaws are found, the team works with the developer so that the flaws are rectified.

Unless a team of computer security experts understand how a voting system works and use that understanding to test the system, OEVT alone can not be used to make broad assertions regarding security of the voting system. In that scenario, most OEVT can tell you is whether the system is vulnerable to specific vulnerabilities for which the system is tested.

1.1 Background

1.1.1 U.S. Election Assistance Commission (EAC)

The U.S. Election Assistance Commission (EAC) was established by the Help America Vote Act of 2002 (HAVA). The Commission serves as a national clearinghouse and resource for information and review of procedures with respect to the administration of Federal elections. HAVA was enacted to establish a program to provide funds to States to replace punch card voting systems, to establish the Election Assistance Commission to assist in the administration of Federal elections and to otherwise provide assistance with the administration of certain Federal election laws and programs, to establish minimum election administration standards for States and units of local government with responsibility for the administration of Federal elections.

1.1.2 Technical Guidelines Development Committee (TGDC)

HAVA also established a 15-member Technical Guidelines Development Committee (TGDC) to assist the Executive Director of the Election Assistance Commission (EAC) in the development of Voluntary Voting System Guidelines (VVSG). HAVA named the Director of the National Institute of Standards and Technology (NIST) to chair the TGDC. In addition, HAVA requires NIST to provide the TGDC with technical support necessary to carry out its duties.

The duties of the TGDC include the gathering and analysis of data and information related to the security of computers, human factors, voter privacy, and methods to detect and prevent fraud.

The TGDC has requested that NIST perform research and draft standards documents requiring testing of voting systems that includes a significant amount of open-ended research for vulnerabilities by an analysis team supplied with complete source code and system documentation and operational voting system hardware.

1.1.3 Security and Transparency Subcommittee (STS)

The EAC has also approved formation of subcommittees, including the Security and Transparency Subcommittee (STS). The purpose of the subcommittee is to deal with relevant issues including security, transparency, human factors, privacy, core standards requirements, and testing of voting systems.

1.2 Purpose

The purpose of this document is to address some (not all) of the security testing aspects of the voting machines. The intent is to use the document to coordinate the views and suggestions from the STS and from TGDC on the Open Ended Vulnerability Testing (OEVT). Once the document is vetted through these subcommittee and committee, it will form the OEVT requirement for certification of voting systems.

1.3 Goals of Open Ended Vulnerability Testing (OEVT)

The goal of OEVT is to discover architecture, design and implementation flaws that have crept into the system which can be exploited or can otherwise provide erroneous results for the election. These flaws are the ones not detected using systematic functional, reliability, and security testing.

The goal of OEVT is not to discover logic bombs, time bombs or other Trojan Horses that may have been introduced in the system hardware, firmware or software. These types of problems can be prevented by using development security controls consisting of physical security, personnel security, procedural security, and technical security for the development environment. These types of problems can be detected by examining the code thoroughly and by conducting code correspondence. These types of problems may also be detected by code analysis tools designed to detect suspicious and malicious code segments.

However, OEVT may deter or discover possible Trojan Horses or the application of the OEVT may identify the degree of difficulty associated with introducing Trojan Horses by bypassing the development security control or distribution security controls.

A goal of OEVT is to identify the nature of collusion that may be required to compromise the security of the voting system.

1.4 Scope

Scope of this document is limited to OEVT. Other forms of testing will be addressed in other document(s). For example, the functional testing, reliability testing, security functional testing, parallel testing, random testing, pre-election testing, etc. are outside the scope of this document.

Scope of the document excludes review and analysis of security documentation and the system security except as it pertains to understanding the system in order to perform effective OEVT. Security analysis of the system is addressed elsewhere. However, there is a great deal of benefit if an expert team analyzed the security of a voting system and then conducted OEVT. In addition, conducting OEVT without, or independent of, or prior to security analysis is not recommended.

The TGDC request to NIST stated that the open-ended vulnerability search and research should not exclude those involving collusion between multiple parties (including vendor insiders). This is interpreted to mean that the scope of OEVT should include pretty much all aspects, including but not limited to the following:

- Development environment
- Distribution
- Configuration
- Physical security while voting machines are:
    - In storage
    - Being configured
    - Being transported, and
    - Being used
- Computer system security

- Communication security

The TGDC request to NIST also stated that the open-ended vulnerability search and research should include those involving adversaries with significant financial and technical resources. This coupled with the fact that most exploitation and associated automation scripts become public, implies that the OEVT needs to be rigorous. In other words, difficulty of the exploit alone should not be used to rule out a vulnerability.

