Attacking the Washington, D.C. Internet Voting System

Scott Wolchok, Eric Wustrow, Dawn Isabel, and J. Alex Halderman

The University of Michigan, Ann Arbor
{swolchok,ewust,dki,jhalderm}@umich.edu

Abstract. In 2010, Washington, D.C. developed an Internet voting pilot project that was intended to allow overseas absentee voters to cast their ballots using a website. Prior to deploying the system in the general election, the District held a unique public trial: a mock election during which anyone was invited to test the system or attempt to compromise its security. This paper describes our experience participating in this trial. Within 48 hours of the system going live, we had gained near-complete control of the election server. We successfully changed every vote and revealed almost every secret ballot. Election officials did not detect our intrusion for nearly two business days — and might have remained unaware for far longer had we not deliberately left a prominent clue. This case study — the first (to our knowledge) to analyze the security of a government Internet voting system from the perspective of an attacker in a realistic pre-election deployment — attempts to illuminate the practical challenges of securing online voting as practiced today by a growing number of jurisdictions.

Keywords: Internet voting, e-voting, penetration testing, case studies

1 Introduction

Conducting elections for public office over the Internet raises grave security risks. A web-based voting system needs to maintain both the integrity of the election result and the secrecy of voters' choices, it must remain available and uncompromised on an open network, and it has to serve voters connecting from untrusted clients. Many security researchers have cataloged threats to Internet voting (e.g. [11,15]), even as others have proposed systems and protocols that may be steps to solutions someday (e.g. [6,12]); meanwhile, a growing number of states and countries have been charging ahead with systems to collect votes online. Estonia [1] and Switzerland [2] have already adopted online voting for national elections. As of 2010, 19 U.S. states employed some form of Internet voting [5], and at least 12 more were reportedly considering adopting it [4].

Among the jurisdictions considering Internet voting, one of the most enthusiastic proponents was the District of Columbia. In 2010, the Washington, D.C. Board of Elections and Ethics (BOEE) embarked on a Federally-funded pilot project that sought to allow overseas voters registered in the District to vote

over the web starting with the November 2010 general election [16]. Though the D.C. system, officially known as the "D.C. Digital Vote-by-Mail Service," was technologically similar to parallel efforts in other states, BOEE officials adopted a unique and laudable level of transparency. The system was developed as an open source project, in partnership with the nonprofit Open Source Digital Voting (OSDV) Foundation [3]. Most significantly, prior to collecting real votes with the system, the District chose to operate a mock election and allow members of the public to test its functionality and security.

We participated in this test, which ran for four days in September and October 2010. Our objective was to approach the system as real attackers would: starting from publicly available information, we looked for weaknesses that would allow us to seize control, unmask secret ballots, and alter the outcome of the mock election. Our simulated attack succeeded at each of these goals and prompted the D.C. BOEE to discontinue its plans to deploy digital ballot return in the November election.

In this paper, we provide a case study of the security of an Internet voting system that, absent our participation, might have been deployed in real elections. Though some prior investigations have analyzed the security of proposed Internet voting systems by reviewing their designs or source code, this is the first instance of which we are aware where researchers have been permitted to attempt attacks on such a system in a realistic deployment intended for use in a general election.

We hope our experiences with the D.C. system will aid future research on secure Internet voting. In particular, we address several little-understood practical aspects of the problem, including the exploitability of implementation errors in carefully developed systems and the ability of election officials to detect, respond, and recover from attacks. Our successful penetration supports the widely held view among security researchers that web-based electronic voting faces high risks of vulnerability, and it cautions against the position of many vendors and election officials who claim that the technology can readily be made safe.

The remainder of this paper is organized as follows: Section 2 introduces the architecture and user interface of the Digital Vote-By-Mail System. In Section 3, we describe how we found and exploited vulnerabilities in the web application software to compromise the mock election. Section 4 describes further vulnerabilities that we found and exploited in low-level network components. Section 5 discusses implications of our case study for other Internet voting systems and future public trials. We survey related work in Section 6 and conclude in Section 7.

2 Background: The D.C. Digital Vote-By-Mail System

Architecture The Digital Vote-by-Mail (DVBM) system is built around an open-source web application[1] developed in partnership with the D.C. BOEE by the OSDV Foundation's TrustTheVote project[2]. The software uses the popular Ruby on Rails framework and is hosted on top of the Apache web server and the

[1] http://github.com/trustthevote/DCdigitalVBM/
[2] http://trustthevote.org

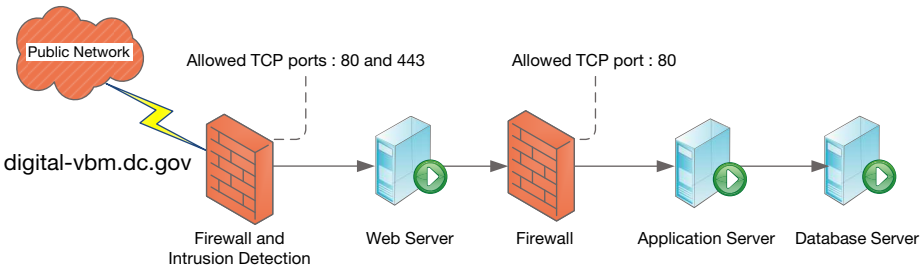


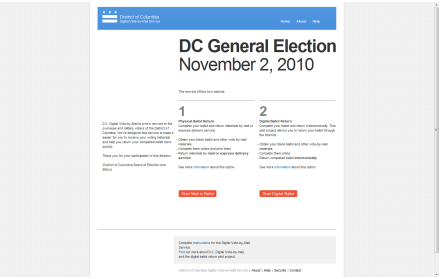
Fig. 1: **Network architecture** — The front-end web server receives HTTPS requests from users and reverse-proxies them to the application server, which hosts the DVBM election software and stores both blank and completed ballots. A MySQL database server stores voter credentials and tracks voted ballots. Multiple firewalls reduce the attack surface and complicate attacks by disallowing outbound TCP connections. The intrusion detection system in front of the web server proved ineffective, as it was unable to decrypt the HTTPS connections that carried our exploit. (Adapted from http://www.dcboee.us/DVM/Visio-BOEE.pdf.)

