WRITTEN TESTIMONY OF DAVID WAGNER, PH.D.
COMPUTER SCIENCE DIVISION
UNIVERSITY OF CALIFORNIA, BERKELEY
BEFORE THE COMMITTEE ON HOUSE ADMINISTRATION, ELECTIONS
SUBCOMMITTEE
U.S. HOUSE OF REPRESENTATIVES
MARCH 15, 2007

Chairwoman Millender-McDonald, Ranking Member Ehlers, committee members, thank you for the opportunity to testify today. My name is David Wagner. I am an associate professor of computer science at U.C. Berkeley. My area of expertise is in computer security and the security of electronic voting. I have an A.B. (1995, Mathematics) from Princeton University and a Ph.D. (2000, Computer Science) from U.C. Berkeley. I have published two books and over 90 peer-reviewed scientific papers. In past work, I have analyzed the security of cellphones, web browsers, wireless networks, and other kinds of widely used information technology. I am a member of the ACCURATE center, a multi-institution, interdisciplinary academic research project funded by the National Science Foundation[1] to conduct novel scientific research on improving election technology. I am a member of the California Secretary of State's Voting Systems Technology Assessment Advisory Board and of the Election Assistance Commission's Technical Guidelines Development Committee (TGDC)[2]. I have served as a poll worker in my county, and I served as a technical advisor to my county's equipment selection committee.

In my testimony today, I will address source code disclosure, the problems it is intended to solve, and its benefits and risks. There are peculiarities in the voting system market and regulatory process that complicate the transition to the disclosure of the voting system source code. While these peculiarities require that such a transition be carefully considered and managed, it is a transition that I view as important for sound elections, for three reasons: (1) security and reliability; (2) public confidence and transparency; and (3) oversight and accountability.

A primer on source code and its the role in elections

What is source code? Source code is the human-readable representation of the instructions that control the operation of a computer. Computers are composed of hardware (the physical devices themselves) and software (which controls the operation of the hardware). The software instructs the computer how to operate; without software, the computer is useless. Source code is the human-readable form in which software is written by computer programmers. Source code is usually written in a programming language that is arcane and incomprehensible to non-specialists but, to a computer programmer, the source code is the master blueprint that reveals and determines how the machine will behave.

Source code could be compared to a recipe: just as a cook follows the instructions in a recipe step-by-step, so a computer executes the sequence of instructions found in the software source code. This is a reasonable analogy, but it is also imperfect. While a good cook will use her discretion and common sense in following a recipe, a computer follows the instructions in the source code in a mechanical and unfailingly literal way; thus, while errors in a recipe might be noticed and

[1]This work was supported by the National Science Foundation under Grant No. CNS-052431 (ACCURATE). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

[2]I do not speak for UC Berkeley, ACCURATE, the California Secretary of State, the EAC, the TGDC, or any other organization. Affiliations are provided for identification purposes only.

corrected by the cook, errors in source code can be disastrous, because the code is executed by the computer exactly as written, whether that was what the programmer intended or not. Also, computer software is vastly more complex than most recipes: while a typical recipe may contain perhaps a dozen steps and fits onto a single 3x5" index card, computer source code often contains hundreds of thousands of steps which, if printed, would fill up thousands of single-spaced 8.5x11" sheets of paper.

What does source code have to do with elections? Over the past several decades, as we have automated more and more of elections operations, elections have become increasingly reliant upon computing technology. For instance, touchscreen voting machines use computers to capture votes; paper ballots are scanned using computer-driven scanning machines; and computers tabulate and tally the votes to determine the winner. This makes the software that controls these machines of critical importance to our elections.

The source code in voting machines is in some ways analogous to the procedures provided to election workers. Procedures are instructions that are provided to people; for instance, the procedures provided to poll workers list a sequence of steps that poll workers should follow to open the polls on election morning. Source code contains instructions, not for people, but for the computers running the election; for instance, the source code for a voting machine determines the steps the machine will take when the polls are opened on election morning.

Who writes election-related software? Today, counties and states buy voting equipment from commercial vendors. These voting system vendors write most of the software in their machines. However, voting system vendors also incorporate software from third-party software vendors into their products. For instance, a voting system vendor like Diebold might license software from Microsoft for use in their touchscreen voting machine. The voting vendor might or might not receive source code to the third-party software; if they do, they normally would not have permission to redistribute this third-party source code to others. Third-party software is sometimes called COTS (commercial off-the-shelf) software, which we'll cover later.