The impact of accidental or intentional mis-configuration is outside the scope of OEVT. This will be analyzed as part of configuration analysis.

While reliability and other errors are not part of OEVT, the scope of OEVT includes ensuring that in case of power outage, most hardware errors, most software errors, the state of voting is known, specially if cast ballots are preserved or not, and if possible the last vote is counted or not.

If cryptography is used in the voting system, the scope of OEVT does not include:

- Cryptographic algorithm, key size, and mode of operation. The voting system shall be required to implement FIPS approved algorithms.

- Cryptographic algorithm implementation. The voting system cryptographic algorithm implementation shall be required to be validated under the NIST Cryptographic Algorithm Validation Program (CAVP). The cryptographic module shall be required to be validated under the NIST Cryptographic Module Validation Program (CMVP).

- Cryptographic protocol analysis if the cryptographic protocol is a FIPS approved or IETF approved standards. However, if the protocol allows options, the selection(s) made by the voting system shall be analyzed in light of the security requirements. For example, if the TLS does not implement client authentication, whether this is acceptable or not, needs to be determined by a cryptographic protocol expert. Yet another example is that of the TLS cipher-suite. The cryptographic protocol expert needs to make sure that the cryptographic algorithms, key sizes and modes of operation are appropriate and in compliance with FIPS.

If cryptography is used in the voting system, the scope of OEVT does include:

- Cryptographic protocol analysis if the cryptographic protocol is neither a FIPS approved nor IETF approved standards. The analyst shall be an expert in cryptographic protocols and shall independently determine which of the security services (e.g., confidentiality, integrity, source authentication, non-repudiation, replay protection, etc.) are required. The expert shall then analyze the cryptographic protocol to ensure that the desires security services are provided by the protocol and there are no flaws in the protocol with respect to these required services.

- Cryptographic protocol implementation.

- Various side channel attacks such as power analysis, error injection, and timing analysis.

- Key management analysis to ensure that the secret and private keys are protected from disclosure and from unauthorized modifications. The scope shall include key management analysis to ensure that the public keys are protected from unauthorized modifications.

- Key management implementation.

Scope of OEVT includes all the hardware, firmware, system software (e.g., operating system, spreadsheet, Database Management System (DBMS), etc.) and the voting application software.

Scope of OEVT need not include exanimation of vulnerabilities that require Internet connectivity if the voting system does not use the Internet. The voting system must remove all network interfaces and software from the machines. If dial-up modems are used, appropriate hardware and software can be left on the machines. Dial-up modems must be securely configured. OEVT must consider dial-up modem based vulnerabilities. This must include accidental or malicious download of the data to a rogue central server. For the central server, the concern will be in the areas of rogue voting machine infecting the server or supplying bogus voting information.

If the voting system uses the Internet, the scope of the OEVT shall include hacking attempts. This kind of testing and passing these tests could be cost prohibitive when one or more components of the voting system use commercial-off-the-shelf operating systems such as Windows and various forms of Unix and Linux. These operating systems are huge in terms of code base and have significant vulnerabilities that can be exploited by hackers over the Internet.

1.5 Assumptions

The OEVT can be a very expensive undertaking. It should not be conducted in isolation. It should build on the secure development methodologies, development testing, and security functional testing. The following assumptions are made for OEVT. In other words, it is assumed that the following has taken place:

- The voting system (including design, source code, test software, etc.) is available to evaluators, testers, and OEVT testers. It is preferred that the system is available in public domain so that it can get widest possible scrutiny and hacking.

- It is assumed that the system was developed using state of the art languages and tools that prevent or significantly reduce the chances of buffer overflow problems.

- It is assumed that the system was designed with security from the start. For instance operating system must be properly locked down. The voting application must use operating system security controls such as identification and authentication, discretionary access control (DAC), and audit to protect the voting software, voting data, voting audit and transaction logs. There is little point in

conducting OEVT on a voting system that has not been design with security in mind; that kind of voting system will never pass OEVT.

