Earl Barr, Matt Bishop, and Mark Gondree

Fixing Federal E-Voting Standards

Without a threat model and a system model, voting standards cannot ensure the integrity or accuracy of the voting process.

In elections throughout the U.S., electronic voting machines have failed to boot, tallied 10 times as many votes as registered voters, and drawn criticism from academics, election officials, and concerned citizens alike. High-profile exploits (such as the Hursti attack [4] and the Princeton group's Diebold virus [2]) have brought media attention to the fragility just below the surface of electronic voting systems. In light of these problems, it is perhaps an understatement to say that election systems have flaws. Yet independent testing authorities certified these very systems as meeting explicit U.S. federal standards for electronic voting machines. What went wrong?

Researchers focus on two explanations: poorly designed systems and a flawed certification process. Concerning design, researchers have shown, and experience has confirmed, that electronic voting machines do not meet reasonable expectations for correctness, availability, accessibility, and security. A large body of work proposes immediate, short-term fixes but, in every case, has found the only long-term remedy is complete system redesign. Concerning certification, researchers have pointed out that the practice of having voting machine vendors pay the independent testing authorities raises questions about the impartiality and rigor of the certification process itself. They have also decried the fact that that process inhibits the incremental improvement of the system by focusing on whether the system passes the certification test, not on what the vendor could do to improve the system. The experience of Ciber, Inc. represents a case study for the oversight and accountability problems with testing authorities, losing its accreditation to certify voting machines, as reported in the *New York Times,* January 4, 2007.

This problem is even more fundamental. The standards on which vendors base their system designs and against which the testing authorities certify the systems are flawed. These standards, promulgated first by the Federal Election Commission, then by the Election Assistance Commission (EAC), do not express a coherent set of requirements for electronic voting systems. They contain no system model or threat model. Lacking these guides, any standard is only a patchwork of ideas and requirements that fails to achieve its goals—if, indeed, these goals are clear. Without clear requirements, no design can be sound nor can any system be meaningfully certified.

Neither the government nor electronic voting system vendors have adequately addressed these design and certification flaws, let alone advanced solutions as national standards, for the simple reason that their effort has been misdirected. They need the computer science community's help and

LISA HANEY

active engagement in writing clear, sound, and testable standards.

Here, we first investigate the lacunae of the most current (2005) federal voting system standards, before offering a number of recommendations for how to improve them. The recommendations do not call for mere additions to the standards but highlight research opportunities and unanswered questions in applying system, threat, and process modeling to secure system standards.

In December 2006, the National Institute of Standards and Technology recommended not to certify paperless voting systems. Even if this recommendation is ultimately adopted, any future voting system will use electronics to at least mark and count paper ballots. Their software could, if poorly or maliciously designed, affect the outcome of elections because of the expense of auditing elections by recounting votes. Thus, research in improving the standards and certification process remains necessary to ensure the correctness and security of any future voting scheme. Improving the standards is the first step toward improving the quality of voting systems for the long term. The goal is voting machines that deliver on their promise and strengthen democracy.

The moral is that one can never verify that a system has no flaws, even if all source code is available.

Voting Standards

The U.S. government enacted the Help America Vote Act (HAVA) in October 2002. It created the EAC, whose duties include the development of voluntary guidelines that states can use to ensure their equipment meets the (rather ambiguous) HAVA requirements. In essence, the government gave the EAC the duty of making these requirements formal and precise, so they are meaningful to vendors and state election officials alike. HAVA also established several important timelines, including the January 1, 2006 deadline by which states were to meet the aforementioned federally mandated improvements to voting systems. As of August 2005, $2.5 billion had been dispersed to the states, territories, and the District of Columbia for this purpose.

The guidelines promulgated by the EAC "provide a set of specifications and requirements against which voting systems can be tested to determine if they provide all the basic functionality, accessibility, and security capabilities required to ensure the integrity of voting systems" (Vol. I, Sec. 1.1).

Although these guidelines apply to voting systems in general, we consider them in the context of electronic voting systems (such as direct recording electronic systems, or DREs, and optical scanning recording systems for paper ballots). However, much of our discussion generalizes readily to other types of voting systems as well.

The first EAC commissioners were appointed in December 2003. The EAC received $1.2 million in funding for fiscal year 2004 and $14 million for fiscal year 2005. During these years, it released a draft of the "Voluntary Voting System Guidelines," opening it to a 90-day period of public comment. At the end of 2005, the EAC released its official version of the standards. Our criticisms here reference the 2005 EAC standards but should be recognizable to those familiar with the earlier standards and drafts, which mostly overlap in content.