Who sees election-related source code? Today, most voting system vendors treat any source code they write as confidential and proprietary. The vendors tightly control access to this source code. Election officials use the equipment, but they are normally not given access to its source code. Candidates, political parties, technical experts, and interested citizens are normally not given access to voting system source code, either.

Federal voting standards require voting system vendors to share their source code with a testing laboratory selected by the vendor, and the testing labs are supposed to check that the system complies with the federal standards. However, the testing labs have come under growing criticism for missing security and reliability problems in deployed voting systems, and many experts have expressed concerns about the ability of the testing labs to ensure that voting systems are fit for use[1] [2].

Most states do not receive or require access to voting source code. However, there are some exceptions[3]. Five states appear to require source code for certified voting systems prior to their use (FL, NY, TX, UT) or have the authority to demand source code at their discretion (CA). Two states go farther and require that the vendor provide source code to representatives of the major parties upon request (NC, MN). In California, three of the four major vendors have pledged that if California passes a law requiring source code disclosure to the public, they would abide by those provisions.

What is COTS? The federal standards provide a special exemption for COTS (commercial off-the-shelf) software. The standards define COTS software as third-party software that is commercially

readily available. COTS source code is exempted from inspection or analysis by the testing labs. This exemption makes it possible for voting system vendors to use software developed by third-party vendors even though they may not be able to provide that source code to the testing labs. In practice, most of the third-party software found in today's voting equipment qualifies as COTS. For this reason, people sometimes loosely use the term COTS to refer to any software from third-party vendors, even though strictly speaking these two concepts are not identical.

What is firmware? In much of the software industry, "firmware" usually refers to software that is embedded in a hardware device by the manufacturer and that cannot be modified. However, in the voting industry, the term has expanded to encompass any software that executes on any elections-related equipment. Therefore, when I refer to "software" in my testimony, it should be understood to include what the voting industry calls "firmware."

What can analysis of election-related source code reveal? Computer programmers are trained in reading and analyzing source code. A programmer can read source code and use this to tell how the machine will work on election day. Source code analysis can find many kinds of defects or problems with the design or implementation of the machine. It can help assess the reliability or accuracy or security of a voting machine. Source code analysis can also help to improve testing: tests devised with the assistance of source code analysis are usually more effective than tests devised without this access.

Many kinds of defects and problems with voting machines can only be found with access to the source code. Security, in particular, is difficult to evaluate without access to source code. These kinds of problems often cannot be detected through testing alone. In general, source code analysis is one of the most effective methods we have for assessing the security, reliability, and accuracy of voting machines.

However, source code analysis nonetheless has significant limitations: it generally cannot guarantee that a voting machine is secure, reliable, accurate, fair, or fit for use in elections. This is due to two reasons. First, it is often difficult to be certain that the source code one is analyzing is the same as what will be executed by the voting machine on election day. Second, given the complexity of election-related software, it is generally not possible to be certain that you have found all the bugs in the software, and it is generally not possible to be certain that the software will work reliably and accurately on election day. This means that source code analysis can be used to show the presence of defects in voting software, but usually it cannot convincingly demonstrate the absence of defects. Source code analysis alone is unlikely to be able to demonstrate that voting machines are trustworthy.

Source code disclosure: pros and cons

Today, candidates, election officials, experts, and interested citizens do not have a right of access to voting system source code; vendors are allowed to keep this source code secret. Should vendors be required by law to disclose their source code more broadly? I will attempt to list the advantages and disadvantages I can see of mandating source code disclosure.

Source code disclosure could follow a number of models. The important variables are (1) who will have access to the source code and (2) what will they be allowed to do with it. I don't propose a specific model here, but parts of my discussion will assume that election jurisdictions and independent experts will have access to source code and will be able to use that access to read and analyze the code.

Arguments for source code disclosure:

- *Transparency:* Historically, one of the abiding principles of election administration has been that the best way to demonstrate that the election is honest is by inviting public scrutiny and being open and transparent about all aspects of the election. When any aspect of election administration is kept secret, it invites questions about whether the secrecy is intended to cover up problems or to stifle debate.

  The trend in elections is towards automation of more and more tasks that were previously performed manually. However, the spread of automation has unintentionally come with the unfortunate side-effect of degrading transparency[4] [5] [6]. When poll workers run elections or elections official count ballots, the public can observe that the actions are being done correctly and openly, and can spot any errors or problems. However, when those same operations are performed by machines, the secrecy surrounding those machines and their programming effectively prevents the public from meaningfully observing or engaging in oversight of the process. Disclosure of voting system source code to the public would help to restore the public's ability to observe and exercise public oversight over the equipment and its role in the administration of the election[7].