- It is assumed that automated tools were used to analyze the software for buffer overflow, memory leaks, dead code, and otherwise suspicious code.

- It is assumed that the developer performed testing on the various levels of abstraction, i.e., unit, module, subsystem, etc.

- It is assumed that the security evaluator performed comprehensive security functional testing. This testing must consider the following:
    - Each external Interface must be tested thoroughly. This must include trying nominal values, boundary values, and erroneous values for all the inputs.
    - Each external interface testing must be sufficiently thorough to create all possible error conditions/codes for the interface.
    - The testing must take the design and implementation into consideration. The testing must cover all code.

- It is assumed that the system has been tested against known or potential vulnerabilities. The testing must include examination of databases such as CVE and the voting forums to ensure that the known and potential vulnerabilities are covered. The testing must be applied not just to the voting application software, but also to system hardware and software.

1.6 Security Analysis, Testing and OEVT Criteria

The evaluation team consisting of the computer security experts shall keep the following objectives in mind while performing the security analysis, while conducting security testing, and while conducting OEVT: Can the voting system security be defeated so that the vote count can be changed. Some of the examples this are:

- Vote is not recorded
- Vote is recorded for the candidate other than selected b the voter
- More than vote is counted for a cast vote
- Vote is not added to final tally
- More than the cast vote is added to the final tally
- Vote count can be manipulated by some one
- Vote is deleted by some one

Secondary area of concern is whether the specific voter's choices can be revealed by the voting system.

In summary, if vote count can not be changed or voter's selections can be revealed. The evaluation team shall be mindful of these security requirements when analyzing and testing the system.

1.7 Document Organization

Section 2 provides the OEVT methodology. This section also described how to apply the OEVT methodology to the various components of the voting system. Finally, this section provides an overview of resources required, and cost and schedule for OEVT.

The document also contains two appendices that will be updated on a regular basis.

2 OEVT Description

It is assumed that the OEVT will benefit from independent testing by the public since the flaws discovered by these testers can be used to develop the vulnerability scenarios. Appendix A of this document is a start towards known vulnerabilities discovered to date in the various voting machines.

OEVT assumes that the tester has already exercised the system for known vulnerabilities. These vulnerabilities can be obtained from CVE data base and voting related forums. NIST proposes to start a clearing house for voting related vulnerabilities. NIST plans to use or build upon CVE taxonomy to catalog the voting system vulnerabilities in order to provide a database that can be used by the OEVT testers and researchers.

The goal of OEVT is to determine if the voting system can be compromised by changing the vote count. More detailed description of the goals in provided in Section 1.6.

In this section, we present OEVT methodology, its application to the various components of the voting system, resources required to conduct OEVT, and cost and schedule for OEVT.

2.1 OEVT Approach

The OEVT methodology was developed by combining the following approaches:

- Flaw Hypothesis;
- Security Assertions Based Hypotheses
- Security Fault Analysis (SFA)
- Ad Hoc Hacking

We first describe each of the four approaches and then a combined approach.

2.1.1 Flaw Hypotheses Methodology

Flaw hypotheses approach consists of the following:

- A team of engineers who fully understand the system interfaces, design and internals brainstorm possible ways to break into the system. During the brainstorming phase, no evaluation of the hypotheses is carried out. The goal is to think outside the box and let the testers develop ideas without being judged

- The team then considers other known vulnerabilities and their applicability to the voting system
    - These could be from similar components
    - These could be from similar or common software engineering errors

- Team evaluates the hypotheses and rejects the ones that are not plausible or have been covered by functional tests

- Team prioritizes the remaining hypotheses based on payoff potential and effort involved to test them

- Team takes the top hypotheses and develops and executes the penetration scenarios

2.1.2 Security Assertion Based Hypotheses Methodology

Security assertion based approach consists of the following steps:

- The developer provides security assertions for the voting system

- The OEVT team verifies that the assertions are sufficient for the requirements

- The OEVT team uses the security requirements and assertions to develop hypotheses to break the system with respect to a requirement or assertions

- Rest of the approach is akin to Flaw Hypotheses

2.1.3 Security Fault Analysis (SFA) Methodology

SFA based approach consists of the following steps:

- The OEVT team with the knowledge of the system identifies inputs and internal probes that will induce software errors that are externally visible and internally handled by the system.