MySQL relational database. Global election state (such as registered voters' names, addresses, hashed credentials, and precinct-ballot mappings, as well as which voters have voted) is stored in the MySQL database. Voted ballots are encrypted and stored in the filesystem. User session state, including the user ID and whether the ballot being cast is digital or physical, is stored in an encrypted session cookie on the user's browser.

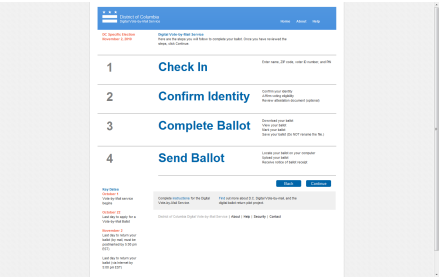
Electronic ballots are served as PDF files which voters fill out using a PDF reader and upload back to the server. To safeguard ballot secrecy, the server encrypts completed ballots with a public key whose corresponding private key is held offline by voting officials. Encrypted ballots are stored on the server until after the election, when officials transfer them to a non-networked computer (the "crypto workstation"), decrypt them using the private key, and print them for counting alongside mail-in absentee ballots.

Figure 1 shows the network architecture deployed for the mock election. HTTPS web requests are interpreted by the web server over TCP port 443. The web server then performs the HTTP request on the user's behalf to the application server, which runs the DVBM application software. The web server, application server, and a MySQL database server all run Linux. Firewalls prevent outbound connections from the web and application servers. Since the web server and application server run on separate machines, a compromise of the application server will not by itself allow an attacker to steal the HTTPS private key.

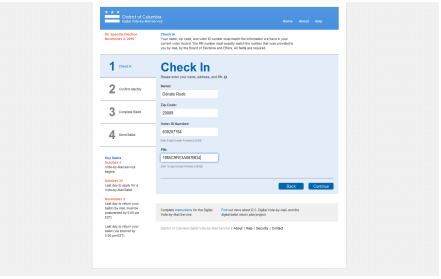
Voter experience The DVBM system was intended to be available to all military and overseas voters registered in the District. Months prior to the election, each eligible voter received a letter by postal mail containing credentials for the system. These credentials contained the voter ID number, registered name, residence ZIP code, and a 16-character hexadecimal personal identification number (PIN). One



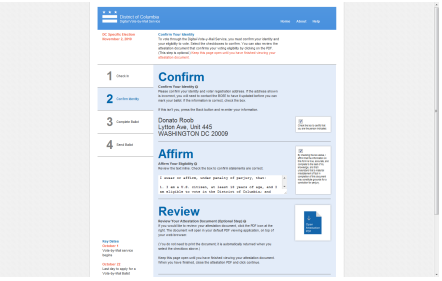
(a) Select online or postal voting



(b) Overview of steps

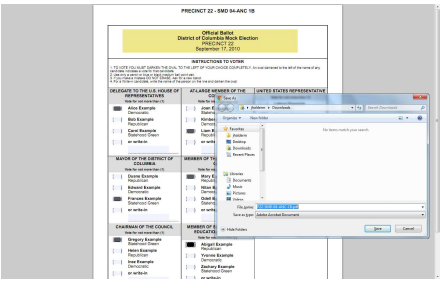


(c) Authenticate with voter ID / PIN

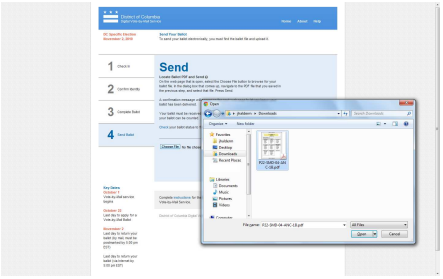


(d) “Affirm” identity

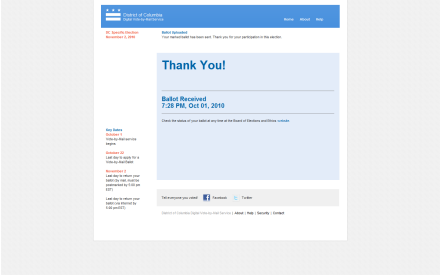
(e) Download blank ballot



(f) Mark ballot in PDF reader and save



(g) Upload completed ballot



(h) “Thank you” screen

Fig. 2: **Screenshots of the D.C. voting system** show a typical voter’s workflow. After opting to digitally return the ballot (a), the voter receives instructions (b) then enters credentials provided by postal mail (c) and attests to his identity (d). He then downloads a PDF file of the ballot (e), selects candidates and saves the file (f), and uploads the completed ballot to the server (g), which returns a confirmation screen (h).

instance of this letter is shown in Figure 5. The letters instructed voters to visit the D.C. Internet voting system website, which guided them through the voting process.

Figure 2 depicts the steps of the online voting user interface. Upon arrival, the voter selects between a digital or postal ballot return. Next, the voter is presented with an overview of the voting process. The voter then logs in with the credentials provided in the mail, and confirms his or her identity. Next, the voter is presented with a blank ballot in PDF format. In the postal return option, the voter simply prints out the ballot, marks it, and mails it to the provided address. For the digital return, the voter marks the ballot electronically using a PDF reader, and saves the ballot to his or her computer. The voter then uploads the marked ballot to the D.C. Internet voting system, which reports that the vote has been recorded by displaying a "Thank You" page. If voters try to log in a second time to cast another ballot, they are redirected to the final Thank You page, disallowing them from voting again.

3 Attacking the Web Application

In this section, we describe vulnerabilities we discovered and exploited in the DVBM server application. Our search for vulnerabilities was primarily conducted by manual inspection of the web application's source code, guided by a focus on the application's attack surface. In particular, we concentrated on voter login, ballot upload and handling, database communication, and other network activity. The fact that the application was open source expedited our search, but motivated attackers could have found vulnerabilities without the source code using alternative methods. For example, one might attack voter login fields, ballot contents, ballot filenames, or session cookies, by either fuzzing or more direct code injection attacks such as embedding snippets of SQL, shell commands, and popular scripting languages with detectable side effects.