- *Informing public debate:* There has recently been considerable public debate about the trustworthiness of voting machines. Some have argued that current voting machines are severely flawed; others have disputed that characterization. However, because of the secrecy surrounding voting software, advocates on both sides of the debate have often been denied access to the information that would be needed to present evidence for their position. The result is that advocates are all too often forced to argue from first principles or based on their professional judgement, rather than from hard evidence.

  Source code disclosure would make it possible to have a more informed debate on the trustworthiness of today's e-voting machines. We could expect and insist that anyone who wants to argue that the voting software from one vendor is flawed should be able to point to where exactly in the source code the flaw may be found. We could expect and insist that anyone who wants to argue that the voting software is flawless should be able to show evidence that the source code is free of flaws. This would create the opportunity for a more informed and scientific debate regarding the trustworthiness of e-voting, and it might raise the level of the debate.

- *Better evaluation:* Source code disclosure would enable independent analysis of voting machine software. Given the importance and public visibility of this topic, I expect source code disclosure would lead some of the country's best independent technical experts to analyze the source code and publish their findings. There is reason to expect that such independent analyses would improve our understanding of the strengths and weaknesses of machines and remedy some of the shortcomings of the federal voting system certification process. This would provide voters and concerned citizens with information to help them assess the equipment they vote on. It would also help local and state election officials to make better procurement and certification decisions.

  The value of independent evaluation is probably most pronounced when it comes to security. Security flaws can sometimes be subtle and easy to miss, even for experts. For this reason, enabling more people, especially security experts, to review the software significantly increases the likelihood that security problems in the code will be found.

- *Accountability:* The testing labs have been criticized for doing a poor job of evaluating voting systems. There have been a series of documented failures of the testing labs to discover

serious security and reliability problems in the voting equipment they approved. In my own examinations of voting system source code at the request of state election officials, I found serious defects in the source code that should have been immediately apparent to anyone with expertise in security. One cannot help but wonder whether the testing labs have anyone qualified in security reviewing the source code.

These failures may be due to structural problems in the way that testing is performed. Because testing labs are paid and selected by the vendor who makes the equipment being tested, testing labs are surely aware that withholding approval too frequently might send vendors to competing testing labs with a reputation for more lenient treatment. Elsewhere in the software industry, a similar "race to the bottom" has been observed in labs that test compliance to international computer security standards[8]. Unfortunately, at present there are few checks and balances that can be used to hold testing labs accountable if they fail to serve the public interest. In the long run, source code disclosure might help to ensure that the process is effective by holding testing labs accountable in the court of public opinion if they approve systems with obvious defects in the source code.

- *Improving voting machines:* In the long term, source code disclosure could have the effect of improving the quality of voting system software. First, source code disclosure allows a large community to spot bugs and problems so they can be corrected before they cause problems in the field. Because it is often hard for people to spot problems in their own work, a fresh eyes can see things that people who are most familiar with the code can miss by providing a fresh perspective. Second, source code disclosure would give vendors a powerful incentive to make sure their code is of high quality, to avoid public embarassment.

- *Promoting competition:* Source code disclosure would eliminate one barrier to interoperability between equipment from different vendors, potentially enhancing competition between vendors and providing more options to local election officials. Today, election officials cannot mix and match equipment from multiple vendors within the same jurisdiction. The business model adopted by the major vendors is based upon locking in counties as a captive customer of a single vendor. If the county wants to upgrade or enhance their system, any components they buy must come from that vendor. Unfortunately, this reduces the choices available to local election officials, reduces competition, and makes it harder for new companies with innovative products to enter the voting system market. Vendors use the proprietary nature of their code as one tool to keep counties captive. Source code disclosure would allow new vendors to enter the markets and build equipment that interoperates with the major vendors' equipment. This could potentially break the sole-source relationship vendors currently have with the counties and provide more alternatives to local election officials. However, achieving the benefits of interoperability would likely require changes to how we certify voting systems to permit certification of mixed-vendor systems.

  Source code disclosure could also allow new companies to provide maintenance and support services for equipment built by the major vendors. This, too, would promote competition and provide election officials with more choices. In today's personal computer (PC) market, one vendor (e.g., Dell) provides the hardware and another (e.g., Microsoft) provides the software. This model has increased competition between vendors, lowering prices for PC users. It is possible that opening the voting market to new vendors could reduce prices for voting systems in the same way that it has for PCs.

Arguments against source code disclosure:

- *Disclosure isn't sufficient:* Source code disclosure alone cannot ensure that voting machines are trustworthy, because of the limitations of source code analysis mentioned earlier. For instance, analysis of disclosed source code cannot ensure that the equipment is free of security vulnerabilities or malicious logic designed to rig an election[9], and it cannot ensure that the voting machines will be fair and accurate.