- The OEVT team also identifies inputs and internal probes that will invoke all code paths that would not be exercised otherwise

- The OEVT team uses the developer testing and testing laboratory test cases to eliminate the OEVT test cases from the above that are redundant.

- The OEVT team executes the tests by inducing the errors.

2.1.4 Ad Hoc Hacking

As the name implies, ad hoc hacking consists of hacking experts trying to break the voting system.

2.1.5 OEVT Methodology

The OEVT methodology combines all the approaches discussed above and tailors them to the part of the voting system being tests. These methodologies are described in Section 2.2 below.

2.2 Application of OEVT

The scope of OEVT includes the voting system components (e.g., individual machines), overall voting system, voting system procedures (e.g., transportation of hardware and software, installation and configuration), physical security, and personnel security. In this section we describe the OEVT methodology for each part of the voting system.

2.2.1 Computer System Components

This section applies to individual voting system components such as voting machine, central tabulator, etc. Figure 2-1 below illustrates an example of voting system component. The dotted line indicates the scope of OEVT.

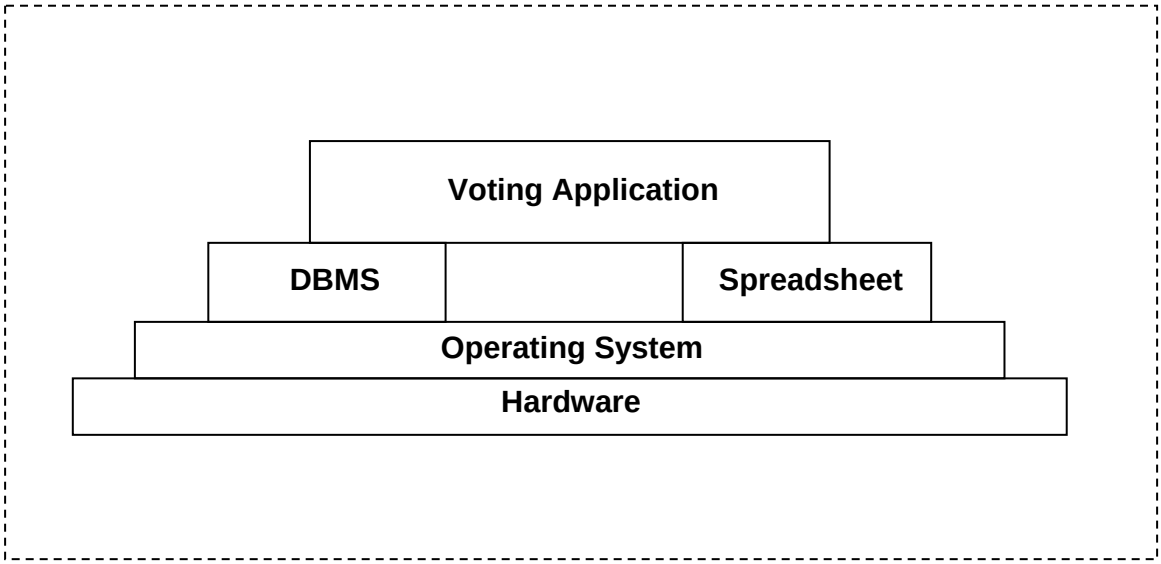

FIGURE 2-1: EXAMPLE OF VOTING COMPUTER SYSTEM COMPONENT

The component includes hardware, system software, and voting application software. For hardware that is also widely used in commercial applications other than voting (e.g., X86 class processors), it may be sufficient to test the hardware interface to demonstrate the correct operation and use the known vulnerabilities database for OEVT. No detailed analysis of hardware internals is required. This also applies to the firmware that is part of widely used hardware.

A voting machine component operating system OEVT could be cost prohibitive if a standard commercial operating system such as Windows or Unix is used. One solution to the problem is that the voting machines use isolation kernels that are minimized for the needs of the voting application. Another alternative is to make simplifying assumptions regarding threats and vulnerabilities. For example, if the voting machine does not use and can not use any of the network interfaces, Internet based hacking vulnerabilities are not applicable. These constitute the majority of threats against the operating systems. Ruling out these vulnerabilities is acceptable only if they truly do not apply, i.e., the system component does not use communication interfaces. Even if Internet hacking threats are ruled out, the size of the operating system code is likely to make local user based vulnerability analysis and testing cost prohibitive.