3.1 Shell-injection vulnerability

After a few hours of examination, we found a shell injection vulnerability that eventually allowed us to compromise the web application server. The vulnerability was located in the code for encrypting voted ballots uploaded by users. The server stores uploaded ballots in a temporary location on disk, and the DVBM application executes the `gpg` command to encrypt the file, using the following code:

```
run("gpg", "--trust-model always -o
    \"#{File.expand_path(dst.path)}\" -e -r
    \"#{@recipient}\" \"#{File.expand_path(src.path)}\"")
```

The `run` method invoked by this code concatenates its first and second arguments, collapses multiple whitespace characters into single characters, and then executes the command string using Ruby's backtick operator, which passes

the provided command to the shell. The Paperclip[3] Rails plugin, which the application uses to handle file uploads, preserves the extension of the uploaded ballot file, and no filtering is performed on this extension, so the result of `File.expand_path(src.path)` is attacker controlled. Unfortunately, in the Bash shell used on the server, double quotes do not prevent the evaluation of shell metacharacters, and so a ballot named `foo.$(cmd)` will result in the execution of `cmd` with the privileges of the web application.

The current release of the Paperclip plugin at the time of our analysis (late September 2010) was version 2.3.3. It appears that a similar vulnerability in Paperclip's built-in `run` method was fixed on April 30, 2010[4]. The first release containing the patch was version 2.3.2, which was tagged in the Paperclip Git repository on June 8, 2010. The degree of similarity between the DVBM application's custom `run` method and the Paperclip `run` method suggests that the DVBM application's implementation is a custom "stripped-down" version of Paperclip's, contrary to the D.C. BOEE's assertion that "a new version of [Paperclip] that had not been fully tested had been released and included in the deployed software" and "did not perform filename checks as expected." [14] Indeed, if DVBM had used the Paperclip `run` method together with an up-to-date version of the Paperclip library, this specific vulnerability would not have been included in the software. The resulting attack serves as a reminder that a small, seemingly minor engineering mistake in practically any layer of the software stack can result in total system compromise.

When we tested the shell injection vulnerability on the mock election server, we discovered that outbound network traffic from the test system was filtered, rendering traditional shellcode and exfiltration attempts (e.g., `nc umich.edu 1234 < /tmp/ballot.pdf`) ineffective. However, we were able to exfiltrate data by writing output to the `images` directory on the compromised server, where it could be retrieved with any HTTP client. To expedite crafting our shell commands, we developed an exploit compiler and a shell-like interface that, on each command, creates a maliciously named ballot file, submits the ballot to the victim server, and retrieves the output from its chosen URL under `/images`.

Interestingly, although the DVBM system included an intrusion detection system (IDS) device, it was deployed in front of the web server and was not configured to intercept and monitor the contents of the encrypted HTTPS connections that carried our attack. Although configuring the IDS with the necessary TLS certificates would no doubt have been labor intensive, failure to do so resulted in a large "blind spot" for the D.C. system administrators.

3.2 Attack payloads

We exploited the shell injection vulnerability to carry out several attacks that illustrate the devastating effects attackers could have during a real election if they gained a similar level of access:

[3] https://github.com/thoughtbot/paperclip

[4] The patch in question is available at https://github.com/thoughtbot/paperclip/commit/724cc7. It modifies `run` to properly quote its arguments using single quotes.

Stealing secrets We retrieved several cryptographic secrets from the application server, including the public key used for encrypting ballots. Despite the use of the term "public key," this key should actually be kept secret, since it allows attackers to substitute arbitrary ballots in place of actual cast ballots should they gain access to the storage device. We also gained access to the database by finding credentials in the bash history file (`mysql -h 10.1.143.75 -udvbm -pP@ssw0rd`).

Changing past and future votes We used the stolen public key to replace all of the encrypted ballot files on the server at the time of our intrusion with a forged ballot of our choosing. In addition, we modified the ballot-processing function to append any subsequently voted ballots to a `.tar` file in the publicly accessible `images` directory (where we could later retrieve them) and replace the originals with our forged ballot. Recovery from this attack 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.

Revealing past and future votes One of the main goals of a voting system is to protect ballot secrecy, which means not only preventing an attacker of the system from determining how a voter voted, but also preventing a voter from willingly revealing their cast ballot to a third party, even if they are coerced or incentivized to do so. While any absentee system that allows voters to vote where they choose allows a voter to reveal his or her vote voluntarily, our attack on the D.C. system allowed us to violate ballot secrecy and determine how nearly all voters voted.

Our modifications to the ballot processing function allowed us to learn the contents of ballots cast following our intrusion. Revealing ballots cast prior to our intrusion was more difficult, because the system was designed to store these ballots in encrypted form, and we did not have the private key needed to decipher them. However, we found that the Paperclip Rails plugin used to handle file uploads stored each ballot file in the `/tmp` directory before it was encrypted. The web application did not remove these unencrypted files, allowing us to recover them. While these ballots do not explicitly specify the voter's ID, they do indicate the precinct and time of voting, and we were able to associate them with voters by using login events and ballot filenames recorded in the server application logs. Thus, we could violate the secret ballot for past and future voters.