Problems with the Standards

In a broad sense, the inadequacy of the standards is not surprising. A basic impossibility result, Rice's theorem, states that no single checklist or algorithm can guarantee that an arbitrary computer, equivalent to a Turing machine, satisfies a nontrivial property (such as correctness and security). In practice, checklists can increase the likelihood that a

system meets some nontrivial property but only when the class of systems to which they apply consists of computers of similar architecture and configuration used with similar procedures in similar environments. When DREs are built on general-purpose computers and the environments in which they are used differ widely, as is currently the case, even the assurance a checklist can provide is undermined.

The way the standards fail in practice is far more basic. They fail to make precise the usability requirements outlined by HAVA. They confuse requirements for accurate voting with requirements for simplifying system testing. They include seemingly arbitrary specifications; for example, the acceptable error rates (Vol. I, Sec. 3.2.1) seem to have been chosen arbitrarily. They mandate impossible features; for example, they require that shared resources not leak information (Vol. I, Sec. 7.5.4), even though there is no way to prevent this leakage. Shared resources can always be used as a covert channel, though it is possible to reduce the channel's effectiveness. In short, the standards seem vague, ad hoc, arbitrary, and even sometimes unreasonable.

Public Availability

Critics of current electronic voting systems frequently point out that because these systems are proprietary, voters cannot inspect source code to ensure the software works correctly. Though true, this complaint reflects a larger issue—that the system design and implementation are known only to the manufacturer.

Consider the software in electronic voting machines. Since 1988, hidden malicious code has been identified as a problem in electronic voting [7]. Locating such code has been compared to "finding the proverbial needle in a haystack" [8] and "much harder than finding a needle in a haystack" [1]. The security community has recognized that this as such a problem, it has become the subject of exercise and recreation. For example, several university computer science courses have assigned a "Hack-A-Vote" homework to graduate students [6, 10], and the "Obfuscated V" contest turns the same concept into a game [3]. The problem is well known in other areas of computer security. For example, a Trojan horse can be placed into a compiler in such a way as to provide illicit access to a system yet be undetectable unless the executables produced by that compiler are analyzed [9]. The moral is that one can never verify that a system has no flaws, even if all source code is available.

This state of affairs reflects reality. We live with imperfect systems. Cars break down. Billing systems malfunction. Electoral processes are susceptible to problems; recall, for example, the electoral corruption of New York City during the era of Tammany Hall. Perfect voting systems do not exist. The goal is to build voting systems that are as good as possible.

Toward this end, making the source code available for public inspection enables voters with the desire and expertise to analyze the code base for flaws. It is analogous to the ability to observe all aspects of a paper-based election, except, to preserve the secrecy of their ballots, individuals mark their votes. In a paper-based election, one might observe the ballots being counted. But many electronic voting systems being used today[1] record votes and ballot images as bits on flash cards and in memory. They tally them and report totals. An observer cannot look inside a computer's memory to verify that the ballots are recorded correctly or the votes tallied correctly because the bits are not visible. Analyzing the source code and system can increase confidence that the votes are recorded correctly.

Assume that the source code for electronic voting systems, their operating systems, and all ancillary source code is available for public inspection. Presumably, experts will examine it, as researchers at the Johns Hopkins University and at Rice University did in 2003 when source code purported to be that of Diebold became publicly available. While it revealed many software flaws, source code

[1]Some jurisdictions use electronic voting systems as ballot-marking devices. However, such systems do not record votes electronically, and marked ballots are counted.

analysis alone cannot provide the level of assurance commensurate with the goals of electronic voting because these goals involve policies and procedures, as well as software assurance.

Software, and computer systems, exist in an environment involving policies and procedures. Unless they are taken into account, reviewing only the software may give a misleading idea of the security of a system. For example, using a Secure Sockets Layer (SSL)-like protocol to provide confidentiality of precinct vote data [5] as it is sent to the central counting system sounds like good security but is irrelevant. Election officials announce precinct vote data when they announce the final results of an election. The transmission of the data requires integrity and mutual authentication, not confidentiality.[2]

So, even though the software might appear to a reviewer to provide the requisite security in transferring precinct vote data, it does not, since the confidentiality it provides is not a requirement. A software security review is not enough. Even if the software is created with high-assurance techniques, we must still ask: What are we assuring that the software will do? Without meaningful standards, meaningful answers will be elusive.

