

Chapter 7

Practical Attacks on Real-World E-Voting

J. Alex Halderman

University of Michigan

CONTENTS

7.1	Introduction	143
7.2	Touchscreen DREs	144
	7.2.1 Diebold	145
	7.2.2 Top to Bottom	149
	7.2.3 The Test of Time	150
	7.2.4 Around the World	151
7.3	Internet Voting	157
	7.3.1 The Washington, D.C., Internet Voting System	158
	7.3.2 Estonia's Internet Voting System	162
	7.3.3 The New South Wales iVote System	165
7.4	Conclusion	169

7.1 Introduction

Many democracies rely on e-voting systems for binding elections, whether for in-person voting at poll sites or for remote voting over the Internet. (See Ch. 3 for an overview of the current state of e-voting worldwide.) Yet in practically every case where a fielded e-voting system has been publicly scrutinized by capable independent security experts, it has turned out to have serious vulnerabilities with the potential to

disrupt elections, compromise results, or expose voters' secret ballots. This chapter highlights some of the most significant results from these studies and attempts to explain why real-world e-voting security failures are so widespread. It focuses on two classes of systems: poll-site DREs (Section 7.2) and online voting (Section 7.3).

E-voting faces a wide variety of potential attackers beyond those considered in traditional elections. These include insider attacks from system administrators, cybercriminals working for dishonest candidates, "hacktivists" seeking to disrupt elections as a form of political protest, and even sophisticated nation-states applying offensive cyberwarfare capabilities. We can roughly divide these attackers' goals into three categories: (1) *Tampering with the election outcome*, e.g., to favor particular candidates; (2) *Discovering how people voted*, e.g., to retaliate against those who voted against the attacker's preferred candidates, as a means of enforcing vote buying or coercion; and (3) *Disrupting or discrediting the election process*, e.g., through denial of service, conspicuous tampering, or the false appearance of such problems.

Defending against all these potential attacks simultaneously is difficult, for several reasons. First, some of the central security properties that an e-voting system needs to provide are in tension. Countermeasures against vote tampering—such as backups, logs, and receipts—tend to make it harder to strongly protect ballot secrecy. Likewise, mechanisms for protecting ballot secrecy—such as encrypting voted ballots and avoiding incremental backups—make detecting and responding to compromise more difficult. Second, election software tends to be more complex than one might naively assume, creating large numbers of opportunities for bugs and vulnerabilities to occur. Third, even if the code that's supposed to be running an election system is perfect, there is no way to guarantee that it is the actual code (and the *only* code) that is running on election day.

Elections are not just an ordinary application that has to be kept safe. Given the money and political power in play during a high-stakes election for public office, the incentive to cheat is enormous, and the consequences of a fraudulent outcome could threaten social order and national sovereignty. Election systems are critical infrastructure, as important to defend as the power grid or financial system. Yet most deployed e-voting systems have been built to the same level of quality as typical commercial IT projects, far below the appropriate level for critical infrastructure.

These challenges have been compounded because many e-voting system vendors and election officials have shied away from rigorous public security scrutiny. They've used electoral laws, intellectual property claims, computer intrusion statutes, and non-disclosure agreements to create impediments for concerned citizens and security researchers. As a result, the hardest part of finding security problems with fielded e-voting systems has often been to legally gain access to the systems for study. (Of course, actual criminals intent on rigging an election are unlikely to be deterred by the need to break the law.) It's unfortunate that officials and skeptical researchers are sometimes at odds, since ultimately both groups are working to see elections conducted well, and since, as this chapter shows, voters have good reason to be worried about the current state of e-voting security.



Figure 7.1: The Diebold AccuVote-TS was the most common DRE in the U.S. [233].

7.2 Touchscreen DREs

Direct recording electronic (DRE) voting machines are essentially general-purpose computers running specialized election software. Computer scientists have long been skeptical that voting systems of this type can be made secure. Experience with computer systems of all kinds shows that it is exceedingly difficult to ensure the reliability and security of complex software or to detect and diagnose problems when they do occur. Yet DREs rely fundamentally on the correct and secure operation of complex software programs.

7.2.1 Diebold

In the United States, the first touchscreen DRE to receive significant public scrutiny was the Diebold AccuVote-TS, shown in Figure 7.1. The AccuVote-TS and its newer relative the AccuVote-TSx were, in the mid-2000s, the most widely deployed electronic voting platform in the U.S. In the November 2006 general election, they were used in 385 counties representing over 10% of registered voters [211]. More than 33,000 of the TS machines were in service nationwide [205].

In most respects, the AccuVote-TS was a prototypical DRE. It interacted with the user via an integrated touchscreen LCD display. To authenticate voters and election officials, it used a motorized smartcard reader. On the side of the machine, behind a locked metal door, was a memory card slot that poll workers used to load ballot definitions and retrieve election results. The machine was also equipped with a small printer that recorded the final vote totals. Internally, the hardware resembled that of a laptop PC or a personal digital assistant. It included a RISC processor, 32 MB of RAM and 16 MB of flash storage. When the machine was first switched on, it loaded a boot-loader from the on-board flash. The boot-loader loaded the operating system, Windows CE 3.0, and Windows launched an application called BallotStation that conducted the election.

The first major study of these machines was carried out in 2003 by Kohno, Stubblefield, Rubin and Wallach, who studied a leaked version of the BallotStation source code and found many design errors and vulnerabilities [351]. (Diebold employees had stored a copy of the code on a company FTP site that was accessible to the public, where it was discovered by Bev Harris of BlackBoxVoting [330].) Diebold responded that the findings were “unrealistic” and pointed out that the researchers did not test with a real voting machine or a production version of the software [206].

Public concern in light of Kohno’s study led the state of Maryland to authorize two security studies. The first, by SAIC, reported that the system was “at high risk of compromise” [521]. The second, conducted by RABA, a security consulting firm, confirmed many of Kohno’s findings and suggested design changes to the Diebold system [475]. A further security assessment was commissioned by the Ohio Secretary of State and carried out by Compuware [168]. It examined several DRE systems including the AccuVote-TS and identified a number of high-risk security problems. Although these commercial studies reached similar conclusions, none of them offered the public a detailed technical description of the security issues they found.

Independent researchers finally had an opportunity to study the AccuVote hardware in 2006. Harri Hursti examined the hardware and compiled boot-loader firmware of AccuVote-TS and TSx systems [315]. He discovered problems with a software update mechanism that could allow malicious parties to replace the programs that operated the machines.