  At present, the best tool we have for ensuring that votes are counted accurately is to use voter-verified paper records and perform routine manual audits of the paper records[10] [11]. Adoption of voter-verified paper records and routine audits would reduce our reliance on source code analysis to ferret out security and reliability problems in the software.

  The TGDC, a body which helps to set federal voting system standards, has recently endorsed a requirement that voting systems be *software-independent*[12]. A voting system is considered software-independent if an undetected change or error in the voting software cannot cause undetectable changes or errors in the outcome of the election[13]. For instance, voting systems with a voter-verified paper record are considered software-independent, because the voter-verified paper records can be used to audit or recount the election results. Software-independence reduces some of the urgency for source code disclosure, by reducing (but not eliminating) the impact that defects in the source code can have.

  In general, we can rate voting systems by the degree to which they rely on software:

  - Paperless e-voting systems are completely dependent on the correctness of their software.
  - Adding a VVPAT printer reduces the dependence on software.
  - Paper-based optical scan systems reduce this dependence even further, and hand-counted paper ballots eliminate dependence on software.

  Generally, the more the system depends on the correctness of its software, the greater the likelihood of reliability and security problems. Of course, software independence is just one among several considerations in the choice of a voting system.

- *Transition risks:* If source code disclosure is mandated with insufficient advance notice and the transition isn't managed properly, there is a risk that in the short term disclosure could create more problems than it solves. Based on my experience[14] [15] reviewing the source code of some voting software, it is my prediction that immediate disclosure of source code would likely lead to discovery of serious problems in all vendors' machines.

  It is not clear that vendors could respond and fix these problems within a single election cycle. Even if they could, the process of repairing all of these problems and approving and deploying the patches could place a heavy burden on existing certification processes and on election officials. In the election world, the time between identification of a flaw and the availability of a patch for it is often painfully long. For instance, it has been over a year since two serious security vulnerabilities were identified in one voting system by Finnish researcher Harri Hursti[16] [17], but still no solution is available to election officials, despite the fact that one of these vulnerabilities was labelled by some security experts as the worst vulnerability they have ever seen in a voting system[18]. As another example, one system contains a security vulnerability that was reported privately to the vendor in 1997[19], disclosed publicly in 2003[20], confirmed to be still present in a 2004 report[21], was still present when I examined the system in 2006[22], and remains unresolved to this day[23]. Looking to the future, it is possible that immediate source code disclosure might lead to the discovery that every e-voting system in

widespread use has multiple problems that cannot be addressed through procedures and that cannot be repaired in time for the election. Depending upon the timing, all machines in the country could have to be re-designed, re-implemented, and re-certified in a single election cycle. In practical terms, this would be a disaster.

These risks can probably be mitigated if appropriate plans are put in place to manage the transition to source code disclosure smoothly and if disclosure requirements are phased in over time.

- *Giving aid to attackers:* One serious concern is that disclosing voting system source code might aid attackers to find and exploit vulnerabilities in voting systems. This is indeed a valid concern. Throughout the history of computer security, experts have struggled with this risk.

  At the same time, this concern must be tempered with a recognition that this is a complex issue. If the voting system contains vulnerabilities, lack of source code will only slow down, but not stop, a dedicated attacker. For that reason, security experts usually recommend that it is far safer to avoid vulnerabilities in the first place, and source code disclosure is one effective way to advance that interest.

  In computer security, it is widely accepted that well-designed systems should be constructed so that disclosing the source code does not endanger security. Kerkhoff's principle, which dates back to the 19th century, states that systems should be designed so that their security does not rely upon the secrecy of their design or implementation[24]. The reason is simple: if the leak of information about how the system works can compromise its security, then the system is fragile[25]. Practical experience shows that these secrets often leak—for instance, one vendor's source code was leaked onto the Internet in 2003—and even in the absence of leaks, a sufficiently dedicated adversary can get access to the same information through reverse engineering. Generally speaking, if the system can be hacked by an adversary with access to the source code, it can also be hacked by an adversary without that kind of access, so the presence of such a vulnerability is very troubling. For these reasons, the consensus in the computer security community is that systems should be designed to ensure that revealing the source code does not endanger system security.

  If we had confidence that existing voting systems were well-designed, we could disclose their source code without fear of helping attackers. Unfortunately, the concern is that existing systems are so poorly designed that source code disclosure could in the short run help attackers. In the long run, my experience is that disclosure helps to raise awareness of the problems among the users of the software, and thereby drives better security practices and forces systems to be better designed. However, this takes time. Therefore, my expectation is that in the long run source code disclosure would improve voting system security more than it hurts, but the transition must be managed carefully.