Discovering that real voter credentials were exposed In addition to decrypted ballots, we noticed that the `/tmp` directory also contained uploaded files that were not PDF ballots but other kinds of files apparently used to exercise error handling code during testing. To our surprise, one of these files was a 937 page PDF document that contained the instruction letters sent to each of the registered voters, which included the real voters' credentials for using the system. The first page of this file is shown in Figure 5. These credentials would have allowed us (or anyone else who penetrated the insecure server) to cast votes as these citizens in

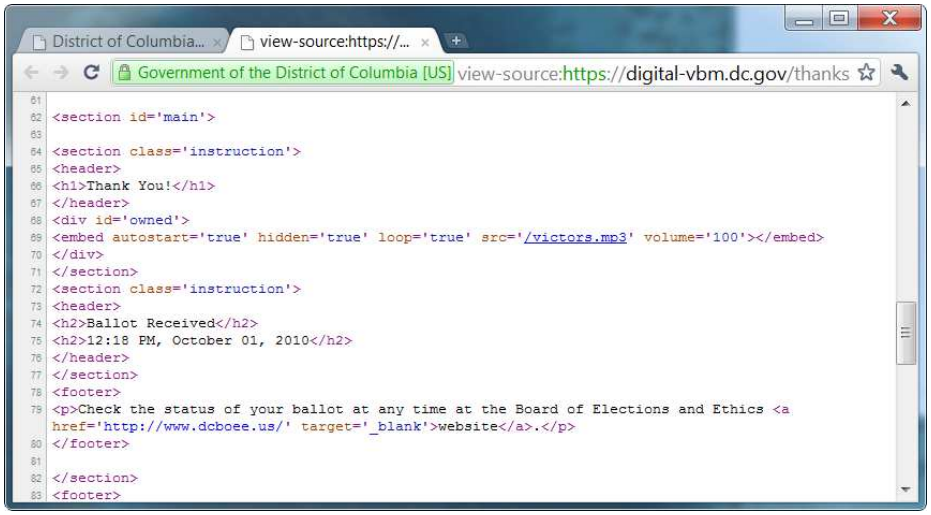


Fig. 3: **Musical "calling card"** — We modified the Thank You page that appears at the end of the voting process to play the University of Michigan fight song, "The Victors." Nevertheless, it took two business days for officials to become aware of the infiltration. Our additions appear on lines 68–70 above.

the real D.C. election that was to begin only days after the test period. Since the system requires that these credentials be delivered via postal mail, it would be infeasible for officials to send updated ones to the voters in time for the election.

Hiding our tracks We were able to hide the evidence of our intrusion with moderate success. We downloaded the DVBM application logs, altered them to remove entries corresponding to our malicious ballot uploads, and, as our final actions, overwrote the application log with our sanitized version and removed our uploaded files from the `/tmp` and `images` directories.

Our calling card To make our control over the voting system more tangible to nontechnical users, we left a "calling card" on the final screen of the digital voting workflow: we uploaded a recording of "The Victors" (the University of Michigan fight song) and modified the confirmation page to play this recording after several seconds had elapsed, as shown in Figure 3. We hoped that this would serve as a clear demonstration that the site had been compromised, while remaining discreet enough to allow the D.C. BOEE system administrators a chance to exercise their intrusion detection and response procedures.

3.3 Other vulnerabilities and potential attacks

Our intention in participating in the trial was to play the role of a real attacker. Therefore, once we had found vulnerabilities that allowed us to compromise the system, our attention shifted to understanding and exploiting these problems. However, along the way we did uncover several additional vulnerabilities in the

DVBM web application that were not necessary for our attack. Two key system deployment tasks were not completed. First, the set of test voter credentials was not regenerated and was identical to those included in the public DVBM Git repository. While the test voter credentials were fictitious, their disclosure constituted a security problem because public testers were asked to contact the D.C. BOEE for credentials, implying that the number of credentials available to each test group was to be limited.

Similarly, the encryption key used for session cookies was unchanged from the default key published in the repository. Disclosure of the key exacerbated a second vulnerability: rather than using the Rails-provided random `session_id` to associate browser sessions with voter credentials, the DVBM developers used the `rid` value, which corresponds to the automatically incremented primary key of the registration table in the system's MySQL database. This means every integer less than or equal to the number of registered voters is guaranteed to correspond to some voter. Combining this with the known encryption key results in a *session forgery* vulnerability. An attacker can construct a valid cookie for some voter simply by choosing an arbitrary valid `rid` value. This vulnerability could have been used to submit a ballot for every voter.

Our attack was expedited because the DVBM application user had permission to write the code of the web application. Without this permission, we would have had to find and exploit a local privilege escalation vulnerability in order to make malicious changes to the application. In fact, the version of the Linux kernel running on the application server (2.6.18-194.11.4.el5) had a known local root exploit (CVE-2010-3081) that could have allowed us to gain root privileges on the machine. As we were able to carry out our attacks as the web application user, we did not need to use this exploit.

We also identified other attack strategies that we ultimately did not need to pursue. For instance, the "crypto workstation" (see Section 2) used for decrypting and tabulating ballots is not directly connected to the Internet, but attackers may be able to compromise it by exploiting vulnerabilities in PDF processing software. PDF readers are notoriously subject to security vulnerabilities; indeed, the Crypto Workstation's lack of Internet connectivity may reduce its security by delaying the application of automated updates in the time leading up to the count. If the Crypto Workstation is compromised, attackers would likely be able to rewrite ballots. Furthermore, the web application allowed uploaded PDF ballots to contain multiple pages. If the printing is done in an automated fashion without restricting printouts to a single page, an attacker could vote multiple ballots.

4 Attacking the Network Infrastructure

In addition to the web application server, we were also able to compromise network infrastructure on the pilot network. This attack was independent from our web application compromise, yet it still had serious ramifications for the real election and showed a second potential path into the system.

Prior to the start of the mock election, the D.C. BOEE released a pilot network design diagram that showed specific server models, the network configuration connecting these servers to the Internet, and a CIDR network block (8.15.195.0/26). Using Nmap, we discovered five of the possible 64 addresses in this address block to be responsive. By using Nmap's OS fingerprinting feature and manually following up with a web browser, we were able to discover a Cisco router (8.15.195.1), a Cisco VPN gateway (8.15.195.4), two networked webcams (8.15.195.11 and 8.15.195.12), and a Digi Passport 8 terminal server[5] (8.15.195.8).

4.1 Infiltrating the terminal server

The Digi Passport 8 terminal server provides an HTTP-based administrative interface. We were able to gain access using the default root password (`dbps`) obtained from an online copy of the user manual. We found that the terminal server was connected to four enterprise-class Cisco switches (which we surmised corresponded to the switches shown on the network diagram provided by the BOEE) and provided access to the switches' serial console configuration interfaces via telnet.

