Are Patched Machines *Really* Fixed?

Ryan W. Gardner
Johns Hopkins University

Matt Bishop
University of California at Davis

Tadayoshi Kohno
University of Washington

Dozens of previously undetected software vulnerabilities are discovered in a variety of programs and systems every day.[1] Once information about a security vulnerability becomes available to a vendor, or particularly to the public, the vendor typically is expected to provide a means of remediation promptly. A common way to do this is to patch or upgrade the software. Quick, effective remediation is especially important in sensitive systems where failures or security compromises are unacceptable. What constitutes a sensitive system depends on both the application of the system and the trust placed in it.

Electronic voting systems are clearly sensitive. They're central to successful elections, which lie at democracy's foundation. Voting enables citizens to elect their public officials and voice their opinions about the laws and policies that form their society. Election outcomes often extensively affect large numbers of people. This wide influence motivates attempts to compromise elections, with far-reaching consequences when reported election results are inaccurate. So, any vulnerability in a voting system might represent a significant hazard. Once such a vulnerability has become public, it must be successfully remediated before the system's next use.

When the remediation is a patch or updated software, people generally consider the vulnerability to be removed. For example, even if organizations such as CERT (Computer Emergency Response Team) become aware of a security hole before it's public, they generally still publish details on the hole after a corresponding patch or update is released, under the assumption that the vulnerability is eliminated.[2] However, this isn't always the case, even with extremely critical systems.

We evaluated a recent version of the software for the Diebold (now Premier) Touch Screen and Optical Scan voting machines as part of a study for the Florida Department of State.[3] Here, we focus on four reported vulnerabilities, including previously studied updates that attempted to fix them. Although the software we evaluated addressed these vulnerabilities, we found that patching and even patching patches doesn't imply fixing. Instead, vulnerabilities might persist through multiple revisions, which are insufficient or introduce new vulnerabilities themselves.

Voter Cards

The voting machines we examined let voters cast a vote when they have a valid smart card. A valid smart card is therefore like a blank ballot that can be cast or a voter's ticket to cast a vote.

In 2003, a report from The Johns Hopkins and Rice Universities noted that these smart cards lacked any cryptography, including authentication of the cards themselves.[4] This meant voters could create functional voting smart cards and cast many votes with them, a notable security weakness that could be avoided with proper practices. Compuware confirmed these results.[5] In 2004, RABA Technologies conducted a study on a newer version of the machines.[6] The study pointed out that voter smart cards then contained a password, which the machine required for users to cast a vote. However, the cards' contents were neither encrypted nor digitally signed.

Several years later, when examining a newer version of the voting-machine software, we found that even if officials selected strong passwords, the protocol was still insecure. In particular, someone with a smart card reader could create a device to imitate a voting machine. (This requires a brief interaction with the voting machine first.) That person could then use the device to obtain the voter card password because the voter card authenticates itself by providing the device with its password. After obtaining the password, anyone could create unlimited voter cards at

home. A more appropriate way to conduct voter card authentication is to have the voter card use a challenge-response protocol to prove it knows the password without revealing the password itself. Many smart cards can do this.

Transferred Data Integrity

Election security can also be compromised by forging or altering vote data in transit. Following an election, vote data on a Premier voting machine must be uploaded to a back-end server for processing and tallying. Preliminary tallies might be transmitted from the polling station to the server through an untrusted medium such as a dial-up connection.

The Johns Hopkins and Rice report noted that data transmitted in this manner had no cryptographic protection. In other words, the vote tallies were sent to the server in cleartext form, and anyone with the ability to tap and interfere with the connection could modify the data while it was in transit. Alternatively, someone could simply create a device to imitate a legitimate voting machine and use it to upload arbitrary votes to the server.

Within a year or two of the report, the vendor updated its software, including what seemed to be an attempt to eliminate this vulnerability. RABA's 2004 study reported that the systems then used a standard SSL (Secure Sockets Layer) connection for this sensitive communication with the server. In other words, the data sent was encrypted and authenticated. However, something important was missing. The SSL certificates were neither signed nor authenticated by the terminals; in other words, the hosts themselves weren't authenticated. So, although it seemed an adversary couldn't eavesdrop on or corrupt an established connection, the adversary could still create machines that effectively pretend to be either the back-end server or a voting machine. As a result, man-in-the-middle attacks remained possible, along with the seemingly simpler attack where the adversary creates a device that imitates a legitimate voting machine and then uploads arbitrary data.