  One must be careful to avoid drawing the wrong conclusion. Some vendors and election officials have suggested that the secret, proprietary nature of voting system code is a key security measure, because giving people the source code would give them directions on how to hack it. Such statements reflect a disturbing lack of familiarity with computer security. I am not aware of any computer security expert who suggests that we should rely upon the secrecy of the source code as a key part of our strategy for securing our elections[26]; this would violate basic principles of secure design[27].

Open source vs. disclosed source. Some advocates have argued that election-related software should be developed through "open source" processes, where any interested party can contribute code to the elections software. "Open source" is a term of art in the computing industry. Open source software is software that is released under relaxed licensing terms. Recipients typically receive the right to modify the software for their own purposes and to re-distribute their modifications freely. This allows users to collaborate to improve the software on their own, without relying upon the original developer of the software. Open source software is often provided to users at no cost, and the software almost always comes with source code. Open source software is often, but not always, written by interested volunteers through a non-corporate, community-driven development process.

It is important to note that open source software is not the same as disclosed source software. Vendors can continue to use traditional software development processes and subsequently disclose the resulting source code, without any need to adopt any of the other distinguishing features of open source software. Source code disclosure policies, licensing terms, and software development processes are three separate matters, and while open source software takes a particular stance on all three topics, it is source code disclosure that matters most to elections.

While "open source" development processes do have advantages, I believe that mandating "open source" development would be inappropriate at this time. In comparison, source code disclosure is a much less radical step. In this model, vendors would continue to write and develop software themselves and would control the contents of the software, but they would be required to disclose the source code to certain parties.

The impact of disclosed source on intellectual property. Source code disclosure would not prevent vendors from competing on the merits of their source code and protecting their legitimate innovations. Source code disclosure would implicate several forms of intellectual property protection, but I wish to focus on issues involving copyright and trade secret protection. My understanding is that source code may be protected simultaneously under copyright law and trade secret law.

Before addressing these issues, however, I'll address an initial question that the previous sections of my testimony might provoke: If source code disclosure, or publication of source code under an open source license, offers long-term advantages to voting system vendors as well as the election system as a whole, why haven't vendors already moved in that direction on their own? The answer, I think, is that if one vendor discloses their source code and another does not, the disclosed-source vendor has no way of knowing whether their rights are being violated by the closed-source vendor. Therefore, the marketplace discourages vendors from going to a disclosed source model on a voluntary basis.

Vendors would retain copyright protection in their source code, even if the code were openly published. This is not unlike publishing a book. When an author publishes a book, it is protected under copyright law, and the author can assert the rights granted by copyright law to prevent others from making unauthorized copies. This allows the author to sell copies while providing recourse against people who would make wholesale copies of the book without permission. Just as importantly, recipients can read the book and quote excerpts for criticism or other kinds of fair use. In a similar vein, recipients of voting system source code under could read and analyze the code, but copyright law would prohibit them from making wholesale copies of the source code. As a result, vendors' interests in preventing competitors from free-riding on their efforts would be protected.

Some source code disclosure models might well threaten a vendor's current ability to require counties to use equipment from that vendor, and that vendor only; but the increased competition, innovation, and flexibility would serve important public interests in the election system.

Source code disclosure does raise difficult questions about trade secret protection. Unrestricted

disclosure would likely destroy any trade secret protection in the source code, but some of the more controlled forms of disclosure (as I discuss later) would preserve the possibility of protecting trade secrets. Whether trade secret protection is appropriate for source code in certified voting systems is a question that I'm not qualified to answer on my own, but it is one that I hope this Committee will examine very carefully. Specifically, the public interest in transparency and accountability warrant close attention[28].

Source code disclosure also eliminates the vendors' "information advantage" over their customers and the public. At present, vendors can make claims about their software (e.g., that it is perfectly secure) without being contradicted. Source code disclosure would force vendors to be more circumspect about their claims—which may reduce the vendors' flexibility, but seems to be in the public interest.

Requiring source code disclosure of all vendors sets a level playing field. To the extent that source code disclosure has costs for vendors, vendors can set their prices to reflect the costs of disclosure.

Policy options

There is a broad spectrum of possible policy options that are available to address these issues.

Do nothing (status quo). One possibility is to make no changes to the status quo regarding source code disclosure and continue to permit vendors to treat their source code as secret and proprietary. The risk of doing nothing is that the lack of transparency may contribute to further loss of confidence in e-voting[29].