The same year, Feldman, Halderman and Felten at Princeton obtained an AccuVote-TS from a private party and reverse engineered its hardware and software [233]. They confirmed the results of the earlier studies by building working demonstrations of several reported attacks, and they also discovered a variety of serious new vulnerabilities. Their main findings were:

1. They confirmed Hursti’s discovery that anyone who had physical access to the machine—or to a memory card that would later be inserted into a machine—could install malicious software. This could be achieved by opening the machine and replacing a socketed ROM chip inside, or, more easily, by exploiting back-door features in Diebold’s boot-loader firmware. When the machine booted, it checked the removable memory card to see whether certain spe-

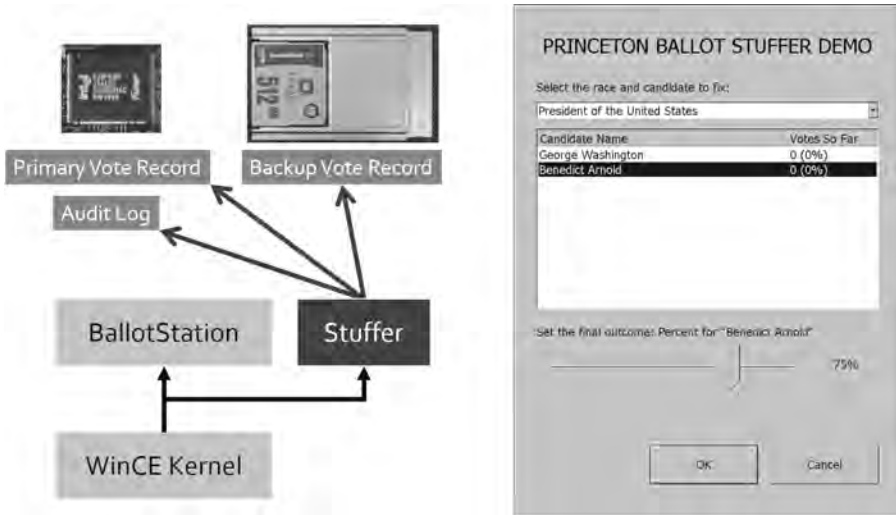


Figure 7.2: The Princeton Diebold study demonstrated vote-stealing malware that could spread from machine to machine while changing all records of the vote [233].

cial files existed. If a file named `explorer.glb` was present, the machine would launch Windows Explorer in place of Diebold’s `BallotStation` election software, allowing an attacker to manipulate the software and files on the machine. Alternatively, if the memory card contained a file named `fboot.nb0` or `nk.bin`, the machine would replace the boot-loader code or operating system partition in its on-board flash memory with the file’s contents. This was apparently intended as a software update mechanism, but there were no cryptographic integrity checks or on-screen confirmation prompts.

2. The Princeton team went on to create demonstration vote-stealing malware that modified all of the vote records, audit logs, and protective counters stored by the machine, so that even careful forensic examination of the files would find nothing amiss. All of these records were under the control of user-space software running on the machine. The demonstration malware, illustrated in Figure 7.2, was a Windows CE application that ran invisibly in the background alongside the `BallotStation` application. It included a user interface that allowed the attacker to interactively control which candidate would receive what fraction of votes. The malware parsed each new ballot as it was cast and then switched the minimum number of votes necessary to ensure that the favored candidate always had at least the desired percentage of the total.
3. The researchers developed a voting machine virus that could spread the vote-stealing code automatically and silently from machine to machine during normal pre- and post-election activities. The virus propagated via the removable memory cards by exploiting the software update mechanism to replace the ex-

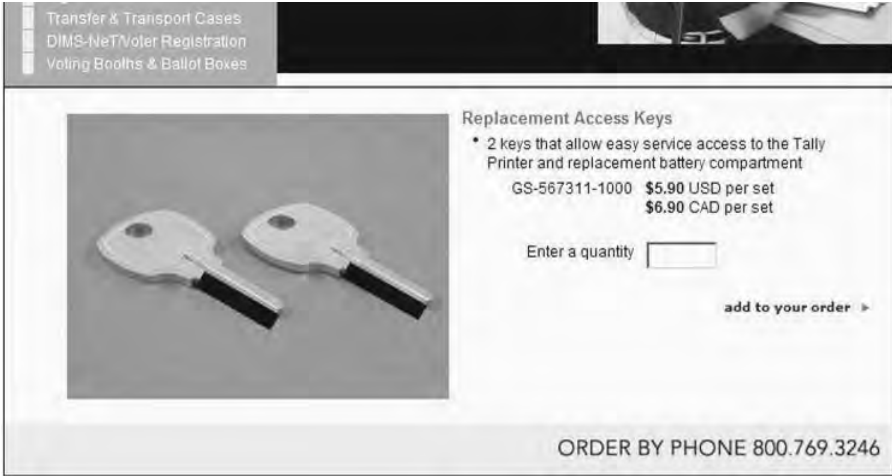


Figure 7.3: Voting machine keys for sale on Diebold’s website [277]. The key cuts (blacked out by this author) were shown clearly enough to produce working replicas.

isting boot-loader in the machine’s on-board flash. Once installed, the virus copied itself to every memory card inserted into the infected machine. If those cards were inserted into other machines, they too would become infected. As a result, an attacker could infect a large population of machines while only having temporary physical access to a single machine or memory card. Once the virus infected a machine, removing it would require factory service, since the malicious boot-loader disabled in-field software updates.

Although the unauthenticated update mechanism made it easy to tamper with the machine’s software given access to the memory card, an attacker might still have to bypass one physical security feature: the lock on the memory card door (Figure 7.1, bottom right). Yet even in this respect, the AccuVote-TS gave attackers a variety of weaknesses to choose from:

1. The lock was of poor quality and could be picked using only paperclips.
2. All AccuVote-TS machines shared the same key cuts, so stealing the key to any one of them would allow an attacker to open them all.
3. The exact same key is commonly used in office furniture, jukeboxes, and hotel minibars, and is for sale at many online retailers. The Princeton team purchased copies of the key from several sources, and all could open the machine [234].
4. Diebold itself sold replacement keys, although they could only be purchased by municipalities that already owned the machines. However, the company’s online catalog included a photograph of the key (Figure 7.3) that was detailed enough to allow anyone to create a working copy by filing a key blank [277].

A poll worker, technician, or other person who had private access to a machine for as little as one minute could use these methods with little risk of detection. Poll workers often do have such access; in a widespread practice called “sleepovers,” machines are sent home with poll workers the night before the election.

While some of these problems could be eliminated by improving Diebold’s software—such as by changing the firmware to properly authenticate software updates—others reflect deeper architectural problems with DREs. Since all records of the vote are under software control, if that software is malicious or malfunctioning, the integrity of the election results is lost.

7.2.2 *Top to Bottom*

Following these Diebold-focused studies, two states commissioned comprehensive security reviews of their election technology that encompassed products from several major e-voting vendors. In 2007, California Secretary of State Debra Bowen organized a study, the California Top-to-Bottom Review (TTBR) [130], that examined systems manufactured by Hart InterCivic, Sequoia Voting Systems, and Diebold Election Systems (which changed its name to Premier Election Solutions during the study). David Wagner of the University of California, Berkeley, led a 40-person team that performed documentation review, source code analysis, red team testing, and accessibility review. The scope of testing included not only the voting machines but also their back-end election management software, making the TTBR the most comprehensive real-world e-voting security evaluation to date.

One subset of the TTBR team analyzed the Diebold AccuVote-TSx DRE (a newer model than the AccuVote-TS studied previously), AccuVote-OS optical scanner and GEMS election management system [129]. They discovered software flaws, including buffer-overflow vulnerabilities, that attackers could exploit to install malicious software on the voting machines and on the election management back-end systems. These flaws could be used to spread a vote-stealing virus that would propagate even more efficiently and be more difficult to detect than the virus developed by the Princeton team. Furthermore, the AccuVote DRE failed to protect ballot secrecy, since the digital ballot records were retained in the order in which they were cast and contained a time stamp for each vote. This would allow election workers who observed the order in which individuals cast their ballots to discover how those individuals voted. The team concluded that because “the vulnerabilities in the Diebold system result from deep architectural flaws, fixing individual defects piecemeal without addressing their underlying causes is unlikely to render the system secure” [129].

