An Experience in Testing the Security of Real-world Electronic Voting Systems

Davide Balzarotti, Greg Banks, Marco Cova, Viktoria Felmetsger, Richard Kemmerer, William Robertson, Fredrik Valeur, and Giovanni Vigna

Abstract—Voting is the process through which a democratic society determines its government. Therefore, voting systems are as important as other well-known critical systems, such as air traffic control systems or nuclear plant monitors. Unfortunately, voting systems have a history of failures that seems to indicate that their quality is not up to the task.
Because of the alarming frequency and impact of the malfunctions of voting systems, in recent years a number of vulnerability analysis exercises have been carried out against voting systems to determine if they can be compromised in order to control the results of an election. We have participated in two such large-scale projects, sponsored by the Secretaries of State of California and Ohio, whose goals were to perform the security testing of the electronic voting systems used in their respective states. As the result of the testing process, we identified major vulnerabilities in all the systems analyzed. We then took advantage of a combination of these vulnerabilities to generate a series of attacks that would spread across the voting systems and would "steal" votes by combining voting record tampering with social engineering approaches. As a response to the two large-scale security evaluations, the Secretaries of State of California and Ohio recommended changes to improve the security of the voting process. In this paper, we describe the methodology that we used in testing the two real-world electronic voting systems we evaluated, the findings of our analysis, our attacks, and the lessons we learned.

Index Terms—Voting Systems, Security Testing, Vulnerability Analysis

1 Introduction

Electronic voting systems are becoming a pivotal element of many modern democracies. National governments and local administrations are continuously looking for ways to streamline the voting process and increase voter participation. The use of computer-based systems to collect and tally votes seems to be a logical and effective choice to accomplish these goals.

Unfortunately, real-world implementations of voting systems have been marked by a series of problems that worry both technologists and the general public. For example, a report published in January 2008 describes the problems encountered in Sarasota County, Florida, when counting the votes in the November 2006 Congressional District 13 election [1]. In this case, 17,846 ballots (14.9% of the total number of votes) cast on electronic voting machines showed no vote for either candidate in the race. In addition, the race was determined by only 369 votes. The report described the system responsible for recording and tallying the votes as a "badly designed, shoddily-built, poorly maintained, aging voting system in a state of critical breakdown."

A quote attributed to Stalin captured the critical role of voting systems years before computer-based systems became widespread: "Those who cast the votes decide nothing. Those who count the votes decide everything." Because of their critical role in determining the outcome of an election, electronic voting systems should be designed and developed with the same care that is given to other critical systems, such as air traffic control systems or nuclear power plant monitors. However, a number of recent studies have shown that most (if not all) of the electronic voting systems being used today are fatally flawed [2], [3], [4] and that their quality does not match the importance of the task that they are supposed to carry out.

Until recently, electronic voting systems have been certified by third-party evaluation companies. Most of the time, these companies test the general functionality of the systems, their usability, and their accessibility for disabled voters. However, in the past no substantial security testing was performed to identify serious security vulnerabilities that affected multiple components of the system and could result in an overall failure to provide reliable vote tallying.

Recently, a number of US states (in particular California, Ohio, Florida, and New York) have commissioned studies to test the security of the electronic voting machines that were to be used in forthcoming elections. Our team was involved in the California Top-To-Bottom Review (TTBR) in July 2007 [5] and in Ohio's Evaluation & Validation of Election-Related Equipment, Standards & Testing (EVEREST) in December 2007 [6]. In the former, we evaluated the Sequoia voting system [7], [8], while, in the latter, we evaluated the ES&S system. Our

- *D. Balzarotti is with the Eurecom Institute in Sophia Antipolis, France. E-mail: Davide.Balzarotti@eurecom.fr*
- *G. Banks, M. Cova, V. Felmetsger, R. Kemmerer, W. Robertson, F. Valeur, and G. Vigna are with the Department of Computer Science, University of California, Santa Barbara, CA, 93106-5110. E-mails: {nomed,marco,rusvika,kemm,wkr,fredrik,vigna}@cs.ucsb.edu*

task was to identify, implement, and execute attacks that could compromise the confidentiality, integrity, and availability of the voting process. As a result of the security testing performed in these studies, the systems used in California were decertified and those used in Ohio were recommended for decertification.

In this paper, we present the methodology and tools that we developed in the security testing of these voting machines, the results of our analysis, the attacks that we devised against the systems, and the lessons learned in the process of testing real-world, widely-used implementations of voting systems. These systems have peculiar characteristics that make the testing of their security particularly challenging. We believe that our experience can help other researchers in this field, as well as policy makers and certifying organizations, develop more rigorous procedures for the security testing of voting systems.

We have studied the systems and voting procedures described in this paper in the context of US-based elections. Therefore, the results and lessons learned reported here might not necessarily apply without modification to other countries. However, we believe that the problems we found would affect any election that relies on the voting systems that we evaluated. In addition, we are confident that some of the lessons learned in our experience with the security evaluation of the two voting systems that we analyzed apply to other voting systems and election procedures as well.

The rest of this paper is structured as follows. In Section 2, we discuss the underlying motivation for performing the security evaluation of the two electronic voting systems that we analyzed. In Section 3, we present some historical remarks on the voting process, and we discuss related work that provided context for the evaluation efforts in which we were involved. In Section 4, we provide an overview of electronic voting systems, and we describe the challenges that one faces when testing the security of real-world voting systems. In Section 5, we present the methodology that we devised (and followed) during the evaluations, while, in Section 6, we present the tools and techniques that we developed in order to support our testing approach. In Section 7, we present, in detail, the findings of our analyses. Then, in Section 8, we show how the vulnerabilities we found could be used to mount a series of attacks whose goal is to tamper with the outcome of an election. Finally, in Section 9, we describe the lessons we learned in evaluating real-world electronic voting systems, and, in Section 10 we conclude.

2 Motivation

Testing and evaluating any system, if done right, is generally a time-consuming and tedious endeavor. One might ask then, "why dedicate all this time and effort to test the security of these particular systems?" In other words, what makes this a novel and useful activity from a scientific research perspective? There are several answers to this question, all of which provide the necessary motivation needed for this research.

First and foremost, the security testing of electronic voting systems is not a well-documented or often-undertaken effort. As was mentioned previously, the testing and certification of these systems is generally focused on usability, is funded by the company that is to profit from the system, and is a private affair (i.e., the results of testing are never made public). The TTBR and EVEREST reviews are the first of their kind, providing the project participants with not only source code, but also the machines and guidance necessary to recreate a realistic electronic voting system. This allowed for the actual execution of attacks that were previously only theoretical because of physical limitations (e.g., the lack of hardware and system binaries) and the validation that the attacks were actually effective.

Second, the nature of such systems (i.e., their common aim and ability to perform a function defined by a set of laws and state-mandated procedures) gives rise to a common high-level design among these systems. This common framework deserves a well-vetted and publicly-executed testing methodology. While current standards provide a "checklist" of characteristics that must be verified, there are no guidelines or suggestions on *how* these characteristics can or should be verified.

Finally, the thorough third-party evaluation of such systems necessitates a certain amount of skill and knowledge that we, as systems security researchers, possess: namely the ability to reverse engineer complex systems in an effort to violate assumptions and successfully coerce a system into an undesirable, insecure state. The two evaluations in which we participated were interesting exercises in applying this knowledge in novel ways to real-world systems in order to develop techniques and tools that can serve to educate and direct future voting system testing.

The focus of these evaluations, then, was twofold:

1) Gain testing experience that can be used to help develop a general methodology for testing electronic voting systems and, thus, provide a stepping stone for future auditors and researchers.
2) Utilize our reverse engineering experience to expose vulnerabilities in the voting system components that we analyzed, develop tools to interact with them, and gain a deeper understanding than would be possible otherwise.

The approach we followed and the results we obtained are reported in the following sections.

3 Electronic Voting History and Related Work

The conduct of elections has changed numerous times and in many ways in the course of democracy [9]. In particular, if one looks at the history of voting in the US, it seems possible to recognize a common, repeating pattern driving its evolution. The introduction of new voting requirements (e.g., the right to voter privacy) and the manifestation of limitations of the existing voting technology repeatedly opened the way to the introduction of novel voting equipment. This cycle of inadequacies and technical advancements has motivated some authors to talk about a "fallacy of the technological fix" in the context of voting [10].

It is possible to identify several examples of these turning points in voting technology. For example, the secret, or "Australian," ballot, in which voters mark their choices privately, on uniform ballots printed at public expense and distributed only at polling places, was introduced in 1888 to substitute for the practice of voting openly (e.g., viva voce) or on custom-made, partisan paper ballots, which made it all too easy to intimidate voters and otherwise influence the vote casting process. Mechanical lever machines became the most commonly used voting device in larger urban centers by the 1930s, when it was clear that the secret ballot was open to subjective ballot interpretation and, thus, was vulnerable to manipulative vote counting. These new machines brought with them additional novelties, such as the use of private curtained booths and new ballots using tabular layouts. Unfortunately, by recording only the total of the votes cast rather than the individual votes, they also made it impossible to perform recounts or otherwise verify the final results. The next wave of innovation occurred in the early 1960s, when first punch card machines and, later, optical mark-sense scanners were introduced. These machines promised more accurate vote recording and faster tallying. Finally, direct recording electronic (DRE) machines started to be utilized for voting purposes around 1975.

The decisive transition toward electronic voting systems, which are currently in use in the vast majority of the US, happened only after the 2000 US presidential election. In the decisive Florida race, Bush led Gore by 537 votes out of over 6 million votes cast, amid a number of irregularities and problems, some of which have been attributed to the use of old and inadequate voting technology. In particular, controversy raised around the confusing design of certain ballots (e.g., the infamous "butterfly" ballot used in Palm Beach County) and the appropriate method of counting ballots in the presence of "hanging" or "dimpled" chads (incompletely-punched holes in ballots with, respectively, one or more corners still attached or all corners attached but a visible indentation).

The legislative answer to the Florida debacle was the Help America Vote Act (HAVA) [11], which was enacted in October 2002 and aimed at completely reforming election administration [12]. In particular, HAVA established minimum election standards, thus providing a least common denominator in an otherwise patchwork of different voting systems and regulations. Second, it created the Election Assistance Commission (EAC), with the tasks of acting as a national clearinghouse regarding election administration, developing and adapting voluntary voting system guidelines, and testing and certifying voting systems through accredited laboratories. Finally, the act provided $3.86 billion in funding over several years to replace older voting technology, train poll workers, increase accessibility, and start research and pilot studies.

The newly-passed legislation and the available funding convinced many states and counties to adopt electronic voting systems. As of 2007, $1.2 billion remained available to states, with the majority of the funds being expended for acquiring voting systems and technologies [13].

Unfortunately, electronic voting systems were far from being the final solution to voting problems. In fact, technology alone does not guarantee the absence of irregularities or problems. For example, in the 2008 senatorial race in Minnesota, almost 7,000 ambiguous ballots were challenged by the two leading campaigns, even though the vast majority of the counties used modern optical scanners for counting ballots. After a review of the disputed ballots, the victor was decided by a mere 225 votes [14]. To put these figures in perspective, it is worthy to remember that Minnesota has 4,131 precincts. In addition, new technologies have often been found to be faulty. Citizen advocacy groups and others have collected and reported malfunctions associated with the use of electronic voting systems. It is not uncommon for these lists to contain hundreds of documented failures affecting essentially all of the existing voting systems [15], [16], [17]. The most recurring problems are related to the reliability of the voting machines, both in terms of accuracy and availability. The consequences of such problems range from long lines at precincts and lengthy recounts to lost votes and sudden switches in election results.

There are several reasons for the poor quality of existing voting systems. First, HAVA has created an incentive for counties to rush into using DREs. In some cases, jurisdictions may have done so without adequate preparation, and, as a consequence, have experienced failures during an election. Similarly, vendors may have rushed their products to market in order to beat competitors, at the expense of careful testing and quality assurance programs.

Second, effectively managing the life cycle of a voting system is a complex task, requiring the careful combination of people, processes, and technology [12]. For example, voting systems are developed by vendors and purchased as commercial-off-the-shelf (COTS) products by the state and local administrators that are actually in charge of running an election. These groups often have limited resources for testing the voting machines, identifying problems, and having them fixed in a timely manner.

Third, voting machines are complex software artifacts that need to account for and conform to a number of technical and procedural requirements and policies. This situation is further complicated by the fact that the US election process can be seen as an assemblage of 51 somewhat distinct election systems (those of the 50 states and the District of Columbia), each with its own rules and exceptions. In addition, it is common for states to further decentralize this process, so that voting details are carried out at the city or county level.

Finally, the standards that set the functional and performance requirements against which the systems are developed, tested, and operated have often been found to be inadequate [12], [18], [19]. Among the reasons for concern, critics include vague and incomplete security guidelines, insufficient documentation requirements, and inadequate descriptions of the configuration of commercial software.

From the security point of view, a natural question to ask is whether the weaknesses demonstrated by electronic voting systems could be leveraged by attackers to subvert the results of an election, for example causing incorrect tallying of votes or violating other properties of the voting process, such as user confidentiality. The first concerns in this area largely predate the HAVA reform [20], [21]; however, we focus here on security evaluations of modern, current electronic voting systems.