Mandate disclosure to the public. Another possibility is to require vendors to disclose the full source code for all the software in their voting equipment to any interested member of the public. This could be accomplished, for instance, by requiring vendors to disclose source code to the EAC as a condition of certification and requiring the EAC to publish it or provide it to members of the public upon request. There should be no possibility for vendors to protest; disclosure would be mandatory.

Intermediate steps. There are many small steps one could take that would incrementally move us towards increased disclosure without going all the way to full public disclosure all at once.

- *Mandate disclosure to the federal and state election officials.* The smallest step would be to require vendors to disclose source code to federal and state election officials. This would permit election officials, at their discretion, to commission independent technical experts to analyze the source code. One shortcoming of this approach is that election officials generally do not have the necessary technical expertise in-house; hiring paid consultants to perform the work is expensive; and some election officials might be reluctant to seek analyses that might reveal embarassing flaws in systems that they have approved and that are in widespread use.

  This step would likely have little effect on the status quo. The EAC already has the authority to demand that vendors disclose the source code to them as a condition of submission for certification, but has declined to exercise that authority[30].

- *Mandate disclosure to candidates.* The next step would be to require source code to be disclosed to all candidates and their representatives, such as any technical experts that they designate. Vendors would not be permitted to protest or prevent such disclosure. To prevent further re-distribution of the source code, the candidates' designated experts could be required to sign agreements not to further disclose the source code to third parties. However, it

is critical that these non-disclosure agreements be written to allow the experts to publicly discuss their findings and provide evidence to support their conclusions. The agreements must also be written to preserve the independence of the candidates' experts; vendors and officials must not be allowed to interfere with, limit, or pressure the candidates' experts. Non-disclosure agreements must not be used as a way to silence dissent or place barriers to meaningful review of the source code.

- *Mandate disclosure to local election officials.* Another option would be to require that source code be disclosed to local election officials and their designees. This would permit county officials, at their discretion, to commission independent technical analysis of the source code. This might help them to choose among multiple systems when buying new equipment, or to understand the strengths and weaknesses of their systems and craft appropriate procedural mitigations.

- *Mandate disclosure to qualified experts.* A final option would be to require that source code be disclosed to any qualified expert upon request, regardless of the expert's affiliation. Those experts might be required to sign non-disclosure agreements, as discussed earlier; access might be restricted to US citizens; and to avoid a conflict of interest, vendors might be forbidden from gaining access to their competitors' code. However, to ensure that such a requirement meets its goals, the definition of qualified expert must be crafted carefully to ensure that qualified people are not wrongly excluded. This is not a theoretical concern: one early attempt to draft such requirements[31] was flawed[32] [33] [34]. For instance, it might be reasonable to require either a graduate degree or at least five years of experience in computing.

Mandating disclosure to qualified experts would help improve voting machines, improve the evaluation process, hold vendors and testing labs accountable for their performance, and lead to more informed debate about voting systems. It would address concerns about public disclosure aiding attackers and help manage the transition.

Ultimately, though, this position is problematic in the long run, because it puts a small cadre of experts in a privileged position. This will be a constant source of dissatisfaction and friction for those who distrust whichever experts are permitted to study the code. While this does enhance security review, restricting disclosure to qualified experts fails to address the public interest in the transparency of voting software.

Phased introduction of disclosure requirements. One way to address the transition risks would be to gradually introduce these requirements over time. For instance, one possible timetable for increased source code disclosure might be as follows:

- One might require vendors to disclose source code to state and federal election officials immediately, and require election officials to promptly commission independent expert security analyses of the systems. Officials could require vendors to fix any security problems found in the code, to make the code safe for broader disclosure.

- Then, one might require source code disclosure to qualified experts, at their request, after enough time has passed to correct any problems found in the prior phase. Two years should suffice.

- Finally, one might require source code disclosure to the public at some future date specified in advance. Five years notice ought to be enough for vendors to prepare their code for public disclosure and to ensure that it can withstand scrutiny, so that we can be confident public disclosure will not assist attackers to attack elections.

It is important that the timetable be set and published now, so that vendors have enough time to ready their systems for public disclosure. Competitive pressures make it difficult for vendors to begin preparations without a concrete deadline.

If vendors are given sufficient advance notice, there is no reason they cannot ensure that their systems will be safe to disclose. A gradual introduction of source code disclosure requirements could minimize the transition risks while advancing the long-term goals of transparency and security.