We hid our presence in the terminal server using a custom JavaScript rootkit, which we installed over an SSH session (the same account names and passwords used in the web interface were accepted for SSH). The rootkit concealed an additional account with administrator privileges, "dev," which we planned to use in case our attack was discovered and the passwords changed. We also used our SSH access to download the terminal server's `/etc/shadow` and `/etc/passwd` files for cracking using the "John the Ripper" password cracker[6]. After about 3.5 hours using the cracker's default settings, we recovered the secondary administrator password `cisco123` from a salted MD5 hash.

Evidence of other attackers When we inspected the terminal server's logs, we noticed that several other attackers were attempting to guess the SSH login passwords. Such attacks are widespread on the Internet, and we believe the ones we observed were not intentionally directed against the D.C. voting system. However, they provide a reminder of the hostile environment in which Internet voting applications must operate.

The first SSH attack we observed came from an IP address located in Iran (80.191.180.102), belonging to Persian Gulf University. We realized that one of the default logins to the terminal server (user: `admin`, password: `admin`) would likely be guessed by the attacker in a short period of time, and therefore decided to protect the device from further compromise that might interfere with the voting system test. We used iptables to block the offending IP addresses and changed the admin password to something much more difficult to guess. We later blocked similar attacks from IP addresses in New Jersey, India, and China.

[5] A *terminal server* is a device that attaches to other pieces of equipment and allows administrators to remotely log in and configure them.

[6] http://www.openwall.com/john/

4.2 Routers and switches

After we compromised the terminal server, we found several devices connected to its serial ports. Initially, there were four Cisco switches: a pair of Nexus 5010s and a pair of Nexus 7010s. Connecting to these serial ports through the terminal server presented us with the switches' login prompts, but previously found and default passwords were unsuccessful.

The terminal server provided built-in support for keystroke logging of serial console sessions and forwarding of logged keystrokes to a remote syslog server, which we enabled and configured to forward to one of our machines. This allowed us to observe in real time as system administrators logged in and configured the switches, and to capture the switches' administrative password, `!@#123abc`.

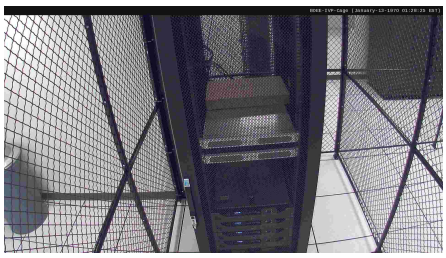
Later in the trial, four additional devices were attached to the terminal server, including a pair of Cisco ASR 9010 routers and a pair of Cisco 7606-series routers. We were again able to observe login sessions and capture passwords. At the end of the public trial, we changed the passwords on the routers and switches—effectively locking the administrators out of their own network—before alerting BOEE officials and giving them the new password.

D.C. officials later told us that the routers and switches we had infiltrated were not intended to be part of the voting system trial and were simply colocated with the DVBM servers at the District's off-site testing facility. They were, however, destined to be deployed in the core D.C. network, over which real election traffic would flow. With the access we had, we could have modified the devices' firmware to install back doors that would have given us persistent access, then later programmed them to redirect Internet voting connections to a malicious server.

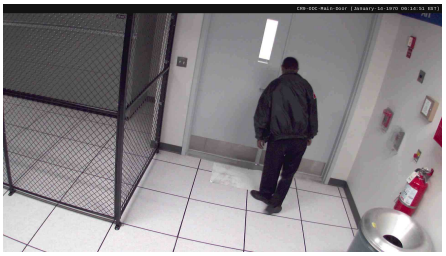
4.3 Network webcams

We found a pair of webcams on the DVBM network—both publicly accessible without any password—that showed views of the server room that housed the pilot. As shown in Figure 4, one camera pointed at the entrance to the room, and we were able to observe several people enter and leave, including a security guard, several officials, and IT staff new hardware. The second camera was directed at a rack of servers.

These webcams may have been intended to increase security by allowing remote surveillance of the server room, but in practice, since they were unsecured, they had the potential to leak information that would be extremely useful to attackers. Malicious intruders viewing the cameras could learn which server architectures were deployed, identify individuals with access to the facility in order to mount social engineering attacks, and learn the pattern of security patrols in the server room. We used them to gauge whether the network administrators had discovered our attacks—when they did, their body language became noticeably more agitated.

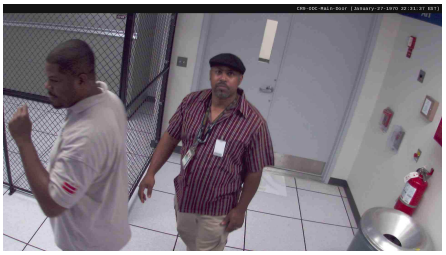


(a) Voting server rack



(b) Security guard

(c) Typical workers, before attack



(d) Workers, after learning of attack

Fig. 4: **Unsecured network surveillance cameras** gave us a real-time view into the network operations center. We could observe whether administrators made physical changes to the servers running the voting system (*a*) and monitor the frequency of patrols by security guards (*b*). We inferred that our attack had not been detected based on the relaxed body language of workers in the facility, e.g. (*c*), which changed dramatically after the BOEE learned of our intrusion (*d*).

5 Discussion

5.1 Attack detection and recovery

After we completed our attack—including our musical calling card on the "Thank You" page—there was a delay of approximately 36 hours before election officials responded and took down the pilot servers for analysis. The attack was apparently brought to officials' attention by an email on a mailing list they monitored that curiously asked, "does anyone know what tune they play for successful voters?" Shortly after another mailing list participant recognized the music as "The Victors," officials abruptly suspended the public examination period, halting the tests five days sooner than scheduled, citing "usability issues."