The first analysis of a major electronic voting system was performed in 2003, when Bev Harris discovered that the software repository of the Diebold system (including source code, precompiled binaries, and other documents) were stored on a publicly-accessible FTP server [22]. After downloading the files and testing the system, in particular the election management component, she discovered various ways to change votes, bypass passwords, and alter audit logs [23].

The same Diebold repository was also analyzed by a team of researchers from Johns Hopkins and Rice Universities [2], [24]. Their analysis, which focused on the source code of the AccuVote-TS voting terminal, found serious and wide-reaching problems and concluded that the system was "far below even the most minimal security standards applicable in other contexts." Subsequent studies taking into account the actual hardware, election procedures, and environment confirmed the main results of the Hopkins-Rice study [25], [26].

The first independent security assessment of a complete voting system, including both the hardware and the software components, is due to Feldman et al. [27]. The Princeton team obtained the system from an undisclosed "private party," proved that malicious code could be easily installed on the machine to undetectably steal votes and modify records and audit logs, and developed a virus with the capability of spreading from one machine to another. They also uncovered several problems with the physical security of the terminal and, in particular, found that standard furniture keys were sufficient to open the door of the machine that protects the memory card that stores the votes [28].

More recently, as access to electronic voting equipment has increased either through officially sponsored initiatives or unofficial channels [29], assessments of their security features have also become more common. Available studies demonstrate that problems are present in most systems, independent of the specific system's vendor. For example, Gonggrijp and Hengeveld examined the Nedap ES3B system used in the Netherlands and reported a number of fundamental weaknesses (including compromising electromagnetic emanations) that make it "unsuitable for use in election" [30]. Proebstel et al. discovered several exploitable issues in Hart's eSlate system [3]. Ryan and Hoke discussed problems with Diebold's GEMS software [31]. Yasinsac et al. discovered several software vulnerabilities in ES&S iVotronic [4]. More recently, some of the other teams involved in the security assessments of California and Ohio have also described their experiences and findings [32], [33].

While most of the studies have been directed at DRE systems, problems are not limited to one category of electronic voting technology. Optical scanners also contain security-relevant vulnerabilities [34], [35], [36].

The possibility of voting over the Internet and Internet-based voting systems have also received great scrutiny and showed similarly severe security issues [37]. For example, the SERVE system was designed to allow US overseas voters and military personnel to vote in the 2004 primary and general elections, but the system was canceled after a study uncovered a number of critical vulnerabilities [38], [39]. Another relevant example is the system in use for the election of the *Assemblée des Français de l'étranger*, which, in turn, nominates twelve members of the French Senate. The system was assessed by Appel, who deemed it to be lacking the basic mechanisms to assure that an election is conducted accurately and without fraud [40].

Flaws in current voting systems, which were discovered through testing and other analysis techniques, have stimulated a number of research efforts to mitigate the problems in deployed voting systems [41]. These efforts focused on ameliorating security primitives, such as the storage of votes [42], [43] and auditing [44], and on formally assessing and making procedures more effective [45], [46]. In addition, a number of efforts have been proposed to improve the design of voting machines [47], [48], to develop new voting technologies [49], or to introduce novel voting protocols, often with particular desirable characteristics such as verifiable voting in absence of trusted components [50] and privacy against adversaries with unbounded computational resources [51].

This paper differs from the aforementioned studies in several significant aspects. With respect to earlier efforts, the reviews in which we participated had a much broader scope: we had (almost) full access to source code, documentation, actual voting machines, and procedure descriptions used in real elections. We also had the opportunity to test complete voting systems rather than single pieces of equipment. These factors allowed us to test the security of the system more thoroughly (e.g., confirming vulnerabilities detected through source code analysis by developing actual, working exploits) and to assess the security implication of combining several vulnerable components (e.g., showing how a virus could spread from one component of the voting system to another).

The systems we tested were used in contexts where the act of casting a vote and the transmission of ballots over a network (e.g., the Internet) was prohibited by law. Therefore, we did not have a chance to explore problems arising in this situation. However, some of the components we reviewed could be interconnected by an internal network, physically separated and disconnected from external networks. We investigated this scenario and documented the corresponding threats.

Finally, the contributions of this paper do not consist of a novel voting technique or system. Instead, this paper provides a comprehensive review of the methodology, techniques, and tools we developed and used in our testing experiences, and a summary of the lessons learned during our studies. The focus and the details we provide on these aspects of our work also differentiate our paper from those produced by other teams involved in the California and Ohio evaluations. We believe that our methodology is general and can be adapted and applied even when the settings of the testing are different than the projects we participated in, and we expect that this information will be useful to other researchers who want to test electronic voting machines.

4 Voting Systems

Electronic voting systems are complex distributed systems, whose components range from general-purpose PCs to optical scanners and touch-screen devices, each running some combination of commercial off-the-shelf components, proprietary firmware, or full-fledged operating systems. In this section, we present a description of the components that most frequently are part of an electronic voting system. Then, we describe what the peculiarities of these systems are and why testing their security is challenging.

4.1 Components of the system

The components of an electronic voting system are:

- DRE - Direct Recording Electronic voting machine - is a device to record the voter's choices. The DRE is usually a touch-screen device where the voter casts his/her vote. Figure 1 shows the DRE voting machines of Sequoia and ES&S.
- VVPAT - Voter-Verified Paper Audit Trail - is a paper-based record of the choices selected by the voter. The VVPAT printer is hooked to the DRE and the paper record is viewable by the voter, but it is under a transparent cover so that it cannot be modified other than through the normal voting process. The VVPAT for the ES&S DRE can be seen on the left of the touch-screen area in Figure 1.
- EMS - Election Management System - is the system responsible for the initialization of the components that collect the votes and also for the final tallying of the votes. The EMS is usually located at election central and it is often implemented as software running on a commodity PC.
- Optical Scanner - is an optical reader that counts votes cast on paper ballots. There is usually one scanner at each polling site and one at election central (e.g., for the counting of absentee ballots[1]). Figure 2 shows Sequoia and ES&S optical scanners.
- DTD - Data Transport Device - is a storage device to transfer data between different components of the systems. These devices are used to transport ballot information to the DREs and optical scanners at the polling site and to transport voting results to the EMS. Figure 3 shows the DTDs used in the ES&S voting system.

Prior to the election, ballot information is prepared on the election management system at election central. This information may be directly entered into the DREs and the optical scanners, or it may be written onto DTDs that are sent to the polling places, separate from the DREs and scanners. Paper ballots for each of the polling places are also prepared at election central.

On election day, prior to the start of the voting process, if the DREs and optical scanners were not initialized at the central location, then they are initialized with the appropriate ballot information at the polling site, using the DTDs that were sent to the polling place separate from the DREs and scanners. After the DREs are initialized (or simply powered up, if they were initialized at election central), they are tested with sample votes to see if they record everything accurately. The optical scanners are tested in a similar way. This is called the pre-election logic and accuracy testing (pre-LAT) phase. If the DREs and the scanners pass the pre-election testing, then they are deemed ready to be used for voting.

When a voter comes to the polling place, he/she registers at a desk. Then, either the voter is given a token (e.g., a smart card) to insert into the DRE to start voting, or the election official carries the token and inserts it into the DRE on the voter's behalf. In the case of the voter carrying the token, he/she removes the token and returns it to the election official when he/she is finished voting. If the election official initiates the voting session, the token is usually removed before the voter starts to cast his/her ballot. The voter's selections are displayed on the DRE screen and are also printed on the VVPAT.

If paper ballots are used, the voter is given a ballot and a marking device to cast his/her vote. When the voter is through, the ballot is handed to an official who inserts it into the optical scanner to be read and recorded. Some optical scanners will report an undervote (voting for less than n choices when n are supposed to be picked) or an

1. An absentee ballot is a ballot that is cast without physically going to a polling place on election day. Absentee ballots are usually sent by mail to election central.

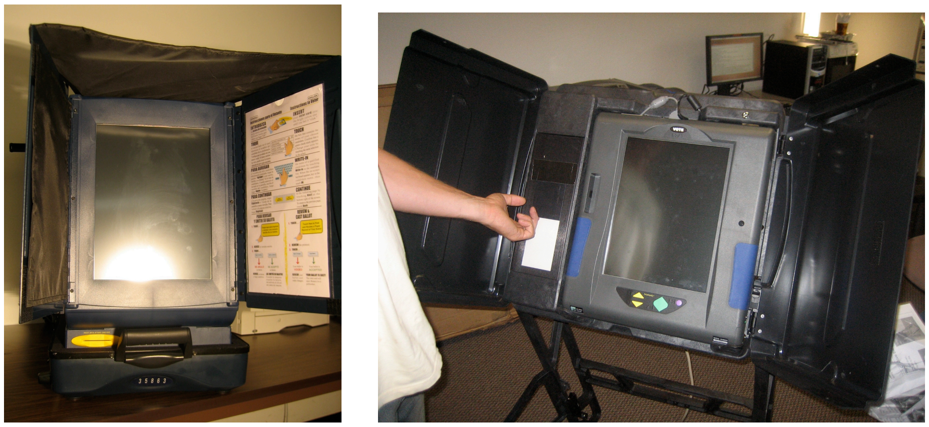


Fig. 1. The DRE voting machines from Sequoia (left) and ES&S (right).

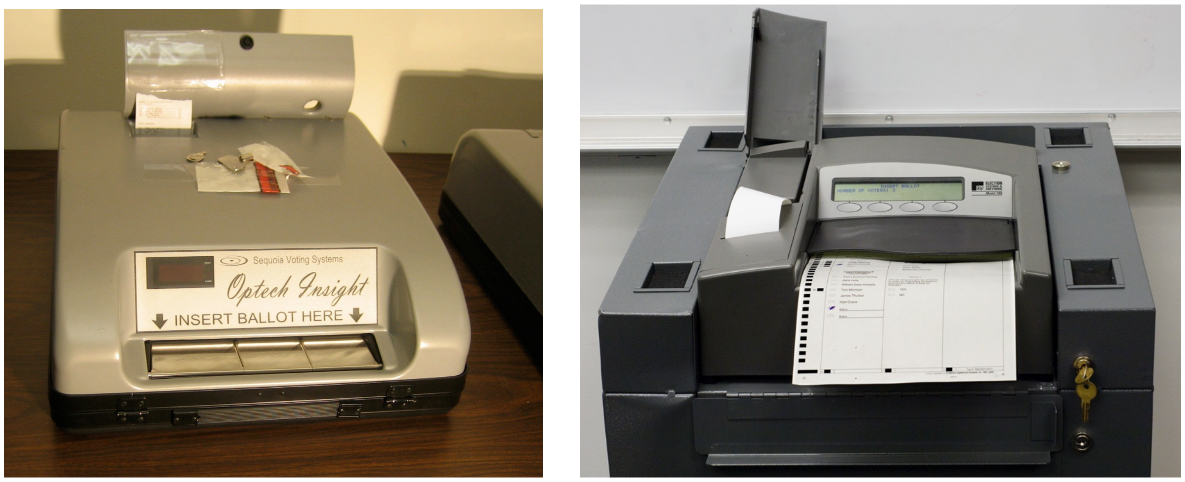


Fig. 2. The Optical Scanners from Sequoia (left) and ES&S (right).

overvote (voting for more than n choices when n is the maximum number that can be marked). If this is the case, the voter is given the opportunity to correct his/her vote.

After the election is closed, the results from each of the DREs and scanners at a polling place are collected on a DTD and returned to election central, where they are read into the election management system to produce a tally for the entire area.

Note that what we have described is a simplified, abstract version of the voting process that reflects how the evaluated voting systems are used in US-based elections. These systems, or some of their components, might be used in different ways in other countries. Nonetheless, we believe that this process is representative of how these voting systems are used in most elections.

4.2 How voting systems differ from other systems

Electronic voting systems differ from other types of systems in a number of ways. One important difference is that their results are hidden from the user. For instance, if one is interacting with an ATM, the user has the cash disbursed by the machine along with the receipt to verify that the transaction occurred correctly. With an electronic voting machine, the most that a voter receives is a receipt indicating that he/she voted. Of course, the receipt never records a vote, because if the voter were to receive such a receipt indicating who was voted for, then votes could be purchased or coerced.

Furthermore, with electronic voting systems, failures are not apparent, because the changes in the overall state of a system as a result of its interaction with the voter (e.g., the total count of the votes associated with a certain candidate), are hidden from the voter. That is, even if the system were not behaving maliciously, the DRE can make mistakes due to configuration problems, such as an inaccurate touch-screen calibration (i.e., the voter touches near the desired icon, but the vote is given to the candidate or party associated with a neighboring icon). This kind of problem can have an enormous effect on the election results. Of course, these errors would be indicated on the tally screen and on the VVPAT, but experiments have shown that these summary screens and the VVPAT are seldom carefully reviewed by voters [52], [53].

