LAWFARE

CYBERSECURITY

Open-Source Software Won't Ensure Election Security

By **Matt Bishop** Thursday, August 24, 2017, 2:00 PM

DayZero: Cybersecurity Law and Policy

The technology behind elections is hard to get right. Elections require security. They also require transparency: anyone should be able to observe enough of the election process, from distribution of ballots, to the counting and canvassing of votes, to verify that the reported winners really won. But if people vote on computers or votes are tallied by computers, key steps of the election are not transparent and additional measures are needed to confirm the results.

In a New York Times op-ed a couple weeks ago, James Woolsey and Brian Fox proposed using "open-source systems that can guard our votes against manipulation." Their hypothesis is that "open-source software is less vulnerable to hacking" than proprietary voting software because "anyone can see how open-source systems operate. Bugs can be spotted and remedied, deterring those who would attempt attacks. This makes them much more secure than closed-source models." This sounds reasonable, but in fact, open-source systems are only one step towards guarding our votes against manipulation—and the hypothesis that using open source software will by itself improve security is questionable at best.

First, with the systems in use today, there is no guarantee that the software running on any machine is in fact the software it is supposed to be running, open source or not. And even if we could know with certainty that the installed software matches the software source, the quality of the software is critical. Poorly written software, whether open source or not, creates vulnerabilities, and is thus vulnerable to hacking. Open source software allows anyone to detect vulnerabilities. We do not believe in "security through obscurity"—that is, relying on secrecy as a primary security strategy—but making source code available to everyone for inspection makes it available to the attackers for inspection. And the attackers are often highly motivated to find vulnerabilities.

Complicating this is the relative ease of identifying one vulnerability and the difficulty of finding them all. Attackers need to find just a single flaw in order to exploit a system. On the other hand, it is very easy for reviewers to miss something—the Heartbleed bug that affected millions of websites and devices in 2014 occurred in open source software—or to make assumptions about the environment in which the source code is executed that turn out to be wrong. Software authors, maintainers, election officials, and other defenders must find every flaw, fix them all, and then distribute the fixed system (or patches) to everyone using the system.

Patch distribution creates its own set of potential problems, as it informs attackers that there was a vulnerability (and where in the code it is), leaving anyone who does not immediately install the patch especially vulnerable. For example, many years ago, a response group announced a patch to a well-known, widely used piece of software. Within thirty minutes, that vulnerability was being exploited around the world. Many sites did not have the time or resources to install the fix. The patch was announced at 5 p.m. East Coast time on a Friday, making things worse.

Open source software is a good thing. Published source, a lower bar, is a useful if less optimal alternative. But visibility of the source is not enough. Security analysts need access to everything that is used to create the system, including operating system source, driver source, compiler source, hardware, and hardware fabrication information, and then directions on how to create the systems used in the voting (such as the voting system, the tally system, and any systems and software used to display the results). Even that isn't enough, as (for example) the fabrication facilities may not follow the directions the analysts are given. The supply chain matters, too—manufacturers or their employees may even be malicious! So the analysts need to monitor the actual system construction to verify everything. Even then, they must be aware that what is done today may not be what is done tomorrow, or what was done yesterday.

Finally, even perfect software does not guarantee trustworthy elections. Trustworthiness is also a product of the way the system, and software, are used. For example, consider a system that uses a difficult-to-guess password, but that password can be found on a website. No amount of scrutiny of the system will reveal this flaw.

So assuming that open source systems are more secure than other systems is a dangerous fallacy, just as assuming closed source systems are more secure than other systems. Properly evaluating security requires more than simply considering the openness of the source.

The question we should be asking is "how can we ensure that election results are accurate when we cannot trust the computers used to run elections?" rather than "how do we make electronic voting secure?" Nothing is ever absolutely secure. But we can often make computers, systems, and processes "secure enough" for their tasks, provided we have an independent way to check the results. One technique is to

produce a voter-verified paper trail, ensure that the paper trail is trustworthy, and manually audit the electronically tabulated results against the paper audit trail. Another technique called "end-to-end verifiability" allows individual voters to verify that their vote was recorded and counted correctly. Simply enabling everyone to examine the source is not sufficient, and could even give voters and election officials a belief that the system is secure when it is not.

We believe there are excellent reasons to move to open-source voting systems. For instance, there are good arguments that the public should own the voting system. Open-source systems allow vendor claims about software to be verified. Open-source systems running on commercial, off-the-shelf (COTS) hardware could be far cheaper to acquire and maintain than proprietary voting systems. Adopting open-source systems could promote a competitive market for technical support for local election officials, also decreasing costs. Open-source systems could be designed to facilitate auditing against the paper trail more efficiently than commercial systems permit. And using open-source systems would make it possible for jurisdictions to customize the software to their needs.

All this needs to be considered as we work to improve the security and transparency of our election systems. But adopting open-source systems would not by itself provide any assurance that computers used in voting are doing what they are supposed to do. Nor would it obviate the need to audit the results.

This piece reflects the contributions of the following individuals:

Matt Bishop, University of California, Davis
Philip Stark, University of California, Berkeley
Josh Benaloh, Microsoft Research
Joseph Kiniry, Free and Fair
Ron Rivest, MIT
Sean Peisert, University of California, Davis
Joseph Hall, Center for Democracy and Technology
Vanessa Teague, University of Melbourne (Australia)

Topics: Cybersecurity

Tags: Election security, election interference, open source

Matt Bishop is a Professor in the Dept. of Computer Science at the University of California, Davis, and co-director of the Computer Security Laboratory there. He received his Ph.D. from Purdue University in 1984. His main research area is the analysis of vulnerabilities in computer systems. He has been active in election work, and electronic voting system work, since 2003. He was on the RABA team that performed a red team test of Maryland's electronic voting systems in 2004, on the Florida team that examined electronic voting systems for problems after the 2006 Congressional District 13 election, and co-led the technical part of the California Top-to-Bottom Review of electronic voting systems used in the state of California. His textbook, "Computer Security: Art and Science," is widely used in university courses on computer security.