Following the trial, we discussed the attack with D.C. officials. They explained that they found our modifications to the application code by comparing the disk image of the server to a previous snapshot, although this required several days of analysis. They confirmed that they were unable to see our attacks in their intrusion detection system logs, that they were unable to detect our presence in the network equipment until after the trial, and that they did not discover the attack until they noticed our intentional calling card. We believe that attack

detection and recovery remain significant challenges for any Internet voting system.

5.2 Adversarial testing and mechanics of the D.C. trial

The D.C. BOEE should be commended for running a public test of their system. Their trial was a step in the right direction toward transparency in voting technology and one of the first of its kind. Nonetheless, we reiterate that adversarial testing of Internet voting applications is not necessary to show that they are likely to be weak. The architectural flaws inherent in Internet voting systems in general and the potential disastrous implications of a single vulnerability were known and expected by researchers prior to the D.C. trial [11]. We hope not to have to repeat this case study in order to highlight these limitations once again.

The key drawback to adversarial testing is that a lack of problems found in testing *does not* imply a lack of problems in the system, despite popular perception to the contrary. It is likely that testers will have more limited resources and weaker incentives than real attackers—or they may simply be less lucky. A scarcity of testers also seems to have been an issue during the D.C. trial. During our compromise of the DVBM server, we were able to view the web access logs, which revealed only a handful of attack probes from other testers, and these were limited to simple failed SQL and XSS injection attempts.

One reason for the lack of participation may have been ambiguity over the legal protections provided to testers by the BOEE. Another possible reason is that the test began on short notice—the final start date was announced only three days in advance. If such a trial must be repeated, we hope that the schedule will be set well in advance, and that legal protections for participants will be strongly in place. In addition to the short notice, the scheduled conclusion of the test was only three days before the system was planned to be opened for use by real voters. Had the test outcome been less dramatic, election officials would have had insufficient time to thoroughly evaluate testers' findings.

Despite these problems, one of the strongest logistical aspects of the D.C. trial was that access to the code—and to some extent, the architecture—was available to the testers. While some observers have suggested that this gave us an unrealistic advantage while attacking the system, there are several reasons why such transparency makes for a more realistic test. Above and beyond the potential security benefits of open source code (pressure to produce better code, feedback from community, etc.), in practice it is difficult to prevent a motivated attacker from gaining access to source code. The code could have been leaked by the authors through an explicit sale by dishonest insiders, as a result of coercion, or through a compromised developer workstation. Since highly plausible attacks such as these are outside the scope of a research evaluation, it is not only fair but realistic to provide the code to the testers.

5.3 Why Internet voting is hard

Practical Internet voting designs tend to suffer from a number of fundamental difficulties, from engineering practice to inherent architectural flaws. We feel it is

important to point them out again given the continued development of Internet voting systems.

Engineering practice Both the DVBM system and the earlier prototype Internet voting system SERVE [11] were built primarily on commercial-off-the-shelf (COTS) software (which, despite the use of the term "commercial," includes most everyday open-source software). Unfortunately, the primary security paradigm 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 voting applications due to the severe consequences of failure.

Inherited DRE threats Relatively simple Internet voting systems like D.C.'s DVBM strongly resemble direct recording electronic (DRE) voting machines, in that there is no independent method for auditing cast ballots. If the voting system software is corrupt, recovery is likely to be impossible, and even detection can be extremely difficult. DRE voting is highly susceptible to insider attacks as well as external compromise through security vulnerabilities. In previous work [7,8,10,13,17], the closed, proprietary nature of DREs has been held as an additional threat to security, since there is no guarantee that even the *intended* code is honest and correct. In contrast, the DVBM system was open source, but the public would have had no guarantee that the deployed voting system was actually running the published code.

Tensions between ballot secrecy and integrity One of the fundamental reasons that voting systems are hard to develop is that two fundamental goals of a secret ballot election—ballot secrecy and ballot integrity—are in tension. Indeed, the D.C. system attempted to protect integrity through the use of logs, backups and intrusion detection, yet these systems can help an intruder compromise ballot secrecy. Other security mechanisms put in place to protect ballot secrecy, such as encrypting completed ballots and avoiding incremental backups make detecting and responding to compromise much more difficult.

Architectural brittleness in web applications The main vulnerability we exploited resulted from a tiny oversight in a single line of code and could have been prevented by using single quotes instead of double quotes. Mistakes like this are all too common. They are also extremely hard to eradicate, not because of their complexity, but because of the multitude of potential places they can exist. If any one place is overlooked, an attacker may be able to leverage it to gain control of the entire system. In this sense, existing web application frameworks tend to be *brittle*. As our case study shows, the wrong choice of which type of quote to use—or countless other seemingly trivial errors—can result in an attacker controlling the outcome of an election.

Internet-based threats Internet voting exposes what might otherwise be a small, local race of little global significance to attackers from around the globe, who may act for a wide range of reasons varying from politics to financial gain to sheer malice. In addition to compromising the central voting server as we did, attackers can launch denial-of-service attacks aimed at disrupting the election, they can

redirect voters to fake voting sites, and they can conduct widespread attacks on voters' client machines [9]. These threats correspond to some of the most difficult unsolved problems in Internet security and are unlikely to be overcome soon.

Comparison to online banking 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 suffer a significant amount of online fraud but write it off as part of the cost of doing business; fraudulent election results cannot be so easily excused.

6 Related Work

Although this is, to the best of our knowledge, the first public penetration test of an Internet voting system scheduled for use in a general election, we are not the first to caution against the adoption of Internet voting.

The most closely related work is the 2004 security analysis of the Secure Electronic Registration and Voting Experiment (SERVE) by Jefferson et al. [11]. Like the D.C. DVBM project, SERVE was an Internet voting "pilot" that was slated for use in an actual election by absentee overseas voters. Jefferson et al. reviewed the system design and pointed out many architectural and conceptual weaknesses that apply to remote Internet voting systems in general, though they did not have an opportunity to conduct a penetration test of a pilot system. On the basis of these weaknesses, Jefferson et al. recommended "shutting down the development of SERVE immediately and not attempting anything like it in the future until both the Internet and the world's home computer infrastructure have been fundamentally redesigned." We emphatically reaffirm that recommendation. Despite incremental advances in computer security in the last eight years, the fundamental architectural flaws Jefferson et al. identified remain largely the same to this day.