In addition to hiding the results of a transaction from the user, the records are both electronically normalized and anonymized. This has two consequences. First, if the

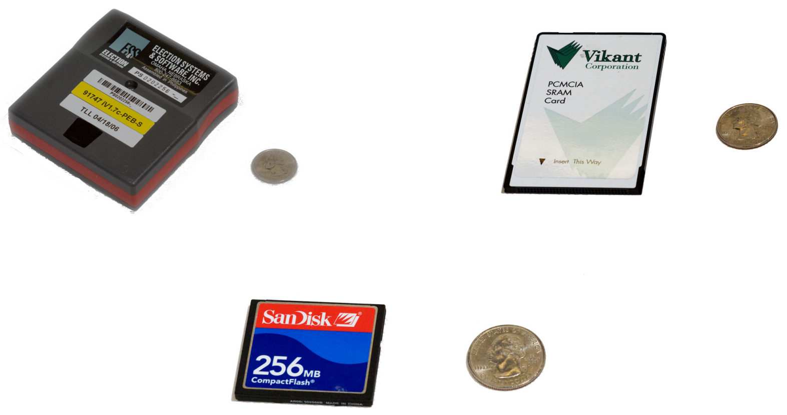


Fig. 3. The Data Transport Devices used in the ES&S voting system. The reference coin is a US quarter, whose diameter is 24.26 millimeters.

records are not properly protected using some form of strong cryptography (used correctly, of course), they can be altered without alerting suspicion beyond statistical figures. Second, electronic records can be forged faster and more easily than physical ones. In the case of voting, a forgery would involve reverse-engineering the results file format and simply writing file(s) as desired, where previous attempts might have involved acquiring a printing press and trying to imitate both the voting form and the nondeterminism of the human hand. The latter is clearly a much more involved task than the former.

To prevent voting systems from being compromised, physical security is of great importance. The systems are locked in warehouses, which often require two-person controls to enter. In addition, every component of a voting system is accompanied by a "chain of custody receipt" in order to generate an audit trail of who had access to which components at what time. Unfortunately, voting systems are often delivered to polling places a week or more ahead of election day, and in the interim are often stored in insecure environments, such as a school gym or an election official's garage. Such practices represent an obvious "weak link" in the chain of custody.

In addition to physical security, much of the security of the voting process is dependent on the poll workers following explicit procedures. Unfortunately, most of the personnel that are required to carry out these procedures have very limited IT training and are not capable of dealing with problems that could arise when using electronic voting systems. The solution to this problem often is to have a representative from the voting system vendor on site (or at least on call) to deal with IT problems that may arise.

Since voting machines are so critical to our democracy, there is a strong desire to assure that they perform correctly. Currently the assurance of these systems involves a lengthy certification process. This certification process is a double-edged sword. Because it takes so long for (re)certification, vendors are often slow to apply patches to their systems. The result is that vulnerabilities are not fixed as soon as they should be, and vulnerable systems are widely deployed. In a testimony before the U.S. House of Representatives [54], Wagner presented a number of problems with the certification process: (i) there is a conflict of interest, because the required certification process performed by testing authorities is paid for by the vendors; (ii) there is a lack of transparency, since the reports are generally not publicly available; (iii) certification does not include the testing and enforcing of required standards; and (iv) there is the lack of a clear decertification path for systems that fail testing.

In addition, the certification standards have their own set of problems. They lack a clear system and threat model, they often propose seemingly arbitrary specifications, they sometimes mandate impossible features, and they cannot easily keep up with the appearance of new security threats, as the definition of certification standards is a time-consuming, lengthy process [55].

4.3 Security testing of voting systems

Security testing is generally an overlooked and under-appreciated part of the electronic voting machine testing process as a whole. The reason for this deficiency stems from many factors. First, the majority of software developers are not security experts, or even security-aware. This leads to software with "bolted-on" security, poorly implemented security, or no security whatsoever. Second, software testing engineers and the organizations that employ them are concerned with proper execution in response to use cases and the advertised functionality of a product. Exceptional and hostile environments are usually not considered in the testing process, even though there is a substantial number of publications on security testing [56], [57]; therefore, security holes are not discovered. Finally, the security of large systems with many developers is often hard to assess, because it requires knowledgeable individuals who are able to understand how one could leverage the complex interactions between the components to bring the system into an unintended and vulnerable state.

The aforementioned characteristics of voting systems imply three important consequences that necessitate proper design and security evaluation. First, the presence of sensitive election information makes the threat of well-funded and motivated attackers a real concern. Second, the distributed nature, complex design, and reliance on proper execution of operational procedures all serve to create a wide and varied attack surface. Finally, the public's relation to voting systems, both in their use and in their effect on the future direction of society, makes public perception and confidence of primary importance when testing these systems. All of these factors, combined with historical implementation issues, set the testing requirements for electronic voting systems apart from the testing of other systems.

In order to ensure the public's confidence in a voting system, a rigorous, objective, and publicly accessible test procedure and report must be developed. Reassurance by the vendor is necessary, but cannot be considered sufficient in this case. Instead, the system of checks and balances that is important in any public arena should also be applied here. It has to be noted that often the public has been lulled into a false perception of the security characteristics of electronic voting system by the vendors of the systems and the administration that used taxpayers' money to acquire them.

Although the certification process can help validate the proper functioning of a voting system under ideal conditions, real-world deployments often rely on operational procedures for this assurance. These operational procedures, however, cannot be the only measure guaranteeing security. Instead, there should be safeguards built-in at both the software and physical layers for cases in which these procedures are not carried out correctly or in good faith. For these reasons, the relationship between the operational procedures, their effect on information flow, and the overall security of the system must be carefully analyzed.

One of the biggest frustrations to the potential testing of current electronic voting systems is that they use both proprietary hardware and software. In many cases, being proprietary makes obtaining source code, documentation, and build environments very hard. In addition, the technologies that are used can be very old and outdated, making the reproduction of a suitable test environment nearly impossible. For these reasons, the resources that are available to a potential objective tester are usually severely constrained, enabling only black-box testing where white-box or gray-box testing is appropriate.

In the two voting system evaluations in which we were involved, we tested specific configurations and versions of the components of the systems. In the case of the Sequoia voting system, we tested the WinEDS EMS (version 3.1.012), the AVC Edge Model I DRE (firmware version 5.0.24), the AVC Edge Model II DRE (firmware version 5.0.24), the Optech 400-C/WinETP optical scanner (firmware version 1.12.4), the Optech Insight optical scanner (APX K2.10, HPX K1.42), the Optech Insight Plus optical scanner (APX K2.10, HPX K1.42), the Card Activator (version 5.0.21), the HAAT Model 50 card activator (version 1.0.69L), and the Memory Pack Reader (firmware version 2.15). In the case of the ES&S voting system, we tested the Unity EMS (version 3.0.1.1), the iVotronic DRE (firmware versions 9.1.6.2 and 9.1.6.4), the Model 100 optical scanner (firmware 5.2.1.0), and the Model 650 optical scanner (firmware 2.1.0.0).

5 Methodology

In this section, we present a two-tiered testing methodology that can help security engineers in designing experiments to evaluate the security of an electronic voting system. Steps 1 and 2 of the methodology deal with an abstract, high-level view of the voting system. Steps 3 through 6 deal with the low-level actual implementation. There is also a preparation for testing step, Step 0. The overall approach focuses on finding software bugs and design errors that lead to vulnerabilities that can potentially be exploited by an attacker to violate the integrity, confidentiality, and availability of the voting process.

Step 0. Information gathering

This step is actually preparation for the testing process. It consists of collecting all the available information on the system under test and preparing the environment in which the testing will be performed. In particular, it is important to obtain the following resources:

- A copy of each of the components that are part of the voting system. Even though some previous analyses of electronic voting systems [2], [22], [24] were based solely on the source code, the availability of the actual hardware greatly increases one's confidence in the results and allows the tester to actually implement and verify the effectiveness of each attack.
- A copy of both the source code and binaries for each software component installed on the voting machines. This is not strictly required in order to test the voting system, but it can help to reduce the amount of reverse engineering required and simplify the vulnerability analysis.
- A copy of all the available documentation (e.g., software user manuals, hardware schematics, and descriptions of the voting procedures) and the results of past testing experiments (if any) performed by other teams on the same voting system. Many of these documents are publicly available on the Internet.

- Vendor support in terms of the training required to properly operate each hardware or software component. In addition, a step-by-step example of a complete election process can be very useful to quickly understand all the involved procedures and the interaction between the different components. This information can be extracted from the documentation or from the analysis of each module. However, the involvement of the vendor can greatly simplify this task.

In our experiments (and therefore in the rest of the paper), we assume that the testers have full access to all of the aforementioned resources. It is important to note that even though this access can greatly improve the quality of the testing, previous studies have shown that an attacker can successfully find exploitable vulnerabilities with very limited access to the hardware/software infrastructure.

Step 1. Identification and analysis of the high-level components and information flow

This step identifies the high-level flow of information in the election process and any assumptions that are made. The first thing to do is to identify the different components used in the election process. These are not the actual pieces of hardware, but are the abstract components, such as an election management system (EMS), a DRE, or a data transport device. After all of the high-level components are enumerated, it is necessary to identify what information, such as a "ballot," is generated, transported, or used by each component. For instance, the EMS generates the ballot, and the DTDs transport the ballot from the EMS to the DRE or to an optical scanner.

Once all the details of each component have been collected, the tester can draw a global picture representing the interaction and the information flow between the devices involved in the voting process. Before starting to look for vulnerabilities, it is important to add one last piece to the puzzle: the voting procedures. Voting procedures are a set of rules and best practices that regulate how a real election must be executed. For example, they describe who is in charge of each operation, who is going to operate the voting devices, and how the devices will be operated.

Taking into account the procedures is very important in designing realistic attack scenarios. However, it is also very important to remember that a procedure cannot be the only defense mechanism against an attacker. For example, if there is a button on the side of an electronic voting machine to reset the system, assuming that during the election a poll worker can check that nobody presses that button is not a solution to the problem.

Next, it is necessary to list the assumptions that are made about the confidentiality, integrity, and availability of this information. For instance, it is assumed that the ballot information loaded on the DTD is identical to what was generated by the EMS. Furthermore, it is assumed that the information on the DTD (e.g., ballots or voting results) is unaltered during transport.

Step 2. Develop misuse cases for violating the assumptions

It is important to devise experiments to test the cases in which some of the assumptions (procedural and otherwise) are violated, intentionally or not. This step constructs misuse cases where if these assumptions are not true the secrecy, integrity, or availability of the election could be jeopardized. For each misuse case the assumption that is violated and the resulting failure are identified. Note that at this step we do not identify the actual attack, or the actor that carries out the threat. An example of a misuse case is violating the assumption that DTDs cannot be altered during transport without being detected. This could result in invalid ballot information or invalid election results.

Step 3. System analysis and identification of the low-level information flow

The goal of this phase is to model the input/output interface of each hardware and software component. First of all, it is important to inspect the hardware and list every input/output channel such as serial ports, memory card slots, or wireless interfaces. For example, even though a DRE is usually not equipped with a keyboard, opening its case can reveal an internal keyboard port that can be very useful for debugging and testing.

The best way to reconstruct the information flow between the different components is through a precise analysis of the source code. However, in order to avoid problems in the rest of the experiments, it is a good practice to initially verify that the source code obtained in step 0 corresponds to the actual software installed on the various machines. Unfortunately, the use of proprietary (or no longer available) build environments can complicate this operation, sometimes making a precise verification impossible.

During this phase, the testers must precisely identify what data is exchanged between the different components, what protocol and data format is used in the communication, and which physical medium carries the information (e.g., an Ethernet cable, a phone line, or a compact flash card). This step begins by identifying the low-level flow(s) that implement each of the high-level flows in step 1. If there is a high-level flow that is not matched in the low-level, then it is necessary to determine whether the high-level flow is erroneous. If it is not, then the missing low-level flow(s) must be identified. An example of an erroneous high-level flow might be where a VVPAT was identified in the high-level, but none was used in the actual implementation. In this case, the high-level flow of information from the DRE to the VVPAT would be erroneous. If there is a low-level information flow identified in this step that was not identified in step 1, then the flow could lead to a way to covertly leak information.

It is also important to understand how each component authenticates and validates the data it receives and how the information is protected from external analysis, eavesdropping, man-in-the-middle attacks, tampering, and replay attacks. For example, it may be possible for an attacker to use the same credentials to vote twice or to sniff the communication containing the voting results.

If the data is encrypted, it is important to understand the way in which the encryption key has been shared between the sender and the receiver. For example, if the key used to encrypt the data on a DTD is transmitted on the same medium, the security of the communication can be easily compromised. That is, an attacker could alter the information on the DTD and produce a new integrity checksum by using the key stored on the DTD. Thus, the alteration would go undetected.

Finally, the analysis of the source code can reveal other valuable information, such as undocumented features or information flows, or the presence of debug functionalities that can be exploited to subvert the voting system.

Step 4. Identification of threats and attack exposures

At this stage it is important to define a precise threat model, which is a model of the possible attackers, their motivations, capabilities, and goals. For instance, an attacker can be interested in deleting or altering the results, in preventing other people from voting, or in discovering the identity of previous voters. Categorizing the attackers is also very important. Given the critical tasks performed by these devices and the amount of money involved in a real election, insiders, as well as outsiders (e.g., regular voters), can be interested in attacking the system. Poll workers and election officials can be bribed, or they may have personal interests in affecting the results of an election. Even malicious vendor employees must be taken into consideration, especially given their access to the low-level voting infrastructure.