The following OEVT steps are carried out for each computer system component such as voting machine, central tabulator, etc.:

1. A team of experts in computer software and security is assembled for OEVT. The OEVT team also includes an expert hacker.

2. The OEVT team must be independent, i.e., the OEVT team must have no financial interest in the voting system being tested.

3. The OEVT team is provided all the documentation and a voting system to become fully familiar with the voting system, including the procedures used to install, configure and operate the system. The documentation includes the security requirements, security assertions, and security architecture.

4. The OEVT team reviews the documentation and uses the system to gain full familiarity with what the system does and how it performs.

5. The OEVT team reviews the independent testing laboratory test to gain insight into the degree of rigor applied.

6. The OEVT team uses its knowledge of the system internals and hacking expertise to brainstorm possible ways to break into the system. During the brainstorming phase, no evaluation of the hypotheses is carried out. The goal is to develop hypotheses to break the system.

7. The hypotheses are recoded.

8. The OEVT team examines the known vulnerability testing conducted by the independent testing laboratory.

9. The OEVT team removes the hypotheses whose associated vulnerabilities have been disproved by the independent testing laboratory.[1]

10. The OEVT team identifies any known vulnerability tests that have not been conducted by performing a search for the known vulnerabilities in the database.

11. The OEVT team adds these vulnerabilities to the list of hypotheses developed in Step 7 above. These vulnerabilities are marked as high pay off since they are derived from known vulnerabilities.

12. The OEVT team makes a broad search for vulnerabilities and flaw hypotheses and their applicability to the voting system. The vulnerabilities and hypotheses would come from the public database.
    a) The OEVT team considers vulnerabilities and hypotheses in similar systems.
    b) The OEVT team analyzes vulnerabilities and hypotheses in other systems to determine their applicability to the voting system being tested
    c) The search applies to the system software such as the operating system, database management system, networking software, spreadsheet, etc.

13. The OEVT team adds these to the list of hypotheses.

[1] The brainstorming is done early at the expense of economy in order to obtain the maximum possible list of potential vulnerabilities.

14. The OEVT team verifies that the security assertions provided by the developer are sufficient to address the security requirements. If not, the OEVT team works with the developer to ensure that the security requirements and security assertions are appropriate for the system.

15. The OEVT team uses the security requirements and assertions to develop hypotheses to break the system with respect to a requirement or assertions.

16. The OEVT team adds these hypotheses to the list of hypotheses.

17. The OEVT team uses its knowledge of the system to identify inputs and internal probes that will induce errors that are either externally visible or internally handled by the system.

18. The OEVT team also identifies inputs and internal probes that will invoke all code paths that would not be exercised otherwise.

19. The OEVT team uses the developer testing and testing laboratory test cases to eliminate the OEVT test cases from the above that are redundant.

20. The OEVT team adds the remaining test cases to the list of hypotheses and marks them has high payoff potential.

21. The OEVT team evaluates the cumulative hypotheses and rejects the ones that are not plausible or have been covered by functional and penetration tests conducted by the independent testing laboratory.

22. The OEVT team prioritizes the remaining hypotheses based on payoff potential and effort involved to test them.

23. The OEVT team takes the top hypotheses and develops and executes the penetration scenarios.

2.2.2 Overall Computer System

Figures 2-2 and 2-3 illustrate two examples of computer voting systems and the scope of OEVT for the computer systems. Again the scope of OEVT is illustrated by the dotted lines. Note that in Figure 2-3 vote selection machine and its interface with the vote counting machine is not critical to the security since audit and counting machine is the official count machine. While the vote selection machine communicates with the audit and vote counting machine, a properly designed vote counting machine can not be infected by the vote selection machine. The external interface of the vote counting machine, however, is part of the OEVT.