By July 2007, the vendor appeared to attempt to fix its fix with a new update.[3] In Touch Screen version 4.6.5, it added SSL certificates with a verification option. Although this was a significant improvement, problems remained. We found that the SSL pseudorandom-number generator was seeded with poor entropy, based on tightly bounded clock measurements. This fault allows someone with rough knowledge of the time that the machine was powered on to potentially determine a connection's encryption and authentication keys much faster than he or she would otherwise. Furthermore, we found that the back-end server would verify that it was communicating with a voting machine but did not verify which voting machine. The voting machines similarly verified that they were speaking to a server. So, an adversary who could obtain a certificate from *any* voting machine could again use that certificate to create a false voting machine imitating any of the others and upload illegitimate votes to the server. Moreover, if an adversary could acquire a certificate for any of the servers, he or she could launch a complete man-in-the-middle attack. Because the machines are often left unsupervised in polling places the night before an election, obtaining such a certificate might be feasible. Hence, even through a series of updates, the integrity of data in transit remains vulnerable.

Software Integrity

Possibly the most severe vulnerability in the voting machines concerns their software's integrity.

The Touch Screen voting machine we reviewed uses an externally accessible (but locked) PCMCIA (Personal Computer Memory Card International Association) card for storing cast votes and configurations, among other things. (All the tens of thousands of machines use the same key.[6] Alex Halderman claims that the exact key was pictured on the vendor's Web site such that you could make a duplicate solely from the picture.[7]) In 2006, Harri Hursti pointed out that if certain files were on the card when a machine was booted, the machine would use those files to install new software on itself without authentication.[8] (RABA also noted this in 2004.[6]) Shortly thereafter, Ariel Feldman and his colleagues demonstrated the power this functionality gave an attacker.[9] They implemented a voting-machine virus that would steal votes from one candidate and give them to another with minimal risk of detection. Once the virus was installed on one machine, it would infect every PCMCIA card inserted into that machine; machines later using the card would install the virus onto themselves. Because the cards can be used for transferring machine configurations and vote data, such spreading of the virus isn't unreasonable. The California Top-to-Bottom Review further detailed the possibility of propagation and emphasized this hazard's severity.[10] Finally, the virus could reside inactive on machines for an arbitrary number of years before stealing votes.

To address this critical threat, the vendor provided another update by July 2007.[3] In the updates, we found that the vendor had added signature verification to some of the PCMCIA card data. Specifically, before installing any new software from the card (including a potential virus), the voting machines first verified a digital signature on the software. It seemed this would prevent viruses that infect machines in the same way that the virus demonstrated by Feldman and his colleagues does.

However, we found that during signature verification, the voting machines didn't verify all of a number's zero padding during one of the computations. Here's a simple analogy. Alice tells Bob that she'll let him into her secret club if he can tell her the product of 6 times 7; Bob responds with 786542. Alice examines the last two digits of his answer and says "okay" but never checks that the higher-order digits are 0. Of course, with digital signatures, the computation is far more complicated. However, as a result of this error, we found a way to forge signatures on any data in a fraction of a second by allowing those supposed zero digits to take other values. (A detailed explanation of the flaw and attack appears in an external technical report.[11] We could also successfully demonstrate our attack.) So, a vote-stealing virus like the one demonstrated by Feldman and his colleagues was still possible, despite updates and patches. (A more recent study of the software has reported that this problem is now corrected in machines using the most recent software.[12])

Optical Scan System Vote Counts

The Optical Scan voting system had similar issues. Hursti demonstrated an attack that showed how someone could undetectably preload votes onto one of these machines by slightly altering data on a removable memory card.[13] His attack was enabled by multiple flaws, including unauthenticated scripts executed by the machine, unauthenticated vote counters stored on the card, and counters that allowed overflow.

The software we evaluated used a digital signature to authenticate the scripts, but the signature suffered from the same weakness as the one for the Touch Screen and could be forged. The newer version didn't allow voter counters to overflow, which prevented an adversary from setting a vote count to an initially negative value However, an adversary could still load positive votes onto the removable memory card, although election officials might be able to detect the excessive number of total votes. Finally, because the data on the memory card still wasn't effectively authenticated, an adversary could transfer all the votes for any candidate to any other candidate by making a simple modification of the card's data.