System and Threat Models

The most striking deficiency of the current standards is the lack of an answer to the question "Against what threats should the system be protected?" The standards presume an implicit system model (such as when polling places and central tabulating facilities both exist, are physically separate, and data must be sent between them) without specifying any associated design requirements. Because it is implicit, the model is so unclear we have no language with which to discuss the threats, and the various players—election officials, testers, and vendors—are left to their own devices. For example, the standards imply roles (such as installer and troubleshooter) for running the electronic voting machines at precincts but do not articulate the details of the implied roles. So access control policies for managing how election staff interact with the system are entirely at the vendor's discretion. The standards' requirement that the access control policy "provide effective voting system security" (Vol. I, Sec. 7.2.1) is, again, vague and untestable, failing to identify the corresponding threats it is meant to address.

Only with the knowledge of a threat can the effectiveness of a countermeasure be ascertained. Only against a description of a system (a model) can our language about processes and procedures be meaningful. With neither, the standards communicate only an ad hoc list of requirements that are open to misinterpretation.

Returning to our earlier example, the standards' section "Telecommunications and Data Transmission" (Vol. I, Sec. 7.5) addresses data integrity during transmission directly: "[v]oting systems that use electrical or optical transmission of data shall ensure the receipt of valid vote records is verified at the receiving station." Verification at the receiving station is necessary but insufficient. A man-in-the-middle attack takes advantage of just this type of scenario, where verification is not performed by both parties. Consider this example: Bob expects a call from Alice on a public phone. Earlier, they arranged a secret code word only they would know. They agree that Bob, upon picking up the phone, will not speak until he hears it. Alice, however, has no way to verify that Bob has picked up the phone, instead of, say, Mallory. By diverting or intercepting Alice's call, Mallory can hear the secret and use it for later impersonation. One-way authentication is insufficient to ensure data integrity.

The same section (Vol. I, Sec. 7.5) also requires appropriate measures be taken to detect an "intrusive process" that may intercept the data. Is an intrusive process one running on the same computer (such as a Trojan horse) or one running on

[2]Used properly, SSL allows the client to authenticate the server and send integrity-checked messages but does not provide mutual authentication; for that, SSL requires a public key infrastructure or other key-distribution mechanism.

the network (such as a port scanner) or a physical process that might intercept data (such as a wiretap)? Versions of the standards before 2005 switch among each of these notions, but the current standards leave "intrusive process" undefined. Furthermore, even the implications of this requirement are unclear. For example, if interception is to be detected, does it ban technologies where interception cannot be detected (such as a broadcast medium like Ethernet or a wireless medium)?

As another example of a problem due to a lack of threat modeling, consider the target error rate of the section "Accuracy Requirements" (Vol. I, Sec. 4.1.1), requiring that the voting system "shall achieve a target error rate of no more than one in 10,000,000 ballot positions." The reason for the number 10,000,000 is not given; as far as the reader can tell, the required rate could just as well be no more than one in 500,000 ballot positions or no more than one in 100,000,000 ballot positions. These rates are all considerably lower than the error rates of paper ballots. The effort to formulate a system model would bring the question of acceptable error rates to the fore and thereby help determine which of these numbers provides the desired level of security and accuracy.

The requirements in the section "Protecting Transmitted Data" (Vol. I, Sec. 7.7.3) pose both motivational and interpretive problems. The standards require that "transmitted data ... needs to be protected to ensure confidentiality and integrity," citing ballot definitions, precinct counts, and the opening and closing of poll signals as examples of information that must be protected. As we discussed earlier, the ballot definition is displayed on the voting machines themselves, the precinct count is published as part of the reporting of election results, and the opening and closing of polls are observable actions. So the reason the related data must be confidential during transmission is unclear. Part b of this section, saying the "capability to transmit non-encrypted and non-authenticated information via wireless communication shall not exist," can be interpreted as either allowing the system to have wireless hardware that is kept turned off or as disallowing that hardware altogether.

The interpretation depends on the threat model. If the model assumes that people with the ability to access and enable the wireless hardware will not do so while the system is in use, then the former interpretation is acceptable; but if insiders are considered threats (either because they are untrustworthy or because they may accidentally enable or leave enabled the wireless hardware), then the latter interpretation is necessary.