To identify actual attack scenarios one considers each of the assumption violations of step 2 that yielded an undesirable result. For each of these, the low-level information is analyzed to see if an attack scenario can be identified. More precisely, the system analysis of step 3 describes how the actual voting system works, the operational procedures in step 1 describe how humans are supposed to interact with the system, and the threat model describes the possible goals and resources of various attacker categories. Combining these three pieces of information allows the security engineer to identify possible attack scenarios. For instance, consider a corrupt election official whose goal is to change the results reported for his/her precinct. The operational procedures of step 1 show that the official has access to the DTD results cartridge. Furthermore, a misuse case from step 2 reveals that being able to alter a DTD in an undetectable manner allows one to alter the reported results. Finally, the details of step 3 reveal that the key for encrypting the data on the DTD is stored on the device. With this information the security tester can develop a scenario where the corrupt election official alters the DTD in an undetectable manner to change the results reported for his/her precinct.

One can visualize the voting process as a chain of trust and information that links together all the machines and the people involved in the voting system. At the beginning, the election officials prepare the ballot definition. The definition is saved into some devices that are then used to initialize the electronic voting machines. At the end of the election, the votes stored in the machines are collected and sent back to the election management system to be tallied.

This process has a cyclic structure (See Figure 4), where the input of a step in the process is the output of the previous step. Enumerating all the attack scenarios means enumerating all the possible ways in which an attacker can compromise a component involved in the process and break the cycle.

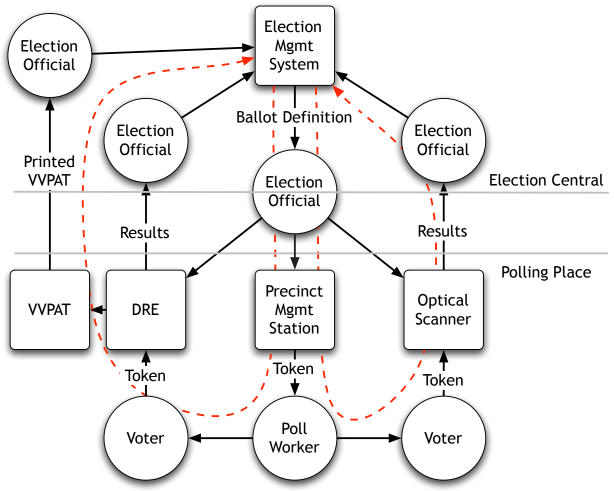


Fig. 4. Graphic description of the cyclical information flow among the components of the voting system.

Step 5. Breaking the cycle: attacking a component of the voting process

The objective of this phase is twofold. First, the tester must perform a vulnerability analysis to identify any software bug or system design error that can lead to a vulnerability that can be exploited to realize one of the attack scenarios that has been identified in the previous step.

When a vulnerability is discovered in one of the components, it is necessary to develop an attack that successfully exploits the vulnerability. Compared with other security testing experiments, this task presents interesting and novel difficulties. First, due to the intrinsic characteristics of the voting environment, it is often necessary to develop a number of *ad hoc* tools in order to interact with the voting devices.

Second, the stealthiness of the attack can be a very important point. Even though a simple exploit that crashes a DRE can be an effective denial of service attack, more advanced attacks that aim at affecting the results of the election need to go unnoticed. This is particularly difficult, because most election systems are designed to identify and audit any error and suspicious condition; often, they rely on a Voter-Verified Paper Audit Trail, which can be very difficult to modify.

Examples of tools and techniques that can be used to circumvent these limitations are presented in Section 6.

Step 6. Closing the cycle: compromising the entire voting system

Sometimes the usefulness of individual attack scenarios are not realized until vulnerabilities of multiple components are known. For instance, the full advantage of being able to modify ballot information on a DTD may not be evident until it is known that there is a buffer overflow in the DRE routine that reads the ballot header.

In the previous step, the testers develop attacks that can be used to compromise a single component of the voting system. In this last phase, the focus shifts from the single component to the entire voting system. In particular, it is now important to evaluate how a compromised component can take advantage of the legitimate information flow to take control of other devices, with the goal of eventually controlling the entire voting infrastructure.

The idea is to use a combination of the attacks developed in the previous step to inject a virus-like malicious software that is programmed to automatically spread to as many voting machines as possible[2]. This can be achieved by copying the virus onto media devices (DTDs) that are later inserted into other components where the malicious data can trigger a local vulnerability. If the virus can reach and infect election central (where components for all of the precincts are initialized and the votes are tallied), the entire voting process can be compromised.

Summary

As mentioned earlier, we believe that our testing methodology can be adapted and applied even when the settings of the testing are different than the projects reported in this paper. In these cases, it will not be possible to complete some of the steps we presented (e.g., testing the interactions of multiple components of the voting system will not be possible if only one is available) or it will be necessary to use different techniques to perform some of the testing (e.g., if the documentation of a component is not provided, reverse engineering may have to be used to understand how the system works).

2. Hereinafter, we refer to the malicious software we have developed as a virus or malware. We describe the behavior of the malicious software as "virus-like" because it spreads from machine to machine.

6 Tools and Techniques

In this section, we describe some of the techniques and tools we used to apply our testing methodology. Because electronic voting systems are implemented using specialized hardware, custom tools are often needed. In particular, it is usually necessary to develop three sets of tools: tools to extract and replace the voting machine's firmware, tools to support the development and testing of exploit payloads, and tools to read and write the voting machine's data transport devices.

How these tools are actually implemented depends on the type of firmware the voting machine utilizes. Therefore, we first review the different firmware types and discuss how they influence the testing process.

6.1 Types of voting machine firmware

The firmware of the voting machines we analyzed can be classified into three different types, based on the amount of COTS components they utilize. The first group of voting system firmware utilizes a COTS operating system and all voting-specific code is run as processes within the operating system. The second class of firmware utilizes a COTS BIOS. In this case, the voting system firmware includes functionality normally performed by the operating system, but utilizes the BIOS for most I/O operations and boot-time initialization. The third class of voting system firmware does not rely on any third-party components. This type of voting system firmware runs completely stand-alone.

Depending on the class of firmware, the type of analysis tools needed for the evaluation differs. For systems utilizing a COTS operating system, the operating system tools and services can be leveraged to perform the analysis. For instance, most operating systems include tools to perform file operations (e.g., a command shell) that can be leveraged in order to replace the voting-system-specific code. In addition, many operating systems include functionality to support the debugging of user-level processes. This debugging functionality is very useful when crafting exploits. Finally, an OS provides process isolation; therefore, if an attack causes the voting system process to crash, the operating system would keep running and allow for uninterrupted debugging support.

Systems that rely on a COTS BIOS but do not run in an operating system require radically different analysis methodologies. This class of voting system firmware does not include all the services normally provided by an operating system. For instance, none of the systems we analyzed contained functionality for attaching a debugger to the voting application process or for manipulating files. Another challenge with these systems is that they have very limited I/O capabilities. A common debugging technique is to create a debug program trace by printing to the console as the program execution progresses. This technique is complicated by the fact that none of the voting machines has a built-in console that could be printed to. In addition, the voting systems of this type that we analyzed were designed so that if the process running the voting system software crashes, then the whole system halts. This complicated the process of doing a post-mortem analysis of failed exploits.

Voting systems that are completely stand-alone have all the challenges of OS-free voting systems, in addition to some specific challenges that the lack of a BIOS causes. One of the main tasks of the BIOS is to facilitate the boot process. The normal boot sequence of a system with a BIOS begins with the processor jumping to a specific address within the BIOS where it starts executing. The BIOS initializes some of the hardware and loads the boot block from the boot drive. The boot block in turn loads the operating system from disk. The operating system is often contained in a regular file on the boot file system. In a BIOS-free system, however, the boot process works differently. The processor starts executing at a specific address after a reset. Since there is no BIOS in the system, the voting system code must be located where the BIOS would be in a COTS system. This means that the voting system code cannot be stored in a file on a file system, but, instead, has to be stored on a ROM or EPROM chip. This fact complicates the analysis. In a BIOS-based system, it is easy to read and replace the voting system firmware since it is located in a regular file on a flash card and hardware adapters to access flash cards are readily available. The BIOS-free systems we analyzed all required specialized and less available hardware in order to read and write the EPROM chips. In addition, in one of the systems we analyzed, the EPROM was soldered on the board and it could not be removed without unsoldering it.

6.2 Firmware reader/writer tools

For the voting machines that had firmware stored on an EPROM chip, we needed a way to read and modify the contents of the chip. While commercial EPROM readers are readily available, they did not suit our needs. EPROM readers require the chip to be extracted from the circuit and inserted into the reader. The voting machines we analyzed had the chips soldered onto a circuit board, and removing them would have been cumbersome and could have caused damage. Fortunately, the processors used in the voting machines with ROM chips all had built-in JTAG [58] support, which could be leveraged to access the EPROM chips. JTAG is a hardware debugging interface. Among other things, it allows the tester to completely bypass the processor logic and control the logic state of the processor's pins directly. By changing the logic state of the processor's pins in a carefully controlled pattern, the EPROM can be accessed through the JTAG port. Unfortunately, we could not find an affordable JTAG tool that supported the particular processor used by the voting device. We ended up extending an open source JTAG tool designed for ARM processors (OpenOCD [59]) to work with the voting machine's processor.

6.3 Exploitation support tools

Developing exploits for a voting machine is also a difficult and error-prone task. Each attack must be accurately tested to tune the exploits with the right memory values. Unfortunately, the adoption of proprietary firmware and the lack of appropriate emulator platforms forced us to perform all the tests directly on the real voting machines. To make this operation feasible, we first had to add debugging capabilities to the DREs. In addition, once the exploit is ready, an appropriate payload must be developed. To simplify this operation, we developed a framework that allows the attacker to inject malicious code in the voting machine by automatically patching its firmware. The details of the debugger and the patching framework are presented in the following paragraphs.

6.3.1 Debugger

One of the most important tools needed to write a functioning exploit is a debugger. The debugger allows the tester to inspect the memory contents of the voting machine, set breakpoints, and single step through the sections of code that contain vulnerabilities. Since none of the DREs we analyzed had built-in support for debugging, we had to add this functionality. We chose to implement GNU debugger (GDB) support over a serial line. This technique allows the tester to attach a debugging computer to the voting machine using a serial cable. The debugging computer runs the GDB application and provides the tester with full debugging support of the target voting machine. In order to implement this functionality, a debugging stub has to be installed on the voting machine.

We were not able to compile the firmware of any of the voting machines we tested because we were not provided with a functional build system. This forced us to binary-patch in the debugging stub. The debugging stub we used was based on a stub shipped with GDB. We modified the stub in order to make it self-contained, since we could not rely on operating system services to access the serial port. After compiling the stub, we copied the binary stub to an unused area of the voting machine's ROM. For the stub to work, it needs to be hooked into the voting machine's interrupt table. We located the interrupt table of the voting machine by disassembling the binary. The interrupt table can easily be found by looking for the assembly instruction used to load the interrupt table. After identifying the location of the interrupt table, we modified the table to point at our debugging stub. By performing these modifications, the code running on the voting machine could be debugged.

6.3.2 Firmware patching framework

After identifying a vulnerability and creating a working exploit, the next step was to modify the firmware and cause it to behave in a malicious way. Since we were not able to compile the source code, we could not just modify the source and compile a malicious firmware version.

Instead, we had to modify the firmware by binary-patching in the new functionality. Manually performing the binary patching can be time-consuming and error-prone. Therefore, we designed a patching framework that allowed us to write extensions to the firmware in C and link these extensions to the original firmware. Two types of linking were needed. First, the extensions needed to be able to call functions in the original firmware. Second, the extensions needed to be able to hook themselves into the original firmware so that the original firmware would call the extensions at an arbitrary location. The framework consisted of two jump tables and a patching script. The two jump tables processed the two kinds of links and allowed the extension to call functions in the original firmware normally. The patching script concatenated the original firmware and the extensions and created a new binary. In addition, the patching script overwrote interesting function calls in the original firmware and diverted the function calls to the extension.

6.4 DTD manipulation tools

Most voting machines we analyzed utilized some kind of data transport device or hardware token for access control, transferring initialization data to the voting machines, and moving the voting results back to the EMS.

The DTDs stored a sizable amount of data that was read into the voting machine during the authentication and initialization processes. Since the voting machine was reading data from these tokens, they represented an interesting attack vector. In order to explore this vector, we developed tools to perform low-level reads and writes of the data contained in the DTDs, which allowed us to create tokens that contained illegal or unusual data. The DTD and M100 filesystem reader/writers are discussed in the following paragraphs.

6.4.1 DTD reader/writer

We developed a number of tools to extract and parse the information contained in various DTDs. Our tools were also able to write blocks of data back to the transport devices, setting all of the headers and checksum values appropriately. Sometimes, as in the case of ES&S personalized electronic ballot (PEB), the data was stored in encrypted format but the decryption key was also stored inside the device itself. In this case our reader/writer tool was able to retrieve the key and to use it to decrypt the information contained inside the device and encrypt our modifications.