Other TTBR review teams analyzed voting systems from Sequoia Voting Systems and Hart InterCivic and reached similar conclusions. The Sequoia team uncovered “numerous programming, logic, and architectural errors” as a result of poor design and software engineering practices. These included buffer-overflow vulnerabilities and weak or ineffective cryptography, largely based on hard-coded cryptographic keys [105]. Once again, these problems could allow vote-stealing code to

spread virally and compromise both the integrity of the election result and the secrecy of voters' ballots. The Hart review team discovered that the machines—which were connected via a local-area network inside the polling place—failed to properly secure their network interfaces, allowing voters or poll workers to connect to unsecured network links and cast votes, eavesdrop on voted ballots and even modify the software [317].

In light of the TTBR's findings, Secretary Bowen decertified DRE voting machines from all three vendors and then recertified them for limited use subject to stringent security and post-election auditing requirements [130]. In particular, California would only permit the Diebold and Sequoia DREs to be used for early voting and to operate a single machine at each poll site for accessibility purposes.

Later in 2007, Ohio Secretary of State Jennifer Brunner initiated a similar state-wide voting technology review, known as Project EVEREST [434]. The study examined systems from Elections Systems and Software (ES&S), Hart InterCivic and Premier Election Solutions, under the leadership of Patrick McDaniel from Penn State [387]. Remarkably, although the EVEREST team studied the same Diebold and Sequoia systems as California, they found yet more vulnerabilities that had been overlooked in the earlier security reviews.

In the wake of these studies, most U.S. states moved away from DRE voting. By the 2014 general election, 70% of American voters were casting ballots on paper [92].

7.2.3 *The Test of Time*

An e-voting system must withstand not only the attacks known when it is designed but also those invented during its intended service lifetime. Because the development, certification and procurement cycle for voting machines is unusually slow, the service lifetime can be 20 or 30 years. It is unrealistic to assume that any design, however good, will remain secure for so long.

Checkoway et al. [154] illustrated this in a study of the Sequoia AVC Advantage, a DRE designed in the early 1980s that was still in use in 2009 in New Jersey Louisiana and various other states. Appel et al. had performed a security review and found that the AVC Advantage could be tampered with by replacing the socketed ROM chips where their software was stored [60], but the question remained whether the machines could be manipulated without physical access.

The AVC Advantage was designed with a form of data-execution prevention (DEP) as a defense against buffer-overflow attacks. A circuit on the motherboard prevents the machine from executing any instruction fetched from its RAM, as opposed to the separate ROM chips containing the election software. This protection seemed strong in the 1980s, but two decades of advances in attack techniques have rendered it less effective. Checkoway et al. showed that they could bypass the defense using return-oriented programming, an exploitation technique introduced in

2007 [523] that allows an attacker to combine short instruction sequences already present in the ROM into a Turing-complete set of “gadgets,” from which any desired behavior can be synthesized. The team demonstrated that they could use this method to inject vote-stealing code from the machine’s removable ballot cartridges [154].

Another question surrounding the life-cycle of DRE voting machines is whether they can be safely discarded when they are taken out of service. This author discovered in 2010 that government-surplus Diebold DREs he had purchased still contained vote records from real elections [279]. As the California TTBR showed, the AccuVote machines store ballots in a predictable order, which allows an attacker to reconstruct the order in which votes were cast. By combining data left in machines’ internal flash memory with records from polling places, it might be possible to determine how individuals voted. Flash memory is tricky to completely erase [573], and more work is needed to establish adequate procedures for sanitizing the internal storage of surplus DREs before they are sent for disposal.

It would be a shame if obsolete or insecure DREs ended up in landfills, since they are often rugged PCs that could be repurposed for a range of useful purposes. Halderman and Feldman demonstrated one such application in 2010 [281]. They started with a Sequoia AVC Edge DRE that had recently been retired from elections in Virginia and reprogrammed it as a fully functional PAC-MAN machine (see Figure 7.4).

7.2.4 Around the World

Elections around the world are remarkably diverse in terms of local laws and technical requirements, and many countries prefer that election equipment is manufactured domestically, as a matter of national sovereignty. Despite this diversity, in every country where DRE voting security has been rigorously assessed, researchers have found serious vulnerabilities.

The Netherlands

In 2006, Gonggrijp and Hengeveld examined the Nedap ES3B, a DRE used by about 90% of Dutch voters [263]. The researchers were affiliated with *Wij vertrouwen stemcomputers niet* (“We do not trust voting computers”), a concerned-citizens’ group founded by Gonggrijp. After obtaining Nedap machines from two municipalities, they reverse engineered the hardware and software and discovered a variety of problems. They concluded that the system’s basic security design was lax—for example, a privileged maintenance mode was accessible using a hard-coded password (“GEHEIM,” the Dutch word for “SECRET”). Moreover, they showed that an attacker with temporary physical access to the machine could quickly replace the socketed EPROM chips containing its software. To demonstrate the risks, they developed vote-stealing malware called Nedap PowerFraud that could surreptitiously change votes while the election was underway. They also reprogrammed one of the machines to play chess (see Figure 7.5), in response to a dare from Nedap’s founder.



Figure 7.4: Voting researchers converted a surplus Sequoia AVC Edge DRE into a working PAC-MAN machine to show how easily its software could be modified [281].

Gonggrijp and Hengeveld also discovered a serious threat to voter privacy that had not been considered in previous research. They discovered unintended radio emanations from the ES3B that could be picked up on an inexpensive radio scanner or short-wave receiver from several meters away. The signal was different depending on whether or not the machine’s display was showing text with an accented character. The display would show the name and party of the selected candidate, and since only one major political party in the Netherlands had an accent in its name, this side-channel could leak substantial information about the voter’s choice [263].

Subsequent to these findings, The Netherlands abandoned DRE voting and returned to manually counted paper ballots [545]. In Germany, where Nedap DREs had also been widely adopted, the findings influenced a legal challenge to the use of electronic voting. In 2009, the German Federal Constitutional Court ruled that paperless DREs were unconstitutional and that e-voting was only permissible if “the essential steps of the voting and of the determination of the result can be examined by the citizen reliably and without any specialist knowledge” [122] (see Ch. 3).



Figure 7.5: Dutch researchers showed that they could quickly replace the firmware ROMs in the Nedap ES3B DRE to tamper with votes ... or make it play chess [263].

Brazil

Brazil has used DRE voting machines nationally since 2000. Although the legislature briefly adopted a VVPAT requirement in 2002, it was eliminated the following year [483]. The electoral authorities have long maintained that the machines are completely secure, but they refused until recently to allow any meaningful independent review of the hardware and software. In 2009, they inaugurated a series of periodic public security tests, and outside experts first had some limited access to the voting machines' source code in 2012, during the second round of this public testing.

A team of researchers from the University of Brasilia, led by Diego Aranha, took part in the 2012 tests [62]. They uncovered serious weaknesses, including mishandling of encryption keys and insecure software engineering practices. Most significantly, they discovered a major flaw in the machines' privacy protections.

Brazil's DREs (see Figure 7.6) store a digital record of each vote, but, to safeguard the secret ballot, the records are kept in a shuffled order produced using a pseudorandom number generator (PRNG). When examining the source code, the researchers found that, rather than using a cryptographically secure PRNG, the Brazilian machines were programmed to use the `C rand()` and `srand()` functions (a notoriously insecure PRNG). Moreover, the PRNG was seeded using the time (in



Figure 7.6: Brazil’s DRE voting machines failed to properly shuffle electronic vote records, potentially allowing malicious insiders to deanonymize them.

seconds) when the machine was initialized. As the researchers pointed out, this time is also recorded on the printed “zero tape” and in the system’s logs that accompany the vote database. As a result, a malicious insider or external attacker who gains access to the data recorded by the DRE could completely reverse the shuffling process and associate every voter in sequence with their ballot choices, fully compromising the secret ballot [62].