More recently, Esteghari and Desmedt [9] developed an attack on the Helios 2.0 [6] open-audit Internet voting system. Their attack exploits an architectural weakness in home computer infrastructure by installing a "browser rootkit" or "man-in-the-browser attack" that detects the ballot web page and modifies votes. Esteghari and Desmedt note that Helios 3.0 is capable of posting audit information to an external web server *before* ballot submission, which can, in theory, be checked using a second trusted computer to detect the action of the rootkit, but it is not clear that such a second computer will be available or a sufficiently large number of nontechnical voters will take advantage of this audit mechanism.

7 Conclusions

Our experience with the D.C. pilot system demonstrates one of the key dangers in many Internet voting designs: one small mistake in the configuration or implementation of the central voting servers or their surrounding network infrastructure can easily undermine the legitimacy of the entire election. We expect that other fielded Internet voting systems will fall prey to such problems, especially if they are developed using standard practices for mass-produced software and websites. Even if the central servers were somehow eliminated or made impervious to external attack, Internet voting is likely to be susceptible to numerous classes of threats, including sabotage from insiders and malware placed on client machines. The twin problems of building secure software affordably and preventing home computers from falling prey to malware attacks would both have to be solved before systems like D.C.'s could be seriously considered. Although new end-to-end verifiable cryptographic voting schemes have the potential to reduce the trust placed in servers and clients, these proposals are significantly more advanced than systems like D.C.'s and may prove even more difficult for developers and election officials to implement correctly. Securing Internet voting in practice will require significant fundamental advances in computer security, and we urge Internet voting proponents to reconsider deployment until and unless major breakthroughs are achieved.

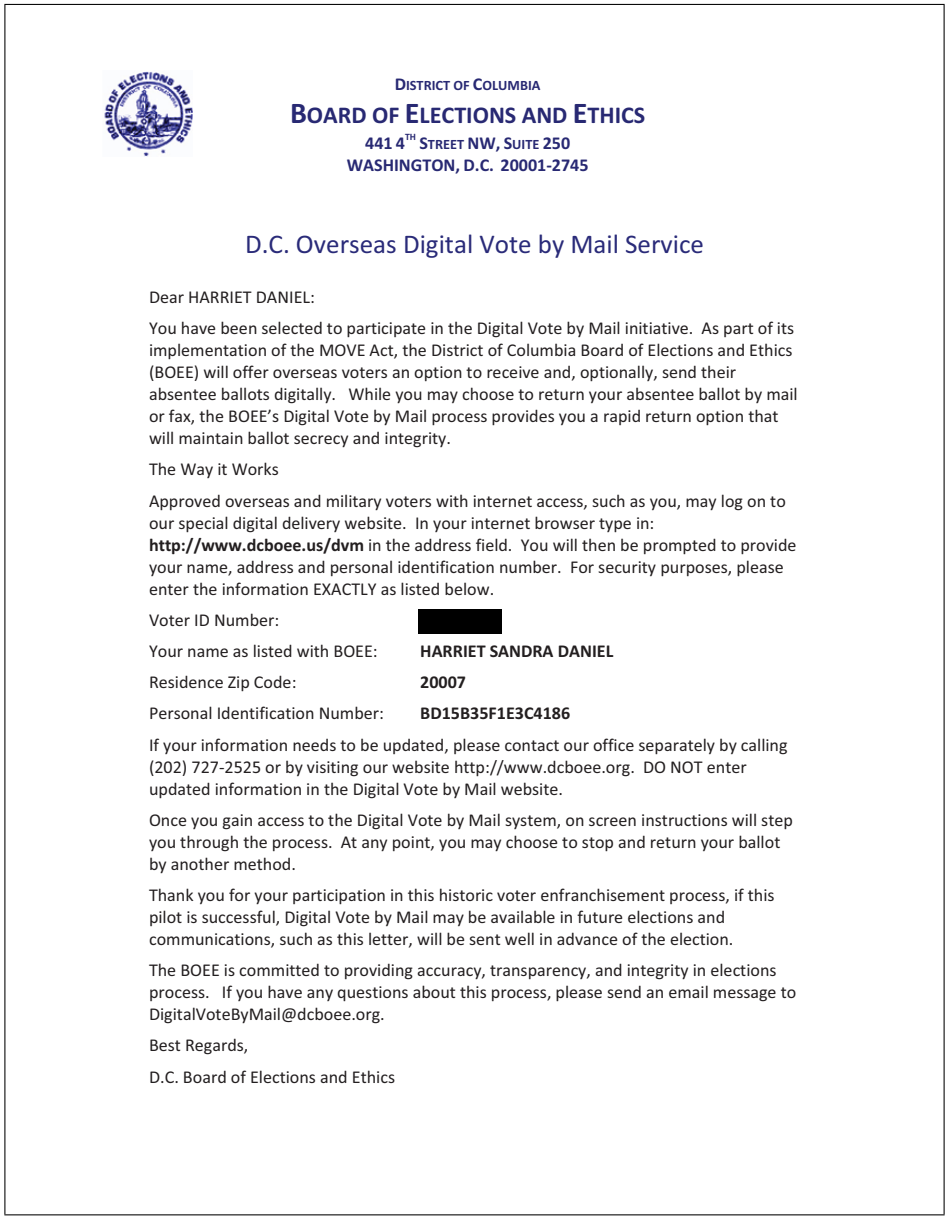
Acknowledgments

We are grateful to the many people who helped make this work possible, including Jeremy Epstein, Susannah Goodman, Nadia Heninger, David Jefferson, Bryan Sivak, Pamela Smith, David Robinson, and especially Joseph Lorenzo Hall. We thank the anonymous reviewers for their constructive feedback and Konstantin Beznosov for shepherding this paper to publication. The authors also wish to thank Rokey Suleman and Paul Stenbjorn of the D.C. BOEE for having the courage to allow the public to test its voting system.