Reducing dependence on software. Another policy option would be to reduce the severity of the source code secrecy problem by reducing our dependence upon software in elections. As discussed earlier, this could be achieved by mandating voter-verified paper records and routine audits. Adoption of paper ballots (whether optically scanned or manually counted) would further reduce the degree of dependence upon secret software and further reduce the need for source code disclosure. This direction would not address the public interest in transparency, but it would reduce or mitigate many of the other problems with secret code.

COTS software

The COTS challenge. COTS code poses a special challenge for mandatory disclosure of voting system source code. Many deployed voting systems contain COTS software written by third-party vendors, and the equipment manufacturer may not have access to the source code for that software or may not have permission to disclose it. Thus, any requirement to disclose the source code for all software in deployed systems could put some vendors in a serious quandary: they would either have to negotiate with the third-party software vendor for the rights to disclose that code; replace the undisclosable third-party software with code that they are free to disclose and seek certification for the new code; or withdraw their equipment from the market. Forcing the decertification of voting equipment that counties have already paid for would make life very difficult for local election officials who have an election to run. The impact of this is likely to vary from vendor to vendor, because some vendors rely more heavily on COTS code than others. While some vendors might not face such a quandary, forcing even one major vendor to recall their equipment on short notice would cause havoc for jurisdictions who use that vendor's equipment.

New systems would be unlikely to face this problem. There is no reason that voting equipment needs to contain undisclosable source code. Any competent engineer should be able to design voting equipment without resorting to third-party COTS code that cannot be disclosed, if source code disclosure is specified as a requirement at design time. Therefore, for new equipment, I do not see any barrier to full source code disclosure.

However, disclosing the source code of systems that were not designed to be publicly disclosed poses significant challenges. The problem is that existing equipment was not designed with source code disclosure in mind, and consequently some voting systems contain third-party COTS code that may not be easy to disclose or replace. This complicates the task of setting policy regarding source code disclosure.

Policy options. This problem with COTS code in legacy voting systems could be addressed in one of several ways.

- If vendors are given sufficient advance notice of the disclosure requirement, they should be able to ensure that their code is free of undisclosable COTS code. However, this "sunset" period for use of COTS code would delay imposition of the full disclosure requirement by several years.

- Another option is to exempt third-party COTS code from the source code disclosure requirement. Vendors would only be required to disclose source code for software they wrote themselves or that they otherwise have permission to disclose.

  However, this option is problematic, because COTS code can still cause problems. From an engineering point of view, COTS code is no safer than vendor-written code. COTS code can contain bugs and defects; it can contain malicious logic deliberately introduced to rig an election; and it can be manipulated or tampered with, just like vendor-written code. Therefore, any exemption for COTS code should probably be time-limited.

- Perhaps the least intrusive option is to introduce source code disclosure in a phased fashion. In the first phase, voting system vendors would be required to disclose as much source code as possible, including (at a minimum) all of the source code that they have written themselves. During the first phase, vendors would qualify for a limited-time exemption for COTS code, if they do not have the right to re-distribute its source code. In the second phase, after enough time has passed to allow vendors to replace all undisclosable COTS code or otherwise re-design their machines to ensure compliance, the COTS exemption would be eliminated and vendors would be required to disclose all election-related source code. To ensure the success of such a phased plan, it would be important to set a clear timetable in advance so that vendors can plan accordingly.

Notes

[1]B. Simons, "Electronic voting systems: the good, the bad, and the stupid", ACM Queue 2(7), Oct. 2004.

[2]D.W. Jones, "Misassessment of Security in Computer-Based Election Systems", Cryptobytes 7(2), Fall 2004, pp.9-13.

[3]P. Smith, "States with Escrow Provisions," March 12, 2007. `https://www.verifiedvotingfoundation.org/article.php?id=6439`

[4]D.W. Jones, "Voting System Transparency and Security: The need for standard models", written testimony before the EAC Technical Guidelines Development Committee, Sept. 20, 2004. `http://www.cs.uiowa.edu/~jones/voting/nist2004.shtml`

[5]J. Hall, "Transparency and Access to Source Code in E-Voting," USENIX/ACCURATE Electronic Voting Technology (EVT'06) Workshop. `http://josephhall.org/papers/jhall_evt06.pdf`

[6]D.K. Mulligan, J.L. Hall, written testimony before the California Senate Elections, Reapportionment & Constitutional Amendments Committee, Feb. 8, 2006. `http://josephhall.org/nqb2/media/Mulligan_Hall_OSHRG_Statement.pdf`

[7]P.G. Neumann, Written testimony before the California Senate Elections Committee, Feb. 8, 2006. `http://www.csl.sri.com/neumann/calsen06.pdf`

[8]R.J. Anderson, *Security Engineering – A Guide to Building Dependable Distributed Systems*, Wiley, 2001, §23.3.