Although the electoral authorities downplayed these findings (the judges in the public test awarded the researchers’ team a score of 0.03 points out of a possible 400!), DRE security issues received widespread public attention following the close outcome of the 2014 presidential race. A legal challenge and security audit conducted by the losing political party concluded that the paperless DREs were not possible to audit [474]. In response, the national legislature overrode a presidential veto to pass a law reintroducing a VVPAT requirement [61].

Argentina

Some municipalities in Argentina use an unusual e-voting system called *Vot.ar*, which relies on paper ballots with embedded RFID chips [570]. Voters use a touch-screen ballot marker that prints their selections while simultaneously storing them in the RFID chip. The ballots are collected in a ballot box and then counted using an

RFID reader. The totals from each polling place are transmitted electronically to a central counting server to produce the election results.

During elections in Buenos Aires in July 2015, various independent researchers uncovered problems with *Vot.ar*. Joaquín Sorianello found that the private keys used to transfer the polling place results were publicly available on the Internet [106], potentially allowing an attacker to monitor or manipulate the returns. An anonymous leaker published the *Vot.ar* source code [471], and it was analyzed by a group of researchers who discovered a major flaw in the vote counting logic [56]. As a result of this flaw, an attacker could manipulate the data stored in the RFID chip so that a ballot would be counted as multiple votes.

Rather than responsibly addressing these concerns, authorities in Argentina attempted to suppress them. The Buenos Aires Metropolitan Police raided Sorianello's home and seized his computers, and a judge ordered Argentinian ISPs to block some of the websites where information about the system's security was being shared [106].

India

India, the world's largest democracy, is also the world's largest user of DRE voting. In the 2014 parliamentary election, more than 550 million voters cast their ballots on 1.4 million machines [575]. India's DREs look very different from those used in the U.S. and Europe. A marvel of engineering minimalism, they are simple, battery-powered embedded systems, consisting of a ballot unit used by voters and a control unit used by poll workers (see Figure 7.7). They lack upgradable software and interfaces for digitally loading ballot designs or offloading results. Instead, workers press buttons to reset the machines and set the number of candidates on the ballot. After the election, the control unit shows the vote total for each candidate on an LED display.

The Election Commission of India has never permitted a rigorous independent security review of the machines, known in India as EVMs, and has kept many details of their design secret, but it has maintained that they are "fully tamper-proof." This claim was challenged in 2010 by a team led by a self-taught engineer and business owner from Hyderabad named Hari K. Prasad. He collaborated with this author, Rop Gonggrijp and several others to perform a detailed security study of a machine provided by an anonymous government whistle-blower [578].

The researchers demonstrated two attacks that involve physically tampering with the machines' hardware. The DREs are stored between elections in low-security facilities that typically contain thousands of machines, so large-scale unauthorized access is a credible threat. The first attack was to replace the LED display in the control unit with a look-alike, shown in Figure 7.8, that would substitute fraudulent totals when showing the election results [578]. The dishonest display contained a hidden Bluetooth radio that the attacker would use to select the winning candidate on a smartphone app.



Figure 7.7: Halderman, Prasad, and Gonggrijp studied an Indian DRE provided by an anonymous whistle-blower [578]. The simple embedded system design consists of a ballot unit used by voters (*left*) and a control unit used by poll workers (*right*).

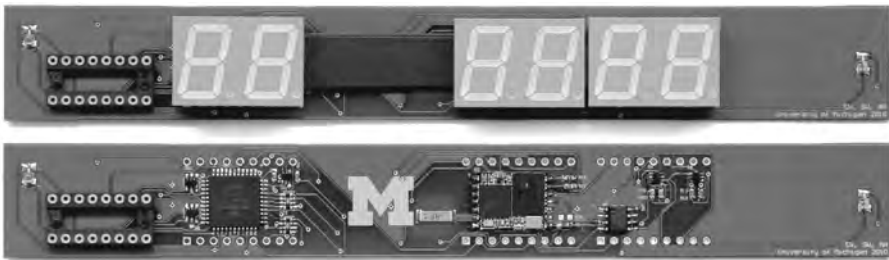


Figure 7.8: Researchers demonstrated how attackers could replace the display in the Indian DRE with a dishonest look-alike. They added parts (*bottom*), hidden under the LEDs, that substitute fraudulent vote totals when showing election results [578].

The second attack was a digital form of ballot box stuffing—which had been a widely reported problem in India prior to the introduction of e-voting [574]. The researchers constructed an inexpensive hand-held device that could be attached to the DREs’ memory chips to quickly modify the vote records [578]. This attack could be performed by criminals who took over a poll site or by insiders during the interval between voting and counting, which may be several weeks in India.

Both attacks were made far easier and cheaper by the machines' minimalist design. (Among other simplifications, the EVMs do not use even basic cryptography to protect vote data.) They illustrate how DREs' vulnerabilities stem not only from complexity, as exhibited by systems in the U.S. and Europe, but more fundamentally from the need to protect against tampering for the entire life-cycle of the equipment.

A few months after Prasad and his coauthors published their study, he was arrested by authorities demanding to know who provided the machine [278]. Although Prasad was detained for more than a month, he refused to name his source [559]. The arrest drew national and international attention and led the leaders of India's major political parties to ask the Election Commission to implement a paper trail and other security measures [313]. The Election Commission subsequently began trials of a VVPAT printer attachment [555]. In 2013, the Supreme Court of India ruled that such a paper trail was "an indispensable requirement of free and fair elections" [549] and directed the government to fully implement the VVPAT, though as of 2015 the printers had only been introduced in some constituencies [413].

7.3 Internet Voting

Conducting elections for public office over the Internet raises severe security risks, beyond even those facing poll-site systems. Election servers must be accessible from the public Internet, exposing them to the potential for remote compromise and denial of service. Voters interact with these servers from their own devices, which are frequently infected with malware. Several researchers have cataloged threats to internet voting (e.g., [322, 492]), even as others have proposed systems and protocols that may be steps to solutions someday (see Section III of this book). Although a number of states and countries have charged ahead with systems for collecting votes online (see Chapters 5 and 6), every such system that has received rigorous independent security scrutiny has been shown to have significant vulnerabilities.

Among the practical challenges to secure internet voting are:

The Poor State of Software Security

Real-world internet voting systems tend to be built on top of commercial-off-the-shelf (COTS) software, which, despite the use of the term "commercial," includes most everyday open-source software. Unfortunately, the dominant security practice for COTS developers is still "penetrate and patch." While this approach is suitable for the economic and risk environment of typical home and business users, it is not appropriate for critical security systems, such as voting applications, due to the severe consequences of failure.

Architectural Brittleness in Web Applications

Getting web security right is complicated, and small mistakes in the implementation and configuration of web applications can result in total compromise. In this sense,

the web is a brittle platform for secure application development. This is illustrated by the vulnerabilities in the Washington, D.C., and New South Wales web-based Internet voting systems, described below. In both cases, vulnerabilities resulting from small oversights—which could have been prevented by changing single lines of code—jeopardized the integrity of election results. Mistakes like these are common in web applications, and they are hard to eradicate because of the multitude of places in the software that they can exist, any one of which might be overlooked.