By leveraging these basic operations, our tools allowed us to dump the contents of a DTD and to create valid DTDs containing arbitrary data.

6.4.2 M100 filesystem reader/modifier

Sometimes, being able to read and write the contents of a DTD was not enough. For example, the ES&S M100 optical ballot scanner allows the firmware to be updated from a PCMCIA card. While reading from and writing to the card was not too complicated, extracting and forging the firmware content was problematic. In fact, the firmware consisted of a filesystem containing a set of hardware drivers, startup scripts, and the main scanner application. Unfortunately the filesystem, a proprietary QNX filesystem specifically made for flash memory, was created with an old version of the QNX development tools, which is no longer supported by the QNX operating system.

Since we could not find any tools or any description of the filesystem specifications, we had to reverse-engineer its format. The fact that the content of the filesystem was compressed in order to save memory further complicated the reverse-engineering process.

We finally created a tool that was able to extract and decompress all the files contained inside a firmware image. Even though the tool did not support the generation of a new filesystem from scratch, it was able to re-compress a file and put it back in an existing firmware image. This was enough to allow us to extract and modify the main optical scanner application.

7 Findings

We performed a security evaluation of the Sequoia voting system as a part of TTBR project for the state of California and the ES&S voting system as a part of EVEREST project for the state of Ohio. Each voting system was currently certified for use in the corresponding state. The exact versions of the reviewed systems and their components can be found in the public reports of the studies [5], [6].

Our security evaluations of both the Sequoia and ES&S voting systems resulted in the discovery of a number of previously-unknown vulnerabilities. Some of the vulnerabilities found were specific to a particular system or a component, and others were common to both systems. More importantly, vulnerabilities discovered in both systems often resulted from serious design flaws and apparent lack of security awareness of system developers. For example, we found that important security mechanisms, such as cryptography, were almost never used correctly (if used at all) and well-known security practices, such as avoidance of the usage of unsafe string handling functions, were often ignored. These findings lead us to conclude that both evaluated voting systems are poorly designed, fundamentally insecure, and have a potential to contain more exploitable vulnerabilities than what was found during the time-bounded studies of the systems that we participated in.

In general, vulnerabilities that we found during the studies are new in the context of the evaluated voting systems, but, nevertheless, they all belong to well-known, in the system security world, classes of vulnerabilities. The fact that similar types of vulnerabilities were also found in other voting systems evaluated either in previous studies (described in Section 3) or as a part of

TTBR and EVEREST studies [60], [61] inevitably leads to the conclusion that all currently used voting systems are insecure. The fact that the majority of the discovered vulnerabilities could be easily avoided if the systems were carefully designed and implemented from the beginning shows that voting system vendors are more interested in releasing profitable rather than reliable and secure products. Finally, the fact that such systems are certified and used throughout the country clearly shows that the voting system certification process is seriously flawed. Thus, our findings once again confirm the concern that the current state of voting technology fails to guarantee fair elections.

In the following sections, we will describe a representative sample of the previously-unknown vulnerabilities that we found.

7.1 EMS vulnerabilities

Tests of both vendors' election management systems (EMS) revealed numerous vulnerabilities. Perhaps the most troubling finding was the presence of exploitable software defects allowing the execution of arbitrary code of an attacker's choosing. For instance, buffer overflows were present throughout the source code for the ES&S EMS, indicating a pervasive ignorance or dismissal of basic security awareness and defensive programming techniques. In one case, we developed a working exploit for a buffer overflow in ES&S election results processing code that allows an attacker to gain full control of the EMS when election, pre-election, or testing results are processed. To exploit this vulnerability, an attacker needs to be able to modify a DTD that is used to transfer election results from a DRE. One way to achieve that is described in Section 8.2. A successful attack can go completely undetected.

Another area of significant concern was the general lack or misuse of cryptographic techniques to authenticate the origin of data processed by the voting system or to ensure the integrity of critical election data. For instance, asymmetric cryptography was completely eschewed in favor of secret keys, which in many cases were hard-coded into a component's software with no apparent strategy for key revocation in the event of a compromise. Additionally, although election data was in some cases protected by a checksum, these were easily forged and, invariably, no cryptographically-strong signing mechanism was used. These oversights allow an attacker, for instance, to forge authentication tokens and election results, in some cases for entire precincts.

A third area in which vulnerabilities were found is that of incomplete specification of system requirements and misconfiguration of system environments. Both vendors support the option of deploying their EMS on customer-provided hardware; in this case, however, documentation relating to proper system configuration and security hardening is often misleading or incomplete, resulting in potentially serious vulnerabilities. For instance, the Sequoia EMS was configured with the Windows "autorun" feature enabled for removable media, which allows an attacker to compromise the machine via the simple insertion of a flash drive or CD-ROM. The EMS also allowed remote users to not only connect to its back-end database as the database administrator, but also allowed remote users to execute arbitrary commands using database extensions that could have been easily disabled. The ES&S EMS, as another example, was shipped with a version of the Java Runtime Environment that includes a known vulnerability in its image processing code. Clearly, a coherent, pellucidly articulated set of configuration procedures and system requirements would largely mitigate such vulnerabilities.

Finally, none of the systems had sufficient access control mechanisms implemented to protect critical election data from unauthorized access. As a result, anyone (not necessarily an election official) with physical or remote access to the EMS, could potentially view or modify election data [3]. In most cases, user-level security was expected to be implemented by the host operating system. Ironically, the Sequoia system documentation states that the recommended OS choices included Windows 98 and Me, which have no user-level security. In the ES&S system, some (but not all) EMS components could optionally be configured to be protected by an authentication and auditing module, which turned out to be vulnerable to a simple SQL injection attack.

To summarize, we found that the security of the election management systems, which are used to store and process the election data for one or more precincts, mostly relied on the assumption that only a limited number of trusted officials can have physical access to them and that a set of predefined procedures for each EMS is strictly followed by the officials. Clearly, both assumptions can easily be violated, intentionally or not, leaving election data virtually unprotected.

7.2 DRE vulnerabilities

In our evaluations, the vendors' respective DRE products suffered from classes of vulnerabilities similar to those found in the election management systems. Both DREs contained multiple buffer overflows in their handling of election data, and working exploits were developed for overflows in the ballot-loading code for each machine. Generally speaking, each of the vendors' source code bases were written without regard to modern security engineering practices, such as avoiding the usage of unsafe string handling functions or performing rigorous input validation checks.

An important characteristic of the vulnerabilities that we successfully exploited on both systems is that they could be automatically and silently exploited during normal election operations and resulted in complete system

3. In general, to protect election-critical data, counties are expected to follow a set of procedures that limit access to the EMS to authorized election officials only. However, if these procedures are violated, intentionally or not, the system can be easily accessed by an unauthorized person.

compromise. For example, we found that it was possible to take complete control of the ES&S DRE by creating a specially-crafted DTD and exploiting a buffer overflow in the poll-opening process. In the Sequoia system, the DRE's firmware could be silently overwritten with code of an attacker's choosing by constructing a malicious version of the DTD that was used to transfer an election definition from the EMS to a DRE. Perhaps even more troubling than the existence of buffer overflows, we found that both systems' DREs did not have sufficient means to detect a firmware replacement or modification. Clearly, hardware support for trusted software execution and the use of non-writable memory would be able to mitigate a large range of similar attacks.

The design of both DREs also exhibited the same ignorance or misapplication of cryptography as in the case of the EMS, with similar implications. During our evaluations, it was trivial for an attacker to forge authentication tokens as well as modify or simply create election data. For example, it was possible to forge the voter cards used by Sequoia DREs when a static key used for their content encryption was recovered from the binary image of the DRE firmware. More interestingly, on both systems, critical security tokens, such as an encryption/decryption key, a password, or an identification number, were often stored on the DTDs themselves and were completely trusted by other system components without performing any cross-checks. For example, in the ES&S system, an election key, which was used by all components of the system as a unique identifier, was stored on a DTD in an encrypted form. However, the key used to encrypt the election key was also stored on the same DTD in unencrypted form. On both systems, the critical lack of cryptographic protection allows a malicious person to impersonate an election official or vendor technician, vote multiple times (for example, by using multiple forged authentication tokens), perform unauthorized reconfiguration, or introduce exploits into the system. Some of these attacks, for example, the ones that change vote counts, can potentially be detected if data (such as the number of registered voters and the number of votes collected at a precinct) originating from different sources is compared. It is our understanding, however, that even if data discrepancies are detected, it is not possible to tell which collected votes are the legitimate ones and which are not. Thus, even if such an attack is detected, it should be considered an effective attack that will affect the outcome of an election.

A particularly disquieting finding was the presence, in both products, of backdoors or expressly-prohibited features in the source code. In the case of the Sequoia DRE, its firmware contained a full-fledged interpreter for a scripting language, which allowed a user to set the "tamper-proof" protective counter of the machine, set the machine's serial number, overwrite arbitrary files (including election data, the firmware, or audit log) on the internal compact flash drive, and reboot the machine.[4] The source code for the ES&S DRE likewise recognized special "initialization" and "factory" authentication tokens that allow one to, for instance, reset or bypass system passwords and erase election and audit data.

Finally, contrary to the claims of the vendors, the physical seals protecting access to critical components of the DRE were, in almost all cases, not tamper-proof or even tamper-evident. In many cases, the seals could be removed without evidence or bypassed altogether by simply removing a small number of screws and disassembling the chassis of the DRE. The lack of physical security allows an attacker to access sensitive poll worker controls and I/O ports during an election, or to directly access the system firmware, election data, and audit logs.

7.3 Optical scanner vulnerabilities

Evaluations of the various optical scanners offered by both vendors followed much the same pattern as the previous voting system components. A patent disregard for cryptographic authentication and integrity checks allows attackers to overwrite a system's firmware with malicious versions and modify or construct election data to be processed by an EMS. For example, in our evaluation, the firmware update procedure for the ES&S M100 optical scanner did not require a password and the option to upload a new firmware was given to the user as soon as a suitable DTD with a correctly formatted firmware was inserted. Thus, a sufficiently motivated attacker with knowledge of the M100 hardware and physical access to the machine could easily install a new firmware on the optical scanner. Moreover, due to the poor system design, a new malicious firmware could be unintentionally installed on the scanner by a distracted or inattentive poll worker who was given a malicious DTD. This situation is possible because the procedure for installing new firmware on the M100 is the same as the procedure for loading a new election definition, which is routinely done during each election. Due to the time constraints and lack of local access, we were not able to evaluate the Sequoia optical scanner as thoroughly as we did for ES&S. However, the similarity of problems found in other systems components allows us to suggest that similar problems can exist in the Sequoia optical scanners.

Physical security measures were lacking in both systems. In particular, the ES&S scanner lock was easily picked with a paper clip during our tests, while the "unpickable" lock on the Sequoia scanner was bypassed by removing a few screws and pulling out the lock cylinder from the scanner's chassis by hand. In both cases, an attacker was able to access the machine internals to potentially execute arbitrary code.

4. "*Self-modifying, dynamically loaded or interpreted code is prohibited, except under the security provisions outlined in section 6.4.e*" [62, Sec. 4.2.2]

8 Attacks

The vulnerabilities that pervade each vendor's voting system allow a multitude of serious attacks to be executed under several threat models. Taken in isolation, these vulnerabilities constitute a sobering threat to the successful execution of a fair election. Clearly, the ability to run arbitrary code on the election management system of a county, or to directly modify election data from a high-speed optical scanner that processes tens of thousands or more votes during a single election, is a cause for alarm. When these vulnerabilities are considered in the context of the system as a whole, however, even more alarming attack scenarios present themselves. To illustrate this point, in the following sections we discuss a class of attacks that was successfully demonstrated on both vendors' voting systems: a voting system virus. We also show how the virus could be used to steal an election.

8.1 Sequoia virus

In the Sequoia voting system deployment, an attacker first obtains access to a county elections office.[5] The attacker surreptitiously drops a maliciously crafted USB flash drive into the pool of drives used to initialize the smart card programming device. When the malicious drive is inserted into the computer hosting the EMS, Windows autorun automatically executes a Trojan hidden on the drive that contains the virus.

The virus silently installs itself and begins monitoring the host for removable media insertion and removal events. Any flash drive inserted into the EMS is infected with a copy of the virus. In addition, results cartridges are modified to contain an exploit for a buffer overflow in the DRE's ballot-loading code as well as a copy of the virus.

Infected results cartridges are subsequently used to initialize DREs prior to the election. The exploit silently executes during ballot loading and installs a malicious firmware on the DRE. The malicious firmware acts normally during pre-election logic and accuracy testing by taking advantage of existing firmware variables that indicate whether the DRE is being tested.

On election day, the malicious firmware begins to execute various vote-stealing attacks. Discussion of these attacks is deferred to Section 8.4.

After the tallying and reporting process has completed, the virus can remove itself to avoid detection, or it can remain dormant on the EMS host until the next election.