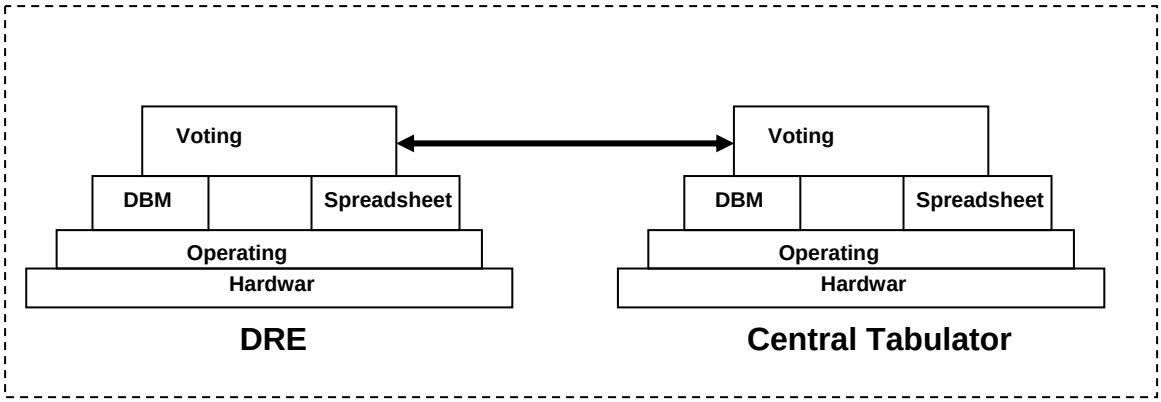

FIGURE 2-2: OVERALL VOTING COMPUTER SYSTEM OEVT: DRE

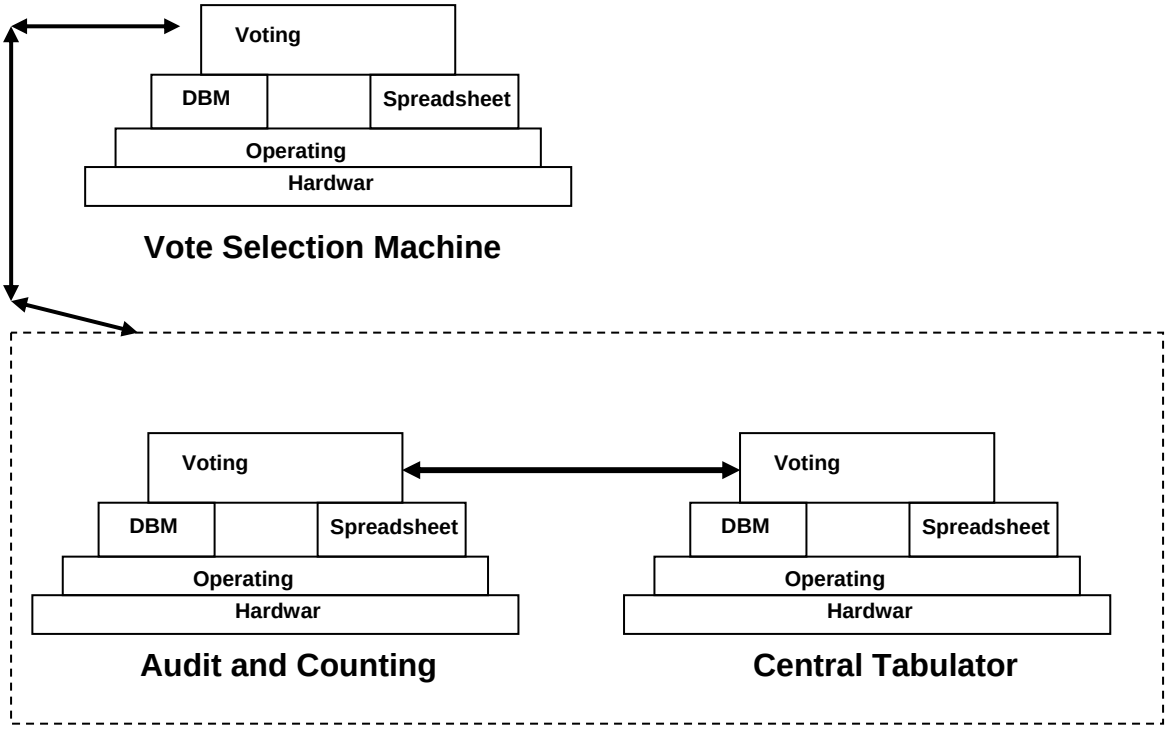

FIGURE 2-3: OVERALL VOTING COMPUTER SYSTEM OEVT: AUDIT MACHINE

The overall computer system OEVT is carried out after OEVT on each individual computer system component is carried out. The same OEVT team that performed the OEVT on the computer system components must perform OEVT on the overall voting computer system.