Exposure to Internet-Based Threats


Unlike poll-site voting, online voting systems necessarily have servers that are accessible from the public Internet. Consequently, they expose what might otherwise be a regional election to attackers from around the globe. Over the past decade, attackers have become increasingly sophisticated, and critical systems such as elections now face potential attacks from advanced cybercriminals and even state-sponsored attacks. In addition to compromising the central voting server, attackers could launch denial-of-service attacks aimed at disrupting the election, they could try to redirect voters to fake voting sites, and they could conduct widespread attacks on voters' client machines, perhaps using pre-existing botnet infections. These threats correspond to some of the most difficult unsolved problems in Internet security and are unlikely to be overcome soon.

While Internet-based financial applications, such as online banking, share some of the threats faced by Internet voting, there is a fundamental difference in ability to deal with compromises after they have occurred. In the case of online banking, transaction records, statements and multiple logs allow customers to detect specific fraudulent transactions—and, in many cases, allow the bank to reverse them. Internet voting systems cannot keep such fine-grained transaction logs without violating ballot secrecy for voters. Even with these protections in place, banks and merchants suffer billions of dollars of online fraud every year but write it off as part of the cost of doing business [364]. Fraudulent election results are more difficult to tolerate.

7.3.1 *The Washington, D.C., Internet Voting System*

In 2010, the District of Columbia developed an Internet voting pilot project that was intended to allow military and overseas absentee voters to cast their ballots using a website [542]. Prior to deploying the system in the general election, the District held a unique public trial: they conducted a mock election during which anyone was invited to test the system or attempt to compromise its security [190].

A team from the University of Michigan, led by this author, participated in the trial. Within 36 hours of the system going live, the Michigan team had gained nearly complete control of the election server. They successfully changed every vote and revealed almost every secret ballot. Election officials did not detect their intrusion for nearly two days—and might have remained unaware for far longer had the intruders not deliberately left a prominent clue. This case study was the first to analyze the



District of Columbia
Digital Vote-by-Mail Service

[Home](#) [About](#) [Help](#)

DC General Election

November 2, 2010

The service offers two options:

1

Physical Ballot Return

Complete your ballot and return materials by mail or express delivery service.

- Obtain your blank ballot and other vote-by-mail materials
- Complete them online and print them
- Return materials by **mail or express delivery service**

See more information about this option.

2

Digital Ballot Return

Complete your ballot and return it electronically. This pilot project allows you to return your ballot through the Internet.

- Obtain your blank ballot and other vote-by-mail materials
- Complete them online
- Return completed ballot **electronically**

See more information about this option.

Start Mail-in Ballot

Start Digital Ballot

Figure 7.9: Washington, D.C., built an Internet voting system to allow overseas military voters and other expats to return votes via a website [579].

security of a government Internet voting system from the perspective of an attacker in a realistic pre-election deployment. The story, which the Michigan team recounted in a blog post [280] and a later research paper [579], dramatically illustrates the dangers and challenges that face Internet voting in practice.

The D.C. system was created by the Washington, D.C., Board of Elections and Ethics (BOEE) and was officially known as the D.C. Digital Vote-by-Mail Service. It was developed as an open-source project in partnership with the nonprofit Open Source Digital Voting Foundation’s TrustTheVote project [447]. The software was written using the popular Ruby-on-Rails framework and the COTS Apache web server and MySQL database. To protect the servers, D.C. used a range of standard security mechanisms, including HTTPS, firewalls, and an intrusion detection system.

Months prior to the election, each eligible voter received a letter by postal mail containing a sixteen-character PIN and instructions for using the system. After visiting the election website (Figure 7.9) and logging in with their PINs, voters would download their ballots as PDF files, fill them in using a PDF reader, and upload them back to the server. To safeguard ballot secrecy, the server immediately encrypted

each uploaded ballot using a public key. When the voting period had ended, officials would transfer the encrypted ballots to an airgapped computer, decrypt them using a private key and print them for counting alongside mail-in absentee ballots.

D.C. opened the election server for the mock election on September 28, 2010. (This trial was scheduled to conclude only three days before the system would be made available for use by real voters.) After a few hours of examining the source code, the Michigan team found a devastating vulnerability.

The problem had to do with the code that processed uploaded ballots. Whenever a voter uploaded a ballot, the server would store it on disk using a temporary filename and encrypt it by executing a command called `gpg` with the name of this temporary file as a parameter, for example: `"gpg /tmp/ballot12345.pdf"`. The Michigan team realized that although the server replaced the filename with an automatically generated name (`"ballot12345"` in this example), it kept whatever file *extension* the voter provided. Instead of a file ending in `".pdf"`, an attacker could upload a file with a name that ended in almost any string, and this string would become part of the constructed command.

This would not have been a problem, except that the voting system developers mistakenly wrapped the filename in double quotes rather than single quotes. The server processed the command using the Bash shell, and in Bash, filenames in double quotes can contain special characters that cause the computer to run other commands. For example, the filename `"ballot12345.$(sleep 10)"` would cause the server to pause for ten seconds before responding (executing the command `"sleep 10"`). Consequently, by uploading ballots with carefully crafted filenames, an attacker could execute arbitrary code on the election server.

On the second day of the mock election, unbeknownst to the voting officials, the Michigan team started exploiting the vulnerability against the D.C. server. They carried out several attacks that illustrate the devastating effects that criminals could have during a real election if they gained a similar level of access:

1. *Stealing secrets.* As soon as they had breached the server, they began collecting crucial secret data, including the database username and password, the public key used to encrypt the ballots, and log, history and configuration files. This information would aid the attackers in compromising the system again if their infiltration was discovered and cut off.
2. *Changing past votes.* Next, they modified all the votes that had already been cast, replacing them with ballots marked with write-in votes for other candidates (see Figure 7.10). Although the system encrypted voted ballots, the attackers simply discarded the encrypted files and replaced them with forged ballots that they encrypted using the public key they had stolen.
3. *Changing future votes.* The attackers modified the code for the server software to rig the system so that future votes would be replaced in the same manner. This was possible because the server had been improperly configured such that the election software had sufficient privileges to modify itself.

Official Ballot
District of Columbia Mock Election
 PRECINCT 22
 September 17, 2010

INSTRUCTIONS TO VOTER

1 TO VOTE YOU MUST DARKEN THE OVAL TO THE LEFT OF YOUR CHOICE COMPLETELY. An oval darkened to the left of the name of any candidate indicates a vote for that candidate.
 2 Use only a pencil or blue or black medium ball point pen.
 3 If you make a mistake DO NOT ERASE. Ask for a new ballot.
 4 For a Write-in candidate, write the name of the person on the line and darken the oval.

DELEGATE TO THE U.S. HOUSE OF REPRESENTATIVES Vote for not more than (1)	AT-LARGE MEMBER OF THE COUNCIL Vote for not more than (1)	UNITED STATES REPRESENTATIVE Vote for not more than (1)
<input type="checkbox"/> Alice Example Democratic <input type="checkbox"/> Bob Example Republican <input type="checkbox"/> Carol Example Statehood Green <input checked="" type="checkbox"/> or write-in Skynet	<input type="checkbox"/> Joan Example Statehood Green <input type="checkbox"/> Kimberley Example Democratic <input type="checkbox"/> Liam Example Republican <input checked="" type="checkbox"/> or write-in Johnny 5	<input type="checkbox"/> Latoya Example Republican <input type="checkbox"/> Marcus Example Statehood Green <input type="checkbox"/> Newton Example Democratic <input checked="" type="checkbox"/> or write-in Colossus
MAYOR OF THE DISTRICT OF COLUMBIA Vote for not more than (1)	MEMBER OF THE COUNCIL WARD ONE Vote for not more than (1)	MEMBER OF ADVISORY NEIGHBORHOOD COMMISSION 1B DISTRICT FOUR Vote for not more than (1)
<input type="checkbox"/> Duane Example	<input type="checkbox"/> Mary Example	<input type="checkbox"/> Orlando Example

Figure 7.10: A team from the University of Michigan hacked into the D.C. Internet voting system during a mock election and replaced every ballot with a set of write-in votes for evil robots and artificial intelligence systems from science fiction [579].