One cannot assume that patches or updates will fix the vulnerabilities they're intended to address, nor should you assume that they improve the security of a system to which they're applied. In the voting systems we examined, several updates missed critical aspects of a security vulnerability or introduced regression flaws that created new holes or reproduced old ones. These examples show that one should not blindly trust changes that claim to fix flaws in systems; one must evaluate and analyze the system as a whole.

=Fortunately, the general security problems we describe are well studied with known solutions. However, if governments are to use electronic voting machines, they must treat strong security as an essential functionality. Moreover, these issues apply to all technologies, not just electronic voting machines.

References

1. *CERT Statistics*, Computer Emergency Response Team, Software Eng. Inst., Carnegie Mellon Univ., 2009; www.cert.org/stats.
2. A. Arora et al., "An Empirical Analysis of Software Vendors Patch Release Behavior: Impact of Vulnerability Disclosure," *Information Systems Research*, 12 June 2009.
3. R. Gardner et al., *Software Review and Security Analysis of the Diebold Voting Machine Software*, tech. report, Florida Dept. of State, July 2007; http://election.dos.state.fl.us/voting-systems/pdf/SAITreport.pdf.
4. T. Kohno et al., "Analysis of an Electronic Voting System," *Proc. 2004 IEEE Symp. Security and Privacy*, IEEE Press, 2004, pp. 27–40.
5. Compuware Corp., *Direct Recording Electronic (DRE) Technical Security Assessment Report*, tech. report, Ohio Secretary of State, Nov. 2003.

6. RABA Technologies, *Trusted Agent Report Diebold AccuVote-TS Voting System*, tech. report, Dept. of Legislative Services, Maryland General Assembly, Jan. 2004.
7. A. Halderman, "Diebold Shows How to Make Your Own Voting Machine Key," blog, 23 Jan. 2007; http://freedom-to-tinker.com/blog/jhalderm/diebold-shows-how-make-your-own-voting-machine-key.
8. H. Hursti, "Critical Security Issues with Diebold TSx," Black Box Voting, May 2006; www.blackboxvoting.org/BBVreportIIunredacted.pdf.
9. A.J. Feldman, J.A. Halderman, and E.W. Felten, *Security Analysis of the Diebold AccuVote-TS Voting Machine*, tech. report, Center for Information Technology Policy, Princeton Univ., Sept. 2006; http://itpolicy.princeton.edu/voting.
10. J.A. Calandrino et al., *Source Code Review of the Diebold Voting System*, tech. report, California Secretary of State, July 2007.
11. R.W. Gardner, T. Kohno, and A. Yasinsac, *Attacking the Diebold Signature Variant—RSA Signatures with Unverified High-Order Padding*, tech. report 1007-22, Johns Hopkins Univ., Oct. 2007; http://cs.jhu.edu/~ryan/diebold_rsa_signature/gky_rsa_signature.pdf.
12. D. Gainey, M. Gerke, and A. Yasinsac, *Software Review and Security Analysis of the Diebold Voting Machine Software: Supplemental Report*, tech. report, Florida Dept. of State, Aug. 2007.
13. H. Hursti, "Critical Security Issues with Diebold Optical Scan Design," Black Box Voting, July 2005; www.blackboxvoting.org/BBVreport.pdf.

Ryan W. Gardner *is is a student in Johns Hopkins University's Department of Computer Science. Contact him at ryan@cs.jhu.edu.*

Matt Bishop *is a professor in the Department of Computer Science at the University of California, Davis and a co-director of the University's Computer Security Laboratory. Contact him at bishop@cs.ucdavis.edu.*

Tadayoshi Kohno *is an assistant professor in the University of Washington's Department of Computer Science and Engineering. Contact him at yoshi@cs.washington.edu.*

Updating and patching has become a ubiquitous part of software maintenance, with particular importance to security. It's especially crucial when the systems in question perform vital functions and security compromises might yield drastic consequences. Unfortunately, updates intended to remediate security problems are sometimes incomplete, are flawed, or introduce new vulnerability themselves. The authors present several examples of such instances in a widely used electronic voting system, a device for which security is critical. A central lesson of the study is that evaluating a system's security by examining changes between revisions is insufficient; you must evaluate and analyze the system as a whole.