[9]J. Bannet, D.W. Price, A. Rudys, J. Singer, D.S. Wallach, "Hack-a-Vote: Demonstrating Security Issues with Electronic Voting Systems", IEEE Security & Privacy Magazine 2(1), Jan./Feb. 2004, pp.32-37.

[10]D.W. Jones, "Auditing Elections", Communications of the ACM 47(10), Oct. 2004, pp.46-50.

[11]A.D. Rubin, Written testimony before the Election Assistance Commission, June 30, 2005. `http://avirubin.com/vote/eac2.pdf`

[12]TGDC Resolution #06-06, "Software Independence of Voting Systems," Dec. 5, 2006.

[13]R.L. Rivest, J.P. Wack, "On the notion of 'software independence' in voting systems", `http://vote.nist.gov/SI-in-voting.pdf`.

[14]"Security Analysis of the Diebold AccuBasic Interpreter", Report of the California Secretary of State's Voting Systems Technology Assessment Advisory Board, Feb. 14, 2006.

[15]A. Yasinsac, D. Wagner, M. Bishop, T. Baker, B. de Medeiros, G. Tyson, M. Shamos, M. Burmester, "Software Review and Security Analysis of the ES&S iVotronic 8.0.1.2 Voting Machine Firmware," Feb. 23, 2007. `http://election.dos.state.fl.us/pdf/FinalAudRepSAIT.pdf`

[16]H. Hursti, Black Box Voting, "Critical Security Issues with Diebold Optical Scan", July 4, 2005.

[17]H. Hursti, Black Box Voting, "Critical Security Issues with Diebold TSx", May 11, 2006.

[18]A. Rubin, E. Felten, "Report Claims Very Serious Diebold Voting Machine Flaws," May 11, 2006. http://www.freedom-to-tinker.com/?p=1014

[19]D.W. Jones, "The Case of the Diebold FTP Site," Oct. 2, 2003. http://www.cs.uiowa.edu/~jones/voting/dieboldftp.html

[20]T. Kohno, A. Stubblefield, A.D. Rubin, D.S. Wallach, "Analysis of an Electronic Voting System", July 24, 2003.

[21]RABA Innovative Solution Cell, "Trusted Agent Report: Diebold AccuVote-TS System", Jan. 20, 2004.

[22]"Security Analysis of the Diebold AccuBasic Interpreter", Report of the California Secretary of State's Voting Systems Technology Assessment Advisory Board, Feb. 14, 2006.

[23]D.W. Jones, "Connecting Work on Threat Analysis to the Real World", June 8, 2006.

[24]A. Kerckhoffs, "La cryptographie militaire", *Journal des sciences militaires*, vol. IX, Jan.-Feb. 1883.

[25]B. Schneier, "Secrecy, Security, and Obscurity," May 15, 2002.

[26]B. Schneier, "Voting Software and Secrecy," Oct. 2, 2006.

[27]J.H. Saltzer, M.D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE* vol 63 no 9, Sept. 1975.

[28]D.S. Levine, "Secrecy and Unaccountability: Trade Secrets in Our Public Infrastructure," *59 Florida Law Review 135.* http://ssrn.com/abstract=900929

[29]D.L. Dill, B. Schneier, B. Simons, "Viewpoint: Voting and technology: who gets to count your vote?", Communications of the ACM, 46(8), Aug. 2003.

[30]A. Burstein, J.L. Hall, "Unlike Ballots, EAC Shouldn't Be Secretive," *Roll Call*, Jan. 22, 2007. http://josephhall.org/papers/Burstein_Hall-Roll_Call_2007-01-26.pdf.

[31]Notice of Rule Development, "Rule 1S-2.004: Procurement, Use and Assessment of Voting Systems," Florida Administrative Weekly, May 26, 2006.

[32]D. Dill, "Comments proposed rules 1S-2.004 and 1S-2.015," June 12, 2006. http://election.dos.state.fl.us/laws/proposedrules/pdf/partC_pubCom_r1s_2_004_2_015.pdf

[33]A. Yasinsac, "Comments on Rule #1S-2.004. Procurement, Use and Assessment of Voting Systems," June 10, 2006. http://election.dos.state.fl.us/laws/proposedrules/pdf/partA_pubCom_r1s_2_004_2_015.pdf

[34]R. Benham, "Proposed Rules 1S-2.004 and 1S-2.015," June 12, 2006. http://election.dos.state.fl.us/laws/proposedrules/pdf/partA_pubCom_r1s_2_004_2_015.pdf