1. The OEVT team uses component OEVT to brainstorm possible ways to break into the overall system. During the brainstorming phase, no evaluation of the hypotheses is carried out. The goal is to develop hypotheses to break the system. The focus is on how the interfaces can be manipulated using a computer system component or communication channel to defeat the security of the voting system. It is assumed that known vulnerabilities and publicly available

hypotheses for the overall computer system have been executed and the system has been found to be free of these vulnerabilities.

2. The OEVT team evaluates the hypotheses and rejects the ones that are not plausible or have been covered by functional and penetration tests conducted by the independent testing laboratory, or by component OEVT.

3. The OEVT team prioritizes the remaining hypotheses based on payoff potential and effort involved to test them.

4. The OEVT team takes the top hypotheses and develops and executes the penetration scenarios.

2.2.3 Overall Voting System

Figure 2-4 illustrates the components of the overall voting system.

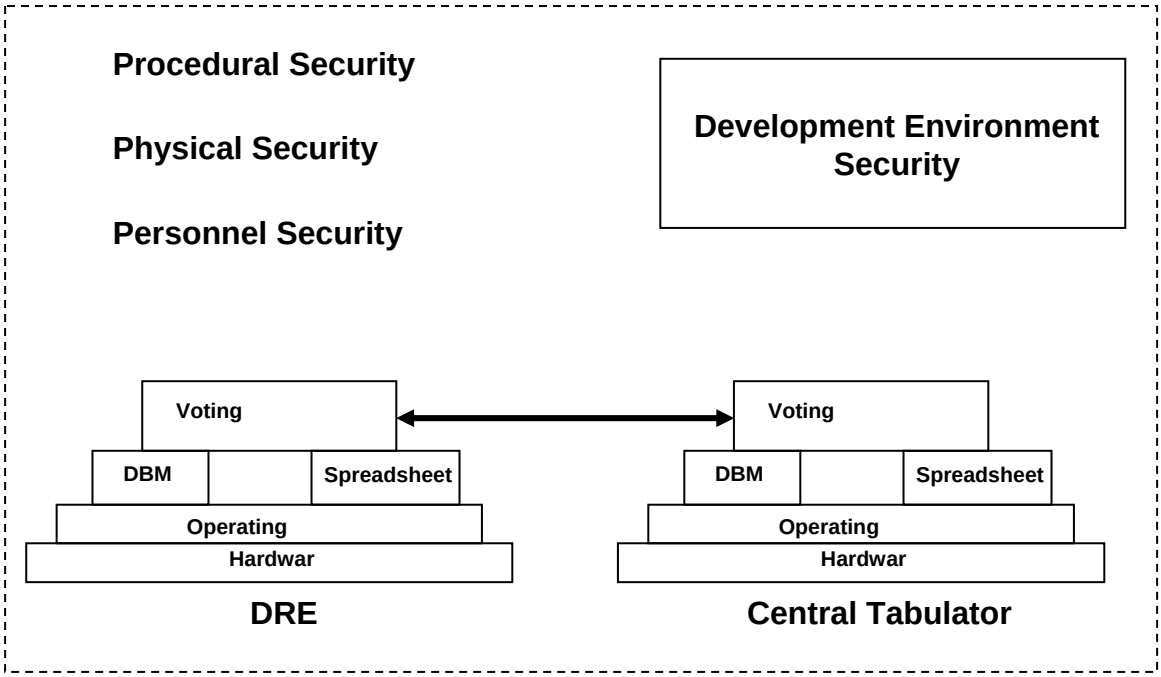

FIGURE 2-4: OVERALL VOTING SYSTEM OEVT SCOPE

The overall voting system OEVT is carried out after the overall computer system OEVT. The same OEVT team that performed the OEVT on the overall computer system must perform OEVT on the overall voting system.

1. The OEVT team uses overall computer system OEVT to brainstorm possible ways to break into the overall system by compromising physical, procedural, and personnel security controls. During the brainstorming phase, no evaluation of the hypotheses is carried out. The goal is to develop hypotheses to break the system.