4. *Compromising the secret ballot.* The attackers installed a backdoor that let them view any ballots that voters cast after the attack. This modification recorded the votes, in unencrypted form, together with the names of the voters who cast them, violating ballot secrecy.
5. *Stealth...and a “calling card.”* To test how well election officials could detect and recover from such an attack, the Michigan team did not immediately announce what they had done. Like a real attacker, they attempted to alter the server’s logs and other files to remove evidence of the intrusion. However, they also left a deliberate clue: they modified the website so that after a user voted, their browser would begin playing “The Victors,” the University of Michigan fight song. Nevertheless, the attack remained active for two days before other testers pointed out the fight song and D.C. officials took the server offline.

Recovery from such attacks is difficult; there is little hope for protecting future ballots from this level of compromise, since the code that processes the ballots is itself suspect. Using backups to ensure that compromises are not able to affect ballots cast prior to the compromise may conflict with ballot secrecy in the event that the backup itself is compromised.

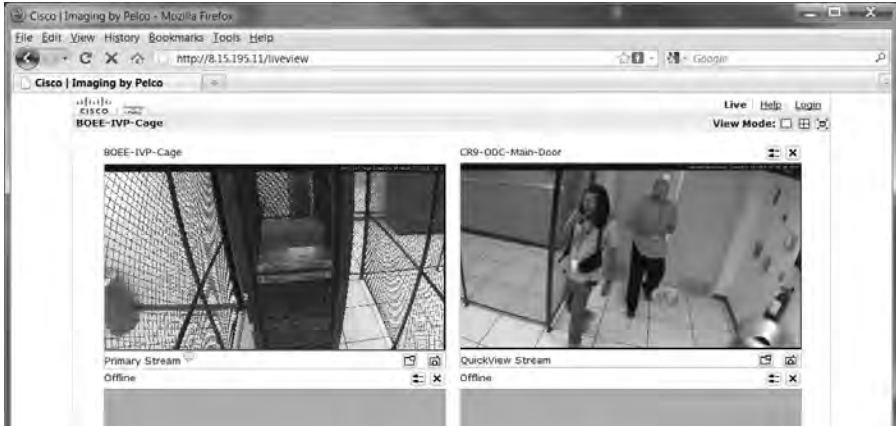


Figure 7.11: D.C. invited the public to try to compromise their Internet voting systems. A team from the University of Michigan hacked in and changed all the votes in a mock election. They also accessed webcams in the data center, shown here [579].

The Michigan attackers found a variety of additional vulnerabilities during the tests, including routers and other network equipment with easily guessed passwords. They also found video cameras in the server data center that had no passwords at all, allowing attackers to identify individuals with access to the facility and learn the schedule of security patrols (see Figure 7.11). Interestingly, although the D.C. system included a network intrusion detection system (IDS) device, it was not configured to intercept and monitor the contents of the encrypted HTTPS connections that carried the attack. This resulted in a large “blind spot” for the system administrators.

The final problem discovered by the Michigan team—and perhaps the most devastating, from an operational perspective—was that one of the election administrators had uploaded a file to the mock election server that contained the login credentials for all of the real voters who were eligible to vote online. An attacker who stole these credentials from the known-insecure test server could have used them to cast votes in the real D.C. election, which was set to begin only days later. Since these credentials had to be delivered by postal mail, there was no time to send replacements.

Based on these results, the D.C. Board of Elections and Ethics decided not to allow online ballot return during the real election. Voters were still able to download and print ballots to return by postal mail, which reduced the round-trip delivery time by about half with far less security risk.

The D.C. trial illustrated that, due to the brittleness of the web platform, small mistakes—like using double quotes in place of single quotes in one line of a complex program—can be enough to compromise all the votes in an online election. Although the specific vulnerabilities that the Michigan team exploited would be simple to fix in retrospect, it is far more difficult to guarantee that no such mistakes exist.

7.3.2 Estonia's Internet Voting System

Several countries have experimented with casting votes over the Internet, but none uses Internet voting for binding political elections to a larger extent than Estonia [329]. When Estonia introduced its online voting system in 2005, it became the first country to offer Internet voting nationally, and, in recent national elections, more than 30% of ballots were cast online [223]. Many Estonians view Internet voting as a source of national pride, but one major political party has repeatedly called for it to be abandoned [225]. Although Estonia's Internet Voting Committee maintains that the system "is as reliable and secure as voting [the] traditional way" [224], its security has been questioned by critics from Estonia (e.g., [368, 436]) and abroad (e.g., [531, 535]). See Chapter 6 for a perspective on the system by one of its creators.

Most Internet voting schemes proposed in the research literature use cryptographic techniques to achieve a property called end-to-end (E2E) verifiability (see Chapter 8). This means that anyone can confirm that the ballots have been counted accurately without having to trust that the computers or officials are behaving honestly. In contrast, Estonia's system is not E2E-verifiable. It uses a conceptually simpler design at the cost of having to trust the integrity of voters' computers, server components, and the election staff. Rather than proving integrity through technical means, Estonia relies on a complicated set of procedural controls.

Security researchers have questioned whether these controls are adequate to secure modern elections [535, 531], pointing out that the threats facing national elections have shifted significantly since the Estonian system was designed. Cyberwarfare, once a largely hypothetical threat, has become a well-documented reality [513, 379, 557, 556], and attacks by foreign states are now a credible threat to a national online voting system. As recently as May 2014, attackers linked to Russia targeted election infrastructure in Ukraine and briefly delayed vote counting [160]. Given that Estonia is an EU and NATO member that borders Russia, multiple states with significant offensive cyber capabilities might be motivated to interfere in its elections.

Despite these concerns, the system was not subjected to a detailed independent security analysis until 2014, when a team of international researchers published a paper pointing out a variety of weaknesses [535]. The team observed operations during the October 2013 local elections, conducted interviews with the system developers and election officials, assessed the software through source code inspection and reverse engineering, and performed tests on a laboratory reproduction of the system.

Although Estonia uses a number of safeguards—including encrypted websites, security chips in national ID cards, and a smartphone-based vote confirmation system, shown in Figure 7.12—the researchers showed that they all can be bypassed by a realistic state-level attacker [535]. They demonstrated client-side malware that steals the voter's credentials and then silently replaces the cast vote. Such malware could be delivered by pre-existing botnet infestations or by infecting the voting client



Figure 7.12: Voters in the Estonian Internet voting system can confirm their votes by scanning a barcode with a smartphone app [535]. Researchers showed that this mechanism could be bypassed by malware on voters’ computers or election servers.

before it is delivered to voters. They also demonstrated server-side attacks that target the centralized vote counting server. Estonia lacks any mechanism to allow voters or election officials to verify the correct operation of the counting server. By introducing malware into the server—say, through a supply-chain attack [337]—a foreign power or dishonest insider could arbitrarily change the reported results.