8.2 ES&S virus

In the ES&S voting system deployment, an attacker with access to a DRE loads a malicious firmware containing the virus into the machine, either by exploiting a vulnerability or by directly modifying the on-board flash memory. When a master DTD is inserted into the DRE to initialize it for the election, the malicious firmware installs a copy of the virus on the DTD itself. Subsequent uses of the DTD to initialize other DREs result in those machines being infected through a ballot-loading exploit.

During pre-election logic and accuracy tests, the firmware behaves as expected. During the election, however, the malicious firmware carries out vote-stealing attacks similar to those described in Section 8.4.

After the election has ended, a master DTD is used to collect the votes from each DRE. During this operation, the malicious firmware infects the DTD with a copy of the virus, if it is not already infected. The DTD is then transported by an elections official to the county elections office, where the votes are transferred into the EMS. During this operation, a vulnerability in the EMS is exploited such that the virus is installed in the EMS, allowing for the possibility of further attacks against the election system.

After the election results are compiled and reported, the virus can remove itself or it can remain dormant on the EMS host until the next election. At that time, the virus will infect the master DTD that is programmed to initialize the DREs for that jurisdiction, and the cycle will continue.

8.3 Stealing an election

On election day, malicious software that has been installed by the virus at various points in the voting system becomes active. The primary goal of the malware is to influence the results of the election such that a designated candidate or set of candidates is reported as receiving the highest number of votes. Clearly, this result can be accomplished in a number of ways. At the DRE or optical scanner, a malicious firmware may modify a subset of ballots as they are cast such that they include votes for the preferred candidates. Additionally, it may surreptitiously insert fake ballots with votes for the preferred candidates. It may also attempt to discard, corrupt, or otherwise fail to properly record votes cast for other candidates. Finally, it may simply report a voting summary to the election headquarters that, rather than reflecting the true distribution of votes recorded, reports a false vote distribution favoring the preferred candidates. A similar approach that deliberately miscounts collected ballots can be employed by malware at the central tabulator at election headquarters.[6]

A secondary, but nonetheless vital, goal of the malware is to escape detection by any countermeasures intended to expose election fraud. This goal is important in order to maintain the illusion of legitimacy with respect to the election, as well as to preserve the anonymity of

5. Note that if the attacker is an insider, such as an elections official or maintenance worker, they already have access to the election office.

6. Note that an attack mounted at an aggregation point for precinct-level ballots depends upon the unavailability of intermediate results at each precinct.

the perpetrators. Countermeasures that must be evaded by the malware may include technical safeguards such as cryptographic verification of election software[7] or paper audit trails that can be examined by conscientious voters or election officials. Procedural measures, such as mandatory recounts in particularly close races, should be considered. Also, post-election analysis of the reported results should not reveal significant statistical anomalies that would constitute evidence of fraud.

The following sections outline, in detail, specific attacks we have identified against the various components of electronic voting systems that achieve the first goal of maliciously influencing election results. Additionally, in many cases the attacks we discuss successfully evade, in practice, current countermeasures.

8.4 DRE attacks

Attacks that can be performed by a malicious firmware installed on the DRE can act either upon individual ballots or on aggregate results to be reported to the central tabulator. The following scenarios are of the former type; a general example discussing both how aggregate results at the DRE can be attacked as well as large-scale strategy for how individual ballots at the DRE can be subverted is presented in Section 8.6.

These attacks assume the presence of a physical audit trail, such as a paper tape, that can be inspected by the voter in order to verify the correctness of their ballot.[8] Therefore, each attack attempts to ensure that the physical audit trail is consistent with the number of votes recorded internally by the DRE. Some attacks may also insert additional votes in both the physical audit trail and the electronic results. These attacks are feasible because, in most cases, the election officials cannot determine which votes have been forged and they will have to either accept all the votes or discard the results of the election altogether.

For the following scenarios we assume that the attacker is only interested in changing the votes for one candidate.

8.4.1 Trusting voter

In this scenario the malicious firmware assumes that the voter is "inattentive," in the sense that the voter is not careful to check the ballot review screen or physical audit trail for discrepancies from the intended selections. For this attack the malicious firmware monitors votes cast by a voter. At the time each selection is made, the intended choice is displayed on the screen by the DRE. However, if the voter does not vote for the preferred candidate, then, once the ballot is complete, the malware changes the voter's selection such that the preferred candidate is selected. This modified ballot is then presented for review by the voter on the DRE screen as well as on the physical audit trail. If the voter fails to notice that the ballot summary is incorrect, the malicious ballot is cast, and the malware continues to execute the attack against subsequent voters.

Despite the heavily publicized reputation for electronic voting machine malfunction, several studies have shown that only a small minority of voters actually notice when the ballot summary, either virtual or physical, contains evidence of tampering [52], [53]. Therefore, in practice, this attack would remain undetected with high likelihood. This scenario has the additional advantage that no evidence of malfeasance resulting from the attack can be detected by post-election audits, since the electronic and physical audit trails will match. Also, if the voter does notice the discrepancy, there is no proof that he/she did not just make a mistake.

8.4.2 Careful voter

In contrast to the previous scenario, this attack assumes that the voter is cognizant of the possibility for electronic voting machine malfunction, and will be careful to check the ballot summary before casting the ballot. In this case, a different attack, which has a lower probability of detection, can be executed.

The malware allows the voting process to continue normally until the ballot is cast by the voter. If the voter did not vote for the preferred candidate, then the malware changes the voter's selection such that the preferred candidate is selected. This modified ballot is then presented for review by the voter on the DRE screen as well as on the physical audit trail. If the "careful" voter happens to notice that the displayed ballot summary is incorrect, the malware allows the voter to edit the ballot in order to correct the "mistake." In this case, the ballot is cast in its intended form. Additionally, the malware considers itself detected and, therefore, disables the attack for a specified period of time (or number of voters). As with the trusting voter scenario, the electronic and physical audit trails will match.

8.4.3 Fleeing voter

A relatively common occurrence when using electronic voting systems is that of the "fleeing" voter. Fleeing voters are voters who leave the polling station before completing the ballot. Fleeing voters are handled differently according to the election jurisdiction; regardless, a malicious firmware can take advantage of this circumstance.

In some jurisdictions abandoned ballots are cast by a poll worker.[9] If the fleeing voter has voted for the preferred candidate, then the malware installed on the DRE will do nothing. However, if the vote was against the preferred candidate, then the malware modifies the ballot as necessary to reflect a vote for the preferred candidate. In both cases, the malicious firmware allows

7. Of course, no effective cryptographic verification exists in the electronic voting systems studied.

8. Without such a device, a malicious firmware could modify ballots at will without risking detection.

9. In California, abandoned ballots are treated in this manner.

the poll worker to cast the ballot. Clearly, both the electronic results and physical audit trail will match.

In other jurisdictions, abandoned ballots are discarded by a poll worker.[10] In this case, again two possibilities arise. First, if the ballot does not contain a selection for the preferred candidate, then the malicious firmware allows the poll worker to discard the ballot, thus suppressing a vote for the undesirable candidate. If, however, the ballot contains a vote for the desired candidate, then the malicious firmware will cast the ballot automatically and the poll worker will not be aware that the voter fled.

8.4.4 Fake fleeing voter

In this attack scenario, the malicious firmware artificially induces a fleeing voter situation through careful manipulation of the DRE user interface. The malware allows the voting process to continue normally until the ballot is cast by the voter. However, instead of actually casting the ballot, the malware simply displays a message to that effect. For instance, a screen saying "Thank you for your vote" could be displayed. The malware then waits for a short period of time, during which the voter is assumed to have left the polling station. After the timeout, the malicious firmware proceeds as in the case of a fleeing voter described previously.

8.4.5 After the fact vote

This scenario is similar to the fake fleeing voter scenario except that the ballot is modified and automatically cast after the voter leaves. The malware allows the voting process to continue normally until the ballot is cast by the voter. If the voter voted against the preferred candidate, then instead of actually casting the ballot, the malware simply displays a message indicating that the voting process is over, as in the previous scenario. The malware then waits for a short period of time, during which the voter is assumed to have left the polling station.

After the timeout, the malware modifies the ballot such that the preferred candidate is selected instead. Additionally, the malware modifies the physical audit trail, canceling the intended ballot containing a vote against the preferred candidate. Physically, this process typically consists of printing a message such as "`CANCELED`" after the original ballot. A new ballot is then recorded that indicates a vote in favor of the candidate. Finally, the malware instructs the printer to scroll the audit trail such that the modifications made in the previous step are no longer visible to the voter, should the voter return to the DRE.

Due to the fact that the modification of the ballot is never displayed on the screen of the DRE and the voter has likely left the polling booth when the physical audit trail is modified, this attack is difficult to detect. Additionally, the physical audit trail remains consistent with the electronic results, with only slight abnormalities in appearance that could be attributed to legitimate malfunction.

10. Ohio is an example of such a jurisdiction.

8.4.6 Vote suppression

In contrast to previous attacks, this scenario involves violating the availability of the DRE as opposed to the integrity of the ballot. The general approach is to simulate an electronic voting machine malfunction, for instance by intentionally miscalibrating the touch-screen display or by displaying a standard error message and refusing to accept further input. This attack could be mounted with a low probability when the malicious firmware detects that a majority of voters are not selecting the preferred candidate, thereby suppressing undesirable votes.[11]

8.5 Optical scanner attacks

Similarly to a DRE, attacks performed by a malicious firmware installed on an optical scanner can target either individual ballots or the aggregate results to be processed by the central tabulator. By definition, a physical audit trail exists for optical scanners, consisting of the ballots themselves. Furthermore, optical scanners do not have the ability to modify the ballots directly. Thus, any ballot modification performed by a malicious firmware will necessarily contradict the physical audit trail.

8.6 Large-scale attacks

The attacks presented in the preceding sections primarily focus on how individual ballots can be subverted. While these attacks are designed to evade detection by countermeasures deployed at that scale, such as voter-verifiable audit trails, indiscriminate execution of these attacks would nevertheless be trivially detectable. For example, consider a close race where two candidates polled equally well prior to the election.[12] If malicious software was installed on every DRE and optical scanner deployed in the jurisdiction, and each instance subverted as many ballots as possible, the perpetrators would risk causing a massive discrepancy between the expected outcome (as predicted by pre-election polling) and the true outcome. Even limiting the scope of the attacks to a subset of the precincts might not be sufficient to evade detection, as an increase in votes for the preferred candidates could still raise suspicion if it could not be explained by, for instance, pre-election polling trends or demographics. On the other hand, fixing the attack rate to a lower value runs the risk of not achieving the desired outcome. Finally, manually tuning the attack rate for every DRE and optical scanner in a jurisdiction is time-consuming and prone to error.

11. Voting machines are often provisioned for each precinct based upon an expected number of voters. Therefore, the disabling of even a small number of machines can be effective in denying the ability to cast a vote to a large number of voters.

12. Incidentally, it is this very scenario where election fraud is most effective and, therefore, most likely to be performed.

These considerations are also applicable to attacks that can be executed against either the results gathered from each DRE and optical scanner. Therefore, it would be necessary to formulate a large-scale strategy that achieves the desired goal of forcing the preferred set of candidates to win, while simultaneously minimizing the risk of detection due to statistical anomalies. Specifically, at this scale, the attack is constrained by several conditions.

1) The number of votes for the preferred candidates must be greater than those of the opponents.
2) The margin between vote tallies for any given race must be greater than that allowing a recount according to law.
3) Vote tallies must be "close" to pre-election polling.

These conditions can naturally be cast as a set of linear constraints, and a general solution to automatically determining the parameters to implement a large-scale strategy can be derived using linear programming. For example, consider a race where the votes reported in a county with m precincts should be biased toward a preferred candidate from a set of n candidates. Let

$$\mathbf{x} = \{x_{0,0}, \ldots, x_{n-1,m-1}\}, 0 \leq i < n, 0 \leq j < m$$

denote the percentage of the votes that each candidate i should receive in precinct j, where $x_{0,j}$ is the preferred candidate. Let $\mathbb{E}[\mathbf{x}]$ denote the expected percentage of votes each candidate will receive per precinct as indicated by pre-election polling, with a margin of error $\epsilon_{i,j}$ associated with each candidate i in precinct j. Finally, let δ be the percentage vote differential below which a recount can take place.

Then, a suitable vote distribution for each precinct can be found by maximizing the objective function

$$f(\mathbf{x}) = \sum_{j=0}^{m-1} \sum_{i=0}^{n-1} x_{i,j}$$

subject to the constraints

$$\begin{aligned} \sum_{i=0}^{n-1} x_{i,j} &= 1 \;\; \forall\, 0 \leq j < m \\ \sum_{j=0}^{m-1} x_{0,j} &\geq \left(\sum_{j=0}^{m-1} x_{i,j} \right) + \delta \;\; \forall\, 0 < i < n \\ x_{i,j} &\geq \mathbb{E}[x_{i,j}] - \epsilon_{i,j} \\ x_{i,j} &\leq \mathbb{E}[x_{i,j}] + \epsilon_{i,j}. \end{aligned}$$

This distribution is used by the malicious firmware installed on each DRE or optical scanner in a precinct as a bound on the percentage of votes each candidate should receive in a given precinct. As long as the number of votes for the preferred candidate and the opposing candidates remain above and below their respective thresholds, the malicious firmware refrains from executing an attack. If, however, a bound for any of the candidates does not hold, the malicious firmware enables attacks against subsequent voters to restore the proper local vote distribution. As long as each local distribution is satisfied, the preferred candidate is guaranteed to receive a sufficient number of votes, with an overall vote distribution that minimizes the likelihood of detection.