References

1. Internet voting in Estonia. Vabariigi Valimiskomisjon. http://www.vvk.ee/public/dok/Internet_Voting_in_Estonia.pdf, Feb. 2007.
2. Uncovering the veil on Geneva's Internet voting solution. Republique Et Canton De Geneve http://www.geneve.ch/evoting/english/doc/Flash_IT_vote_electronique_SIDP_final_english.pdf, Feb. 2009.
3. District of Columbia's Board of Elections and Ethics adopts open source digital voting foundation technology to support ballot delivery. OSDV Press Release. http://osdv.org/wp-content/uploads/2010/06/osdv-press-release-final-62210.pdf, June 2010.
4. Internet voting, still in beta. The New York Times editorial. http://www.nytimes.com/2010/01/28/opinion/28thu4.html, Jan. 2010.
5. Internet voting. Verified Voting. http://www.verifiedvoting.org/article.php?list=type&type=27, May 2011.

6. Adida, B. Helios: Web-based open-audit voting. In *Proc. 17th USENIX Security Symposium* (July 2008).
7. Appel, A. W., Ginsburg, M., Hursti, H., Kernighan, B. W., Richards, C. D., Tan, G., and Venetis, P. The New Jersey voting-machine lawsuit and the AVC Advantage DRE voting machine. In *Proc. 2009 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE)* (Aug. 2009).
8. Butler, K., Enck, W., Hursti, H., McLaughlin, S., Traynor, P., and McDaniel, P. Systemic issues in the Hart InterCivic and Premier voting systems: Reflections on project EVEREST. In *Proc. 2008 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE)* (July 2008).
9. Esteghari, S., and Desmedt, Y. Exploiting the client vulnerabilities in Internet e-voting systems: Hacking Helios 2.0 as an example. In *Proc. 2010 Electronic Voting Technology Workship / Workshop on Trustworthy Elections (EVT/WOTE)* (Aug. 2010).
10. Feldman, A. J., Halderman, J. A., and Felten, E. W. Security analysis of the Diebold AccuVote-TS voting machine. In *Proc. 2007 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE)* (Aug. 2007).
11. Jefferson, D., Rubin, A. D., Simons, B., and Wagner, D. A security analysis of the secure electronic registration and voting experiment (SERVE). http://servesecurityreport.org/paper.pdf, Jan. 2004.
12. Kiayias, A., Korman, M., and Walluck, D. An Internet voting system supporting user privacy. In *22nd Annual Computer Security Applications Conference*.
13. Kohno, T., Stubblefield, A., Rubin, A. D., and Wallach, D. S. Analysis of an electronic voting system. In *IEEE Symposium on Security and Privacy* (May 2004), pp. 27–40.
14. Rokey W. Suleman, I., McGhie, K. W., Togo D. West, J., and Lowery, C. Making reform a reality: An after-action report on implementation of the Omnibus Election Reform Act. DCBOEE. http://www.dcboee.org/popup.asp?url=/pdf_files/nr_687.pdf, Feb. 2011.
15. Rubin, A. Security considerations for remote electronic voting over the Internet. http://avirubin.com/e-voting.security.html.
16. Stenbjorn, P. An overview and design rationale memo. DCBOEE. http://www.dcboee.us/dvm/DCdVBM-DesignRationale-v3.pdf, Sept. 2010.
17. Wolchok, S., Wustrow, E., Halderman, J. A., Prasad, H. K., Kankipati, A., Sakhamuri, S. K., Yagati, V., and Gonggrijp, R. Security analysis of India's electronic voting machines. In *Proc. 17th ACM Conference on Computer and Communications Security (CCS)* (Oct. 2010).



DISTRICT OF COLUMBIA
BOARD OF ELECTIONS AND ETHICS
441 4TH STREET NW, SUITE 250
WASHINGTON, D.C. 20001-2745

D.C. Overseas Digital Vote by Mail Service

Dear HARRIET DANIEL:

You have been selected to participate in the Digital Vote by Mail initiative. As part of its implementation of the MOVE Act, the District of Columbia Board of Elections and Ethics (BOEE) will offer overseas voters an option to receive and, optionally, send their absentee ballots digitally. While you may choose to return your absentee ballot by mail or fax, the BOEE's Digital Vote by Mail process provides you a rapid return option that will maintain ballot secrecy and integrity.

The Way it Works

Approved overseas and military voters with internet access, such as you, may log on to our special digital delivery website. In your internet browser type in: **http://www.dcboee.us/dvm** in the address field. You will then be prompted to provide your name, address and personal identification number. For security purposes, please enter the information EXACTLY as listed below.

Voter ID Number: [redacted]

Your name as listed with BOEE: **HARRIET SANDRA DANIEL**

Residence Zip Code: **20007**

Personal Identification Number: **BD15B35F1E3C4186**

If your information needs to be updated, please contact our office separately by calling (202) 727-2525 or by visiting our website http://www.dcboee.org. DO NOT enter updated information in the Digital Vote by Mail website.

Once you gain access to the Digital Vote by Mail system, on screen instructions will step you through the process. At any point, you may choose to stop and return your ballot by another method.

Thank you for your participation in this historic voter enfranchisement process, if this pilot is successful, Digital Vote by Mail may be available in future elections and communications, such as this letter, will be sent well in advance of the election.

The BOEE is committed to providing accuracy, transparency, and integrity in elections process. If you have any questions about this process, please send an email message to DigitalVoteByMail@dcboee.org.

Best Regards,

D.C. Board of Elections and Ethics

Fig. 5: **Voter instructions and credentials** — D.C. overseas voters received letters like this, containing instructions and credentials for using the online voting system. The letters, which were mailed prior to the pilot test, assert that the system would "maintain ballot secrecy and integrity." After we infiltrated the pilot server, we discovered a PDF file, apparently uploaded during testing, that contained all 937 letters sent to actual voters, including the secret credentials. (This is the first page from that file; we have redacted the voter ID number for privacy.) It would have been impossible for D.C. to provide new credentials to all voters in time for the upcoming election.