The researchers also observed serious lapses in the operational security practices of Estonian election officials [535]. These include administrators downloading security-critical software over unsecured Internet connections, typing secret passwords and PINs on camera in videos published to YouTube during the election, and preparing the voting software for public distribution on insecure personal laptops (see Figure 7.13), among other examples. While practices like these might be considered acceptable risks or understandable accidents in a low-security system, a critical system such as a national election platform calls for much stricter procedural controls.

The 2014 study concluded that there are multiple ways that state-level attackers, sophisticated online criminals or dishonest insiders could successfully attack the Estonian Internet voting system, and that such an attacker could plausibly change votes, compromise the secret ballot, disrupt elections or cast doubt on the integrity of results. Since these problems stem from basic architectural choices and fundamental limitations on the security and transparency that can be provided by procedural controls, the researchers recommended that Estonia suspend the use of the system [535].

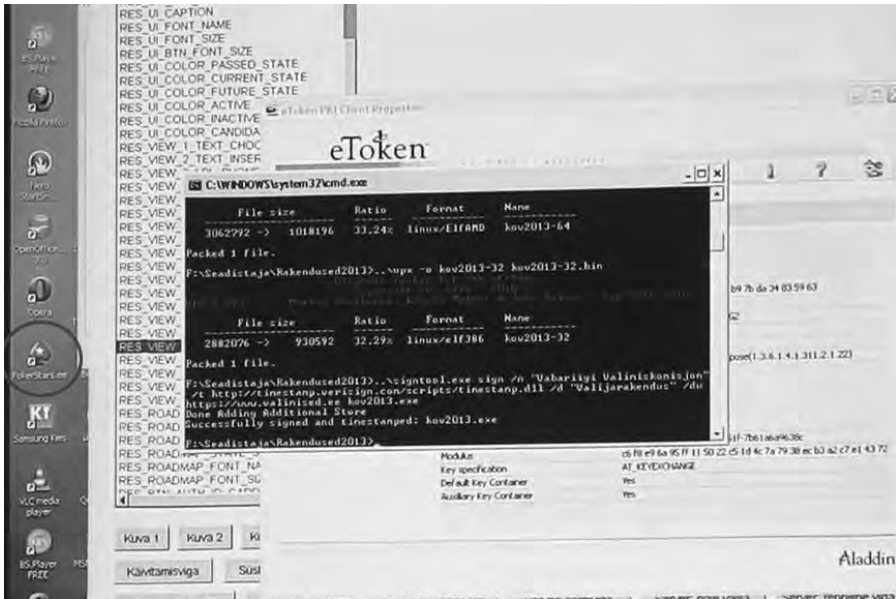


Figure 7.13: Researchers noted dangerous lapses in operational security during Estonia’s 2013 election. Here, officials digitally sign the Internet voting client for distribution to the public using a laptop that has online gambling software installed [535].

Unfortunately, Estonian public discourse concerning election technology has become dominated by partisanship, and the national leadership has opted to continue online voting for now in spite of the known security risks.

7.3.3 The New South Wales iVote System

The world’s largest deployment of online voting to-date was during the March 2015 state election in New South Wales, Australia. In this election, absentee voters had the option to use a web-based online voting system called iVote, which was developed by e-voting vendor Scytl in partnership with the New South Wales Electoral Commission. Over 280,000 votes were returned through iVote (about 5% of the total), exceeding the 70,090 Norwegian online votes in 2013 [522] and the 176,491 in the 2015 Estonian election [223].

Voters registered and cast their votes using websites managed by the electoral commission. Upon online registration, they were given an iVote ID number and asked to choose a six-digit PIN. These allowed them to log in to an online voting application written in JavaScript and HTML. After casting their votes, they received twelve-digit

receipt numbers. Optionally, voters could call a telephone verification service and enter their receipt numbers to hear an automated system read back their votes.

The system's design is further described in reports published by the electoral commission [423, 115]. However, no source code was made available to the public, and the published descriptions lack sufficient detail for independent experts to completely assess the system's security. Prior to the election, the commission performed its own security studies (e.g., [425, 424]), and officials declared that the vote was "... completely secret. It's fully encrypted and safeguarded, it can't be tampered with" [39].

While online voting in the March 2015 election was underway, Vanessa Teague of the University of Melbourne and this author performed an independent, uninvited security analysis of public portions of the iVote system [282]. They discovered critical security flaws that had been overlooked by the commission's analyses and testing.

The central flaw that Halderman and Teague found had to do with iVote's use of HTTPS. The system relied on HTTPS to deliver the voting web application to users' browsers without tampering. Unfortunately, configuring a web server to use HTTPS correctly is nontrivial—operators must provision a browser-trusted certificate, select appropriately strong cryptographic ciphersuites and ensure that older, insecure versions of TLS and SSL are disabled, among other details [401].

The researchers tested the main iVote server, `cvs.ivote.nsw.gov.au`, and found that it used a safe HTTPS configuration. However, iVote also included third-party JavaScript from an external server, `ivote.piwikpro.com`, as shown in Figure 7.14. (Piwik is an analytics tool used to track site visitors.) The PiwikPro server turned out to have poor HTTPS security and was vulnerable to several attacks.

One of these vulnerabilities was the FREAK attack [102, 209], short for Factoring RSA Export Keys. FREAK is a TLS vulnerability that was publicly disclosed on March 3, 2015, less than two weeks before the iVote system opened. Halderman and Teague showed that a network-based man-in-the-middle could exploit FREAK to impersonate the Piwik server and inject malicious JavaScript into the iVote web application. Many popular browsers were susceptible to FREAK, including Internet Explorer, Safari, and Chrome for Mac OS and Android [209]. Although patches were released for most browsers about a week before iVote opened, it is likely that many users had not yet applied the updates.

The Piwik server was also vulnerable to an even more powerful TLS attack that affected *all* popular browsers: the Logjam attack [50], which was publicly disclosed two months after the election. The researchers were aware of this flaw at the time because Halderman was part of the team that discovered it, but they could not talk about it publicly since responsible disclosure was still underway. In other words, the voting security researchers had a zero-day TLS vulnerability that would have allowed them to attack any voter's iVote session.

Halderman and Teague showed that these flaws would allow an attacker to violate ballot privacy or steal votes. To do this, the attacker would intercept connections from

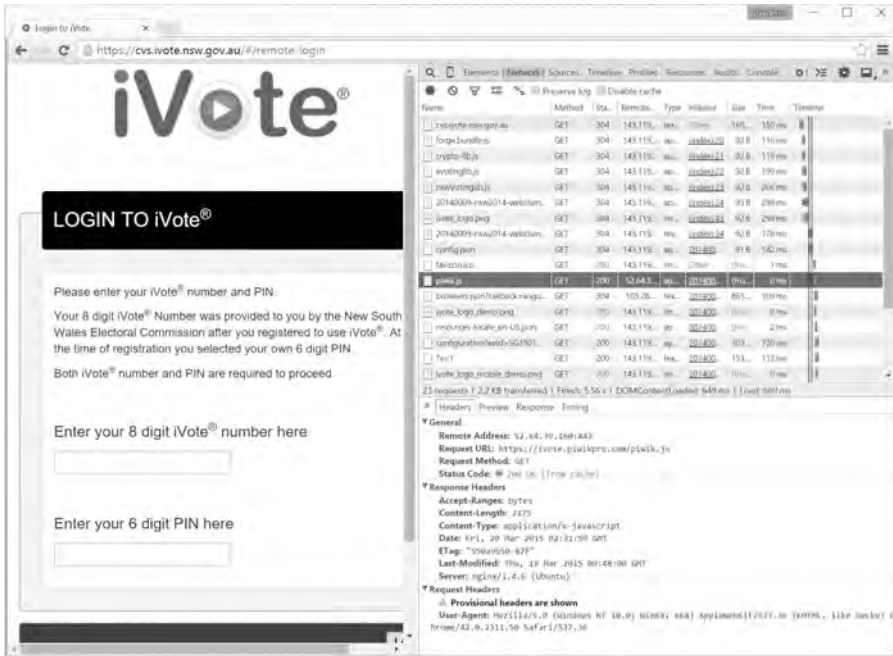


Figure 7.14: The New South Wales Internet voting website imported JavaScript from a third-party server that suffered from several TLS vulnerabilities [282].

the voter’s browser to the Piwik server and use FREAK or Logjam to replace the real Piwik code with malicious JavaScript, as shown in Figure 7.15. This code would execute in the context of the user’s iVote session, where it could arbitrarily change the operation of the iVote web application. The researchers demonstrated malware that would steal the voter’s PIN and the content of their secret ballot, then substitute a different vote of the attacker’s choosing.