2. The OEVT team evaluates the hypotheses and rejects the ones that are not plausible or have been covered by functional and penetration tests conducted by the independent testing laboratory, or by component OEVT.

3. The OEVT team prioritizes the remaining hypotheses based on payoff potential and effort involved to test them.

4. The OEVT team takes the top hypotheses and develops and executes the penetration scenarios.

2.2.4 Development Environment

The same OEVT team also performs a high level OEVT on the development environment for the voting system software.

The OEVT team need not perform this step for hardware that is widely used on other commercial application (e.g., X86 processor)

The OEVT team need not perform this step for system software that is widely used on other commercial application. Examples of these include Windows operating system, Unix operating system, DBMS such as Oracle, Informix, SQL Server, etc.

The purpose of OEVT on the development environment is to determine possibility of injection of internal errors in the voting software. The OEVT team carries out the following activities:

1. Reviews the personnel, physical, and procedural security controls from two points of view.
    a) Are the controls sufficient in light of the overall development environment security; and
    b) Are the controls identified in place? The OEVT could rely on the findings of the independent testing laboratory in terms of whether the physical and personnel controls are in place.

2. Review the technical controls including network security controls (both for the Internet and for the LAN), computer security controls, and cryptographic protection used to determine how plausible it is for a hacker to install Trojan in the voting system.

3. Review the procedural and technical controls such as code reviews, code correspondence activities, and use of automated tools to determine how plausible it is for an insider to install a Trojan that can go undetected.

2.3 Resource Requirements

2.3.1 OEVT Team Expertise

The OEVT team should consist of 3-7 computer software, security, and hacking experts. The size of the team depends on the size and complexity of the voting system. All testers must have analytical skills to understand the voting system design. This requires educational and/or implementation experience in computer systems, preferably in system software such as operating systems. The OEVT team also requires at least one

expert hacker. In addition, if cryptography is involved, at least one expert in analyzing the security of cryptographic protocols is required.

2.3.2 Effort Required

The effort required is dependent on the size and complexity of the software in voting system components. To illustrate the effort expected, about 6 weeks of 4 experts is required to perform OEVT on an operating system with 50,000 lines of code. Assuming 2 weeks of additional effort in assessing physical, procedural, personnel and development environment security, a voting system with 50,000 lines of code (including operating system, other system software, and voting application) will require about 32 staff weeks or about 1,300 hours. Assuming a rate of $250 per hour for an expert, this is about $325,000. These numbers represent a data point only. Furthermore, these numbers represent the effort estimate when the security is designed from the start, i.e., the requirements are well specified and the developer has designed the system with security in mind.

However, note that no current voting machine is of such simplicity due to the size of the operating system. In addition, the effort estimate is for OEVT when the system has been designed with security in mind and other security evaluation activities (outside the scope of this effort estimate) have been conducted. Examples of these activities include security analysis, design analysis, document review, and security functional testing.

Finally, when the system size (e.g., lines of code) rises, the effort estimate for OEVT is not likely to rise linearly, but exponentially.

Appendix A: Known Vulnerabilities

This appendix contains the list vulnerabilities that have been exhibited by the current voting machines and voting systems. It also includes the vulnerabilities that have been surmised, but not fully demonstrated.

Appendix B: Flaw Hypotheses

This appendix provides a list of flaw hypotheses. The hypotheses are different from known vulnerabilities, but known vulnerabilities have played a part in list them.

These hypotheses are for illustrative purposes only. The OEVT tester must use the methodology described in this document and the actual voting system to develop applicable hypotheses.

List of Acronyms

CAVP	Cryptographic Algorithm Validation Program
CMVP	Cryptographic Module Validation Program
CVE	Common Vulnerability and Exposures
DAC	Discretationary Access Control
DBMS	Data Base Management System
DRE	Direct Recording Electronic
EAC	Election Assistance Commission
FIPS	Federal Information Processing Standard
HAVA	Help America vote act
IETF	Internet Engineering Task Force
LAN	Local Area Network
NIST	National Institute of Standards and Technology
OEVT	Open Ended Vulnerability Testing
SFA	Security Fault Analysis
SQL	Structured Query Language
STS	Security and Transparency Subcommittee
TGDC	Technical Guidelines Development Committee
U.S.	United States
VVSG	Voluntary Voting System Guidelines