Of course, if the pre-election polling data does not indicate that the preferred candidate is likely to win (i.e., a plurality of votes is not within the margin of error), then the constraint system will not be feasible. In this case, the margin of error ϵ must be increased until a solution is found. This, of course, results in an increased deviation from pre-election polling data and, consequently, a higher likelihood of detection.

8.7 Discussion

In the case of both the Sequoia and ES&S voting systems, analysis of the information flow of the voting system along with the capabilities of each individual attack discovered allowed for the identification of large-scale threats against the voting systems that would not have been recognized in a piecewise analysis. Virus-style attacks clearly pose a greater threat to fair elections than attacks against single components, since a greater number of votes can be affected, and they are persistent between elections.

Additionally, the wide variety and large number of vulnerabilities discovered resulted in many vectors for the introduction of such a virus. For instance, in the case of the Sequoia voting system, the virus could be introduced by exploiting a remote vulnerability in the back-end database of the EMS. Similarly, for ES&S, the virus could be introduced into the system by exploiting the EMS during the tallying process for the preceding election.

Finally, we want to stress that these scenarios have been implemented and tested against real, certified voting systems that are in use today.[13] Far from being the purview of the mythical über-hacker, our findings indicate that large-scale exploitation of electronic voting systems is well within the capabilities of "persons having ordinary skill in the art."

9 Lessons Learned

As we have shown above, the electronic voting systems that we reviewed are neither secure nor well-designed. In this section, we summarize what we found to be the major pitfalls of both systems.

Poor integration leads to insecurity. One of the problems with most of the electronic voting systems that are being used today is that these systems are put together by integrating election components created by different companies or groups. As a consequence, often there is no overall system design and no coherent structure. In

13. A video that demonstrates the execution of these scenarios against one of the systems we analyzed is available at http://www.cs.ucsb.edu/~seclab/projects/voting/.

fact, one of the reviewed systems was a mish-mash of legacy software. Almost every component was using its own database (often storing duplicate data) and had its own authentication system, if any. *When integrating election components that were designed to be stand-alone, it is necessary to take into account the overall system design.*

Cryptography is hard to get right. One of the major areas of concern was the use of cryptography. In both systems we found that most uses of cryptographic techniques could be classified into three main categories: naive use, incorrect use, or no use at all. For example, in one of the analyzed systems, data on a DTD was protected via encryption using a strong symmetric block cipher algorithm, but the encryption key was stored in the clear on the same media. In other cases, when election data was protected by a checksum, the checksum could be changed to match the maliciously modified data. Even worse, in both systems, no cryptographically-strong signing mechanisms were used to protect the integrity of sensitive data. *A mindful usage of strong encryption algorithms with strong, well-protected keys along with data signing are a must for building secure voting systems.*

Unfounded trust assumptions enable compromise. Another major problem with both reviewed systems was a lack of mechanisms allowing one to check the origin of data along with a lack of appropriate input validation. In fact, most of the components that we reviewed assumed that input data came from a specific system component, disregarding the fact that in many cases it could easily be forged. For example, checksums were often taken as proof of data origin. Also, data that was expected to come from other components (for example, data on a DTD that was supposed to be generated by an EMS) was often unchecked for boundary cases. Many such cases resulted in exploitable vulnerabilities. It is well-known in the security community that the lack of input validation is one of the major premises for the existence of vulnerabilities. *One of the main premises for building a secure voting system is the absence of any unfounded assumptions and the careful checking of all inputs.*

Certification and standards that are currently used are not enough for security. Both of the systems analyzed were certified, and their source code was officially compliant with at least one of the standards in use today. Nevertheless, both systems were inherently insecure. The problem is that currently used source code standards are not security-oriented, and even if they were, a simple checklist-based verification would not be enough. For instance, to prevent buffer overflows, a standard could require that any use of a function writing data to a buffer should be preceded with a boundary check of input size against the size of the destination buffer. In this case, while the standard would enforce the usage of checks before each case, it would still fail to guarantee that the checks were correct. In fact, one of the exploitable buffer overflows that we found was a result of a mistake in a similar check. Also, during the review, we found that systems are not as compliant with standards as they claim to be. *A more thorough and security-oriented certification process for evaluating voting systems is needed.*

Logic and accuracy testing gives a false sense of security. One of the selling points of both systems was the fact that they provide a built-in way of testing their systems for accuracy, which can be done right before an election. In practice, from a security perspective we found the tests to be completely useless, since testing was done only while in a special testing mode, which was enabled through a switch in the system's firmware. Clearly, since the system itself is aware of the testing mode, any malicious code that is implanted into the firmware could easily pass the accuracy tests. Interestingly enough, one of the vendors seemed to have a strong belief that their logic and accuracy test is capable of identifying malicious code. *The only way to make logic and accuracy tests realistic is to, at the very least, have the firmware totally unaware of any testing mode.*

COTS components are difficult to configure in a secure way. We found that the use of COTS components in some cases made the voting systems more vulnerable. The main problem is that COTS components often come with a lot of functionality and can be hard to configure in a secure way. For example, the EMS for both systems was based on the Windows operating system, which is a very complex system of its own, with a large number of pre-configured settings. Adequate hardening of such a system requires a high level of expertise. Nevertheless, the systems that were given to us came mostly with default configurations and no specification on how to configure the system's security was given in either case. The "autorun" vulnerability presented in Section 7 is one example of this problem. *When COTS components are used, vendors should either provide a detailed specification of how the systems should be configured or they should provide pre-configured systems.*

Voting procedures underestimate the power of potential adversaries. Another common problem that we found is that the security and integrity of both systems often depend on poll workers following an explicit set of procedures. In fact, we found that the physical security of most components depended more on compliance with a set of procedures than on strong physical guards. It was often the case that the seals that were used to protect critical system components could be easily bypassed. Interestingly, the vendors seem to fail to realize that procedures cannot substitute for built-in system security and can be easily violated, intentionally or not. In fact, the rebuttal of one of the vendors to a discovered security problem was that the problem cannot occur because it violates the procedures. *Procedures should never be relied upon as the only guarantee of system security; rather, each component of a system should implement a complete set of security mechanisms necessary for its protection.*

Security training of developers is not sufficient. One of the most frustrating discoveries that we made is the apparent lack of adequate security training of the voting system developers. For example, it was often the case that code written in C consistently used the infamous *strcpy()* function without checking the size of the copied data against the size of the destination buffer, which is one of the most common causes of buffer overflows. Even more surprisingly, we found cases where the more secure *strncpy()* function was used, but incorrectly; the size of the input was checked against itself rather than against the destination buffer. *Knowledge of basic security concepts, their application, and defensive programming practices should be prerequisites for the developers of critical systems, such as an electronic voting system.*

Summary. In both electronic voting systems studied, we found that security was not a part of the design and security features were often added in an *ad hoc* way. Furthermore, the "security through obscurity" principle was often used as one of the main protection mechanisms. While undoubtedly the proprietary nature of the voting software makes it harder for an attacker to develop a working exploit for the system, we have shown that it does not make a system completely secure. Given sufficient time and determination, an attacker can successfully reverse-engineer a system, starting with very little information.

Application of the lessons presented in this section would significantly improve the security of both systems. For instance, secure software development practices, static source code vulnerability analysis, automated testing frameworks, and regular penetration testing are all means by which the quality of the various software components and system design as a whole could be improved. The proper use of standard cryptographic primitives for ensuring data integrity and provenance would render some of the attacks presented difficult or impossible to execute. Proper attention to system integration and configuration would greatly decrease the attack surface of electronic voting systems that incorporate COTS components.

Unfortunately, one cannot consider adoption of the recommendations outlined here as a panacea for electronic voting. The design and implementation of distributed electronic voting systems that are robust against both external and insider threats, where each component from the application layer to the hardware is verifiably free from vulnerabilities and malicious code, and that satisfy the seemingly contradictory goals of transparency and protection against coercion, remains an open problem. Certainly, adoption of the above recommendations would improve the security of existing electronic voting systems. Whether the integrity of the democratic process should be entrusted to such a system is another question entirely.

10 Conclusions

In this paper, we presented our analysis of two real-world electronic voting systems. These analyses were performed as part of state-wide efforts that were unprecedented in terms of access to the hardware and software components of the systems.

As part of these exercises, we devised a testing methodology, developed new tools that are specifically tailored to the security analysis of these systems, and learned a number of lessons, all of which should be of use to other testers that need to evaluate similar systems.

In both the systems that we analyzed, we found major security vulnerabilities that could compromise the confidentiality, integrity, and availability of the voting process. These vulnerabilities allowed us to develop virus-like malware that can spread from one component of the system to another, eventually taking control of all aspects of vote casting and tallying.

The results of our study suggest that there is a need for a drastic change in the way in which electronic systems are designed, developed, and tested. Researchers, practitioners, and policy makers need to define novel testing approaches that take into account the peculiar information flow of these systems, as well as the combination of computer security mechanisms and physical procedures necessary to provide a high level of assurance.

However, experience with other critical application domains has shown that building provably secure systems is not attainable in practice. Therefore, implementing secure voting processes will require improvements in hardware design, software development, voting procedures, and voter education.

Acknowledgments

We want to thank the Secretaries of State of California and Ohio, Debra Bowen and Jennifer Brunner, respectively. Without their courageous approach to the security testing of electronic voting systems these studies would not have been possible. In addition, we want to thank all the people that worked side-by-side with us during the TTBR and EVEREST evaluations, including Matt Bishop, David Wagner, Patrick McDaniel, Matt Blaze, Robert P. Abbott, Adam Aviv, Aaron J. Burstein, Kevin Butler, Joseph A. Calandrino, Pavol Cerny, Sandy Clark, Arel Cordero, Eric Cronin, Mark Davis, Joseph Edmonds, William Enck, Sophie Engle, Ariel J. Feldman, Luke Florer, Nathan S. Good, J. Alex Halderman, Joseph Lorenzo Hall, Candice Hoke, Harri Hursti, Srinivas Inguva, Chris Karlof, Dave Kettyle, Steve McLaughlin, Deirdre K. Mulligan, Brian Porter, Elliot Proebstel, Laura Quilter, Eric Rescorla, Naveen Sastry, Hovav Shacham, Gaurav Shah, Sujeet Shenoi, Micah Sherr, Jacob Stauffer, Till Stegers, Patrick Traynor, Dan S. Wallach, Ka-Ping Yee, Harlan Yu, and William P. Zeller.

References

[1] S. Pynchon and K. Garber, "Sarasota's Vanished Votes: An Investigation into the Cause of Uncounted Votes in the 2006 Congressional District 13 Race in Sarasota County, Florida," Florida Fair Elections Center Report, January 2008.

[2] T. Kohno, A. Stubblefield, A. Rubin, and D. Wallach, "Analysis of an Electronic Voting System," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2004, pp. 27–40.

[3] E. Proebstel, S. Riddle, F. Hsu, J. Cummins, F. Oakley, T. Stanionis, and M. Bishop, "An Analysis of the Hart Intercivic DAU eSlate," in *Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop*, 2007.

[4] A. Yasinsac, D. Wagner, M. Bishop, T. Baker, B. de Medeiros, G. Tyson, M. Shamos, and M. Burmester, "Software Review and Security Analysis of the ES&S iVotronic 8.0.1.2 Voting Machine Firmware," Security and Assurance in Information Technology Laboratory, Florida State University, Tallahassee, FL, Tech. Rep., 2007.

[5] G. Vigna, R. Kemmerer, D. Balzarotti, G. Banks, M. Cova, V. Felmetsger, W. Robertson, and F. Valeur, "Security Evaluation of the Sequoia Voting System," Top-To-Bottom Review of the California Voting Machines, July 2007.

[6] P. McDaniel, M. Blaze, and G. Vigna, "EVEREST: Evaluation and Validation of Election-Related Equipment, Standards and Testing," Ohio Secretary of State's EVEREST Project Report, December 2007.

[7] "Sequoia Voting Systems," www.sequoiavote.com/, May 2009.

[8] "Election Systems & Software," www.essvote.com/, May 2009.

[9] D. Jones, "A Brief Illustrated History of Voting," http://www.cs.uiowa.edu/~jones/voting/pictures/, 2003.

[10] A. Gumbel, *Steal This Vote: Dirty Elections and the Rotten History of Democracy in America*. Nation Books, 2005.

[11] 107th Congress, "Help America Vote Act," Public Law 107–252, 2002.

[12] R. Hite, "All Levels of Government Are Needed to Address Electronic Voting System Challenges," GAO, Tech. Rep., 2007.

[13] Election Assistance Commission, "State Governments' Use of Help America Vote Act Funds," EAC, Tech. Rep., 2007.