Although these attacks require becoming a man-in-the-middle, criminal attackers have many well-documented ways to achieve this, including compromising insecure WiFi access points, poisoning ISP DNS caches, attacking vulnerable routers, and hijacking BGP prefixes. (Such attacks are one of the main threats that HTTPS is intended to guard against.) These attacks are especially practical in the context of a large election, since the attacker can opportunistically target any insecure hosts or infrastructure in the region where voting is taking place.

Such an attack might be caught by iVote’s optional telephone-based verification system, but Halderman and Teague pointed out several ways that an attacker could circumvent this mechanism, including a variety of tricks to reduce the probability that a given voter would notice a problem. For instance, they could misdirect the voter to a fake verification phone number that reads back the voter’s intended choices.

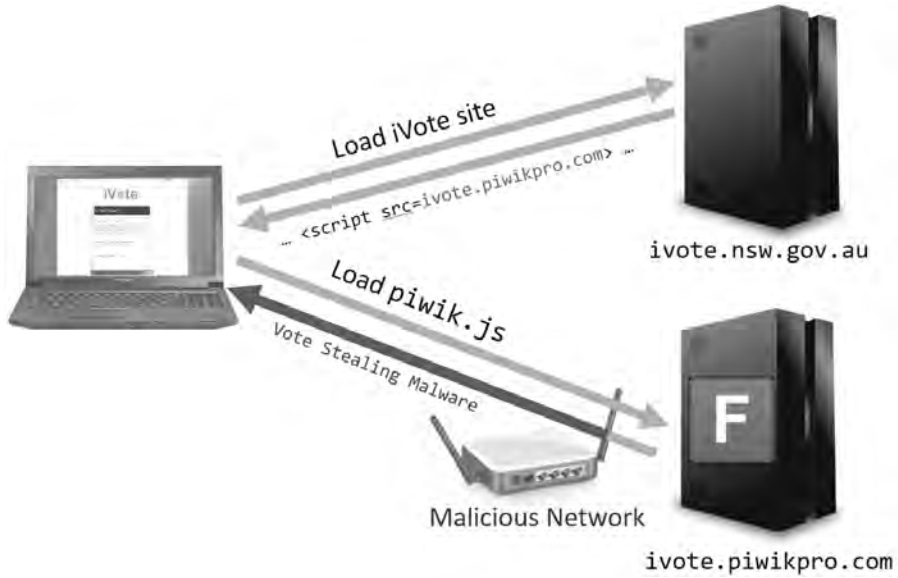


Figure 7.15: Researchers showed that a man-in-the-middle attacker could exploit the vulnerabilities in the Australian iVote system to inject vote-stealing code [282].

Even more simply, the attacker could delay submitting the real vote and displaying a receipt number for a few seconds, in hopes that the voter does not intend to verify and leaves the website. If the voter navigates away without seeing a receipt number, there is no chance to verify, and the attacker can substitute a fraudulent vote.

Another way to circumvent verification would be to mount a variant of the “clash” attack [357]. The attacker would intercept a voter’s online registration session and assign him or her the iVote ID and PIN of a like-minded person who had already voted, preferably one who cast a simple vote likely to be repeated. Later, if the victim’s choices match those of the first voter, all of the verification will look right to both voters. The attacker can safely reuse the target voter’s registration credentials to get a new iVote ID and PIN and cast an arbitrary vote. While the registration server itself was protected by HTTPS, the main iVote gateway from which voters reached it ran plain HTTP. This gave a man-in-the-middle attacker the opportunity to misdirect registration attempts to a fake site of the attacker’s choosing.

Of course, the verification design also compromises privacy and exposes voters to coercion, since it makes it easy to prove how you voted to anyone just by handing over your credentials and receipt number. Furthermore, the design does not provide strong evidence to support or disprove voter complaints, so it is difficult to distinguish an attack from the baseline level of complaints due to voter error.

After confirming that the attack was possible, Halderman and Teague notified the authorities of the vulnerability by contacting the Australian CERT [282]. The electoral commission responded the next day by modifying the iVote server configuration to disable Piwik. After the vulnerabilities were removed, the researchers made their findings public [551, 550] and later published a formal paper [282].

By the time the vulnerability was fixed, online voting had been taking place for five days, and 66,000 votes had already been cast. The closest seat in the parliamentary results was eventually decided by a margin of 3,177 votes, less than 5% of this number [422]. While we may never know whether anyone exploited the opportunity for electoral fraud, we know the opportunity was there—and if Internet voting continues to be deployed on such large scales, it will not be the last opportunity.

7.4 Conclusion

Democracy relies on voters having well-founded trust in the processes used to collect and count their votes. Unfortunately, when it comes to real-world e-voting systems, the case studies in this chapter provide abundant grounds for skepticism.

In light of such results, restoring trust will require shifting the burden of proof from the skeptics to system designers and operators. E-voting systems need to be designed to produce sufficient evidence to convince rationally skeptical observers that the outcome is correct. This could take the form of observable physical records (such as paper ballots) or cryptographic proofs, so long as they can be publicly audited to the necessary levels of assurance.

Yet evidence in support of the outcome is not sufficient. It is preferable to prevent problems with the outcome, rather than merely detecting them. And many other problems discussed in this chapter would not be caught by auditing such evidence—for example, attempts to disrupt the election or to violate voter privacy. To safeguard against these, e-voting systems need to be engineered to a level of security quality far greater than that of typical information technology systems, on par with other kinds of critical infrastructure. Elections should provide convincing evidence that this level of quality has been achieved, by being transparent about their security measures and engineering processes, and by making source code available for public review.

As we have seen, typical real-world e-voting systems are not designed to provide these kinds of evidence—of the integrity of the outcome or of the quality of their engineering. In most cases, they rely fundamentally on the correct and secure operation of secret software that is tasked with collecting and counting votes in secret.

One promising way forward is to hasten the transition of end-to-end verification (Chapter 8) from research to practice. Most of the fielded systems described in this chapter look quite different from the e-voting systems proposed by researchers that will be discussed later in this book. On one hand, real-world e-voting systems tend to be less sophisticated—sometimes their functionality amounts to little more than

counting button presses. On the other hand, unlike research prototypes, they have to cope with practical consideration ranging from price competition to the necessity of being usable enough for non-technical voters and poll workers. We need more research—and closer collaboration between researchers and election officials—to develop verifiable e-voting systems that are suitable for real-world use.