Incorporating system and threat models into the standards will make clear the purpose and reasons for certain aspects of the standards that are currently nebulous. As a final and archetypal example, consider that the current standards insist voting machines pass a "system test" before each election. This requirement has led some vendors to write a simple program that does nothing more than display "System Test Passed" on start up. No one knows what a "system test" is or what it is meant to defend against; what is clear is that the machines will not be certified without this message.

Failure of the standards to include a threat model and a system model makes certification of voting systems haphazard at best, increasing the dif-

Rewriting the standards to include a threat model would enable vendors, election officials, computer scientists, as well as all citizens, to see what threats the developers of the standards consider necessary to protect against.

We need to move beyond the "patch" approach to electronic voting systems, using high-assurance techniques to design and implement systems that provide the requisite guarantees.

ficulty of developing, testing, and certifying any electronic voting systems.

Rewriting the standards to include a threat model would enable vendors, election officials, computer scientists, as well as all citizens, to see what threats the developers of the standards consider necessary to protect against. The testers would know what their tests needed to check, and additional parts of the standards could describe the limits of the testing.

The focus of academic and commercial work in electronic voting systems has been on detecting and remediating problems in existing systems. This work has been invaluable in gathering evidence of the sorry state of the art, both in the systems themselves and in the standards used to certify them. Now is the time to work on making these technologies, and the electoral process in which they are used, work at least as well as their conventional, paper-based counterparts. In particular, open source election software is not enough. The transparency of ballot design, configuration for the systems (such as access controls), and procedures for running the election will determine whether observers are able to assess the fairness of this process and the accuracy of its results.

Furthermore, the standards and the certification system must all respond to feedback from election administrators, incorporating the results of relevant research (such as error rates and ballot design). Perhaps the model CERT/CC uses to disseminate information that affects the security of computer systems could be adapted here. Finally, we need to move beyond the "patch" approach to electronic voting systems, using high-assurance techniques to design and implement systems that provide the requisite guarantees.

Computer scientists have much to contribute to the development of effective and meaningful standards for electronic voting machines. In concert with voting officials and the body politic, they can help secure these systems and, thereby, our rights as voters. C

References

1. Dill, D. Electronic voting: An overview of the problem. Talk presented to the Carter-Baker Commission on Federal Election Reform, Washington, D.C. (Apr. 18, 2005); www.verifiedvotingfoundation.org/article.php?id=5731.
2. Feldman, A., Halderman, J., and Felten, E. *Security Analysis of the Diebold AccuVote-TS Voting Machine.* Center for Information Technology Policy, Princeton University, Princeton, NJ (Sept. 13, 2006); itpolicy.princeton.edu/voting/.
3. Horn, D. *The Obfuscated V Contest* (Nov. 2, 2004); graphics.stanford.edu/~danielrh/vote/vote.html.
4. Hursti, H. *Diebold TSx Evaluation.* Black Box Voting, Renton, WA (May 11, 2006); www.blackboxvoting.org/BBVtsxstudy.pdf.
5. RABA Innovative Solution Cell. *Trusted Agent Report: Diebold AccuVote-TS Voting System.* RABA Technologies, Columbia, MD (Jan. 20, 2004); www.raba.com/press/TA_Report_AccuVote.pdf.
6. Rubin, A. *CS600.443 Project 2.* Department of Computer Science, Johns Hopkins University, Baltimore, MD (Mar. 27, 2004); www.cs.jhu.edu/~rubin/courses/sp04/proj2.txt, 2004.
7. Saltman, R. Accuracy, integrity, and security in computerized vote-tallying. *Commun. ACM 31,* 10 (Oct. 1988), 1184–1191.
8. Simons, B. A response to: The League of Women Voters of the U.S. questions and answers on direct recording electronic (DRE) voting systems (2004); www.leagueissues.org/lwvqa.html.
9. Thompson, K. Reflections on trusting trust. *Commun. ACM 27,* 8 (Aug. 1984), 761–763.
10. Wallach, D. *CS523 Hack-A-Vote Project.* Department of Computer Science, Rice University, Houston, TX (Sept. 2, 2003); www.cs.rice.edu/~dwallach/courses/comp527_f2003/voteproject.html.

Earl Barr (barre@acm.org) is a Ph.D. candidate in the Department of Computer Science at the University of California, Davis.
Matt Bishop (bishop@cs.ucdavis.edu) is a professor in the Department of Computer Science at the University of California, Davis.
Mark Gondree (gondree@cs.ucdavis.edu) is a graduate student in the Department of Computer Science at the University of California, Davis.