[14] T. Tibbetts and S. Mullis, "Challenged ballots: You be the judge," http://minnesota.publicradio.org/features/2008/11/19_challenged_ballots, 2008.

[15] Common Cause and VotersUnite!, "A Master List of 70+ Voting Machine Failures and Miscounts by State."

[16] Verified Voting Foundation, "Electronic Miscounts and Malfunctions in Recent Elections," http://verifiedvotingfoundation.org/downloads/resources/documents/ElectronicsInRecentElections.pdf.

[17] VotersUnite!, "ES&S in the News – A Partial List of Documented Failures," http://www.votersunite.org/info/ES&Sinthenews.pdf.

[18] M. Gondree, P. Wheeler, and D. DeFigueiredo, "A Critique of the 2002 FEC VSPT E-Voting Standards," University of California, Davis, Tech. Rep., 2005.

[19] R. Mercuri, "Voting System Guidelines Comments," http://www.wheresthepaper.org/VVSGComment.pdf, 2005.

[20] P. Neumann, "Security Criteria for Electronic Voting," in *Proceedings of the National Computer Security Conference*, 1993.

[21] R. Saltman, "Accuracy, Integrity, and Security in Computerized Vote-Tallying," Institute for Computer Sciences and Technology, National Bureau of Standards, Tech. Rep., 1988.

[22] B. Harris, *Black Box Voting: Ballot Tampering in the 21st Century*. Elon House/Plan Nine, 2003.

[23] ——, "Inside a U.S. Vote Counting Program," http://www.scoop.co.nz/stories/HL0307/S00065.htm, July 2003.

[24] A. Rubin, *Brave New Ballot*. Broadway, 2006.

[25] SAIC, "Risk Assessment Report: Diebold AccuVote-TS Voting System and Processes," Science Applications International Corporation, Tech. Rep., 2003.

[26] M. Wertheimer, "Trusted Agent Report: Diebold AccuVote-TS Voting System," RABA Technologies, LLC, Tech. Rep., 2004.

[27] A. Feldman, J. Halderman, and E. Felten, "Security Analysis of the Diebold AccuVote-TS Voting Machine," in *Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop*, 2007.

[28] E. Felten, ""Hotel Minibar" Keys Open Diebold Voting Machines," http://www.freedom-to-tinker.com/?p=1064.

[29] A. Appel, "How I bought used voting machines on the Internet," http://www.cs.princeton.edu/~appel/avc/, February 2007.

[30] R. Gonggrijp and W. Hengeveld, "Studying the Nedap/Groenendaal ES3B Voting Computer: A Computer Security Perspective," in *Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop*, 2007.

[31] T. Ryan and C. Hoke, "GEMS Tabulation Database Design Issues in Relation to Voting Systems Certification Standards," in *Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop*, 2007.

[32] A. Aviv, P. Cerný, S. Clark, E. Cronin, G. Shah, M. Sherr, and M. Blaze, "Security Evaluation of ES&S Voting Machines and Election Management System," in *Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop*, 2008.

[33] K. Butler, W. Enck, H. Hursti, S. McLaughlin, P. Traynor, and P. McDaniel, "Systemic Issues in the Hart InterCivic and Premier Voting Systems: Reflections on Project EVEREST," in *Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop*, 2008.

[34] H. Hursti, "Critical Security Issues with Diebold Optical Scan Design," Black Box Voting Project, Tech. Rep., July 2005.

[35] A. Kiayias, L. Michel, A. Russell, N. Shashidhar, and A. See, "Tampering with Special Purpose Trusted Computing Devices: A Case Study in Optical Scan E-Voting," in *Proceedings of the Annual Computer Security Applications Conference*, 2007.

[36] A. Kiayias, L. Michel, A. Russell, N. Shashidhar, A. See, and A. Shvartsman, "An Authentication and Ballot Layout Attack against an Optical Scan Voting Terminal," in *Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop*, 2007.

[37] A. Rubin, "Security Considerations for Remote Electronic Voting," *Communications of the ACM*, vol. 45, no. 12, pp. 39–44, 2002.

[38] D. Jefferson, A. Rubin, B. Simons, and D. Wagner, "A Security Analysis of the Secure Electronic Registration and Voting Experiment (SERVE)," US Department of Defense, Tech. Rep., 2004.

[39] ——, "Analyzing Internet Voting Security," *Communications of the ACM*, vol. 47, no. 10, pp. 59–64, 2004.

[40] A. Appel, "Ceci n'est pas une urne: On the Internet vote for the Assemblée des Français de l'Etranger," http://www.cs.princeton.edu/~appel/urne.html.

[41] J. Halderman, E. Rescorla, H. Shacham, and D. Wagner, "You Go to Elections with the Voting System You Have: Stop-Gap Mitigations for Deployed Voting Systems," in *Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop*, 2008.

[42] J. Bethencourt, D. Boneh, and B. Waters, "Cryptographic Methods for Storing Ballots on a Voting Machine," in *Proceedings of the Network and Distributed System Security Symposium*, 2007.

[43] D. Molnar, T. Kohno, N. Sastry, and D. Wagner, "Tamper-Evident, History-Independent, Subliminal-Free Data Structures on PROM Storage-or-How to Store Ballots on a Voting Machine (Extended Abstract)," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2006, pp. 365–370.

[44] S. Garera and A. Rubin, "An Independent Audit Framework for Software Dependent Voting Systems," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2007, pp. 256–265.

[45] J. Hall, "Improving the Security, Transparency and Efficiency of California's 1% Manual Tally Procedures," in *Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop*, 2008.

[46] K. Weldemariam and A. Villafiorita, "Modeling and Analysis of Procedural Security in (e)Voting: the Trentino's Approach and Experiences," in *Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop*, 2008.

[47] N. Sastry and D. Wagner, "Designing Voting Machines for Verification," in *Proceedings of the USENIX Security Symposium*, 2006, pp. 321–336.

[48] K.-P. Yee, "Building Reliable Voting Machine Software," Ph.D. dissertation, University of California, Berkeley, 2007.

[49] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. Rivest, P. Ryan, E. Shen, and A. Sherman, "Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes," in *Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop*, 2008.

[50] C. Karlof, N. Sastry, and D. Wagner, "Cryptographic Voting Protocols: A Systems Perspective," in *Proceedings of the USENIX Security Symposium*, 2005, pp. 33–50.

[51] T. Moran and M. Naor, "Split-Ballot Voting: Everlasting Privacy With Distributed Trust," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2007, pp. 246–255.

[52] S. Everett, "The Usability of Electronic Voting Machines and How Votes Can Be Changed Without Detection," Ph.D. dissertation, Rice University, 2007.

[53] T. Selker and S. Cohen, "An Active Approach to Voting Verification," http://vote.caltech.edu/media/documents/wps/vtp_

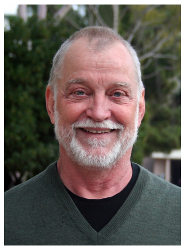
wp28.pdf, Caltech/MIT Voting Technology Project, Tech. Rep. 28, May 2005.
[54] D. Wagner, "Testimony before U.S. House of Representatives at joint hearing of the Committee on Science and Committee on House Administration," http://www.cs.berkeley.edu/~daw/papers/testimony-house06.pdf, 2006.
[55] E. Barr, M. Bishop, and M. Gondree, "Fixing Federal E-Voting Standards," *Communications of the ACM*, vol. 50, no. 3, pp. 19–24, March 2007.
[56] C. Wysopal, L. Nelson, D. D. Zovi, and E. Dustin, *The Art of Software Security Testing: Identifying Software Security Flaw*. Symantec Press, Nov 2006.
[57] G. Hoglund and G. McGraw, *Exploiting Software: How to Break Code*. Addison-Wesley, February 2004.
[58] Institute of Electrical and Electronics Engineers, *IEEE Std 1149.1-1990 IEEE Standard Test Access Port and Boundary-Scan Architecture*. IEEE, 1990.
[59] D. Rath, "Open On-Chip Debugger," http://openocd.berlios.de/web, 2008.
[60] C. S. o. S. D. Bowen, "Top-to-Bottom Review," http://www.sos.ca.gov/elections/elections_vsr.htm, July 2007.
[61] O. S. o. S. J. Brunner, "Ohio EVEREST Voting Study," http://siis.cse.psu.edu/everest.html, December 2007.
[62] United States Election Assistance Commission, "Voting System Standards," http://www.eac.gov/votingsystems/voluntary-voting-guidelines/2002-voting-system-standards, 2002.

Davide Balzarotti is an Assistant Professor at the Eurecom Institute in France. He received both his Laurea degree and his Ph.D. in Computer Engineering from Politecnico di Milano, Italy in 2002 and 2006, respectively. His research interests include most aspects of system security and in particular the areas of intrusion detection and prevention, binary and malware analysis, reverse engineering, and web security.

Greg Banks received his M.S. in Computer Science from the University of California, Santa Barbara, in December 2008. His research interests include malware analysis and malicious web infrastructure. He is currently living and working in the Bay Area.

Marco Cova is currently a Ph.D. student at the University of California, Santa Barbara. He holds a Laurea degree in Electrical and Computer Engineering from the University of Bologna, Italy. His research interests include malware analysis, web security, intrusion detection, and electronic voting security.

Viktoria Felmetsger is a Ph.D. candidate in the Department of Computer Science at the University of California, Santa Barbara. Her research interests include all aspects of vulnerability analysis with emphasis on web applications.



Richard A. Kemmerer is the Computer Science Leadership Professor and past Chair of the Department of Computer Science at the University of California, Santa Barbara. He has been a Visiting Scientist at the Massachusetts Institute of Technology, and a Visiting Professor at the Wang Institute and the Politecnico di Milano. From 1966 to 1974 he worked as a programmer and systems consultant for North American Rockwell and the Institute of Transportation and Traffic Engineering at UCLA. His research interests include formal specification and verification of systems, computer system security and reliability, programming and specification language design, and software engineering. He is the author of the book ”Formal Specification and Verification of an Operating System Security Kernel” and a co-author of ”Computers at Risk: Safe Computing in the Information Age,” ”For the Record: Protecting Electronic Health Information,” and ”Realizing the Potential of C4I: Fundamental Challenges.”

Dr. Kemmerer received the B.S. degree in Mathematics from the Pennsylvania State University in 1966, and the M.S. and Ph.D. degrees in Computer Science from the University of California, Los Angeles, in 1976 and 1979, respectively. He has served as a member of the National Academy of Science’s Committee on Computer Security in the DOE, the System Security Study Committee, the Committee for Review of the Oversight Mechanisms for Space Shuttle Flight Software Processes, the Committee on Maintaining Privacy and Security in Health Care Applications of the National Information Infrastructure, and the Committee on the Review of Programs for C4I. He has also served as a member of the National Computer Security Center’s Formal Verification Working Group and was a member of the NIST’s Computer and Telecommunications Security Council. Dr. Kemmerer is also the past Chair of the IEEE Technical Committee on Security and Privacy and a past member of the Advisory Board for the ACM’s Special Interest Group on Security, Audit, and Control. He is a Fellow of the IEEE Computer Society, a Fellow of the Association for Computing Machinery, and the 2007 recipient of the Applied Security Associates Distinguished Practitioner Award. He is a member of the IFIP Working Group 11.3 on Database Security, and a member of the International Association for Cryptologic Research. He is a past Editor-in-Chief of IEEE Transactions on Software Engineering and has served on the editorial boards of the ACM Computing Surveys and IEEE Security and Privacy.

William Robertson is a researcher with the Computer Security Group at the University of California, Santa Barbara, where he received his Ph.D. in June 2009. His research interests include web application security, anomaly detection, testing and evasion of intrusion detection systems, static and dynamic analysis, and electronic voting security.

Fredrik Valeur received his Ph.D. from the University of California, Santa Barbara. His research interests include web application security, penetration testing and electronic voting machines.

Giovanni Vigna is a Professor in the Department of Computer Science at the University of California in Santa Barbara. He received his M.S. with honors and Ph.D. from Politecnico di Milano, Italy, in 1994 and 1998, respectively. His current research interests include web security, malware analysis, vulnerability assessment, and intrusion detection. Giovanni Vigna edited a book on Security and Mobile Agents and authored one on Intrusion Correlation. He has been the Program Chair of the International Symposium on Recent Advances in Intrusion Detection (RAID 2003), the Network and Distributed System Security Symposium (NDSS 2009), and he is the co-Chair for the IEEE Symposium on Security and Privacy (S&P 2010). He is on the editorial board of the ACM Transactions on Information and System Security (ACM TISSEC) and of the IEEE Transactions on Dependable and Secure Computing (IEEE TDSC). He was also on the editorial boards of the Journal of Computer Security (JSC) and of the IEEE Security & Privacy Magazine.

In addition to his academic research, Giovanni Vigna is also interested in every aspect of hacking. He participated to various hacking competitions, and he lead the Shellphish team in several editions of the DefCon CTF hacking competition, which the Shellphish team won in 2005. Finally, he is known for organizing and running the largest inter-university Capture The Flag hacking contest (iCTF), which involves dozens of institutions and hundreds of students around the world every year. Giovanni Vigna is a member of IEEE and ACM.