Title
Security and Elections

Journal
IEEE Security & Privacy, 10(5)

ISSN
1540-7993

Author
Bishop, Matt

Publication Date
2012-09-01

Peer reviewed

University of California, Davis educators teach numerous computer security classes for undergraduate majors and nonmajors and for graduate students. These classes have used elections, and electronic-voting systems, both as lecture material and in class projects. This column presents some of their experiences and what they've learned.

Keywords: elections, election security, computer security, privacy, voting fraud, electronic voting, e-voting, computer security education

Security and Elections

Matt Bishop and Sean Peisert
University of California, Davis

Elections are common to almost all societies. Periodically, groups of people determine their representatives, leaders, neighborhood spokespersons, corporate executives, or union representatives by casting ballots and counting votes using a variety of schemes. Those who don't participate see others around them doing so. And stories abound about rigged elections or results considered compromised by accident or poor communication.

US-based elections follow a general pattern of voter registration, determining items to vote on, generating ballots, distributing election materials to the polling places, voting, counting the votes, declaring winners, and auditing the results. The details differ among jurisdictions, but each step requires considerable care to ensure the election's integrity. So, elections are an ideal mechanism for teaching about security.

At the University of California, Davis, we teach numerous computer security classes for undergraduate majors and nonmajors and for graduate students. This column presents some of our experiences using elections and e-voting systems as lecture material and as a class project done with the Yolo County Elections Office.

Learning Objectives

We focused on five specific learning objectives, taken from those that Matt Bishop and Deborah Frincke described:[1]

- "D. An ability to function on multidisciplinary teams
- E. An ability to identify, formulate, and solve engineering problems
- F. An understanding of professional and ethical responsibility
- G. An ability to communicate effectively
- H. The broad education necessary to understand the impact of engineering solutions in a global, economic, environmental, and societal context"

These objectives come from the Accreditation Board for Engineering and Technology Behavioral Outcome Criteria. We wove these throughout lectures and projects.

Besides these broad learning objectives, we had two security-related objectives. The first was to teach students how to handle conflicting security requirements. The second was to teach them a penetration-testing methodology.

Lectures

E-voting systems are an excellent platform on which to demonstrate clashes of requirements.[1] Domain

experts determine the requirements' relative importance. The students' analysis and discussion of the requirements had to meet this balance. This satisfied objective E.

For example, US-based elections' goals include ballot privacy and anonymity, accuracy of the counting, and the results' credibility. The last goal means that the counting must not only be accurate but also be accepted as accurate, so some form of verification is necessary. Typically, the requirement for auditing (the vehicle for verification) requires tying data (votes) to the subjects supplying it (voters). This conflicts with the requirement that no voter can be associated with a cast ballot, even if the voter wants to prove such an association (because then the voter could sell his or her vote). Rarely do two requirements (election auditability and ballot secrecy) conflict so directly.[2] Requirements analysis showed that two key security elements—confidentiality and integrity—can contradict each other. This caused students to examine auditing systems that collect as much data as possible and to ask whether doing so might actually compromise security.

Also consider availability, which ensures that those authorized by law to vote can do so. Balancing this against the need to prevent those not authorized from voting leads to mechanisms that might enable illegitimate voters to vote or deny legitimate voters their vote. For example, requiring government-issued identification might reduce the first problem. However, many legitimate voters might not have the requisite identification, increasing the lack of availability to authorized voters. Again, conflict.

E-voting systems are part of an election process, so the process dictates their use. When evaluating the system's security, the evaluators must consider the environment—the process—in which the system is used. For example, most mission-critical systems assume the people who set them up and use them are trained to use them. But in an election, you can't make this assumption. Even county IT workers might not be well equipped to participate in audits when something goes wrong.[3]

This process-oriented view showed students that security depends as much on assumptions about the environment and people as it does on technology.[4] It also introduced them to limitations on those who provide and use engineering solutions, giving them an often unexpected context in which they had to evaluate those solutions. This met objective H.

Furthermore, US e-voting systems must meet certain standards, which led to a discussion of those standards. The current federal standards' flaws[5] lend grist to the assertion that not all standards are good and that you need a basis for asserting requirements of systems that are to be trusted. The concept of "building security in" invariably arose in class. So, we discussed the difficulties of adding security onto a system after it's built—especially when it wasn't designed with security in mind or when the standards and requirements change after a system has been designed and implemented.

"Logic and accuracy" tests, conducted by election officials and made public, aim to validate that the systems are configured and initialized properly before voters use them to cast votes. Explaining to nontechnical people what such tests show—and, more important, don't show—helps them understand whether the tests do what's desired. Students—especially advanced computer science majors—can have difficulty grasping the need to do this. But such a grasp is critical to understanding why so many security problems arise. And knowing how to speak to non-computer experts in a language and through a means they'll understand is a key skill for all computer security professionals (objective G).

The lectures concluded with a discussion of validation after the election. This step, called the *canvass*, typically involves manual recounting of a certain number of ballots. Election officials compare the counts to the electronic tallies; if a discrepancy exists, a resolution process is necessary.

Elections' complexity often surprised students. But because they understood elections' importance to civic life, they became enthusiastic about exploring security in general and elections in particular. We harnessed their enthusiasm with a project that let them delve into the problem further.

The Project

As part of teaching penetration testing, we taught the *flaw hypothesis methodology*:[6]

1. Gather information about the system and environment.
2. Using this information, hypothesize flaws.
3. Test the hypothesized flaws to see whether they're actual flaws.
4. Generalize the flaws found.

The project aimed to teach students how to apply this methodology, which is the basis for many other methods.

Of course, we wanted to pick a system that the students would find interesting. Yolo County, California, makes both paper ballots and e-voting systems available; voters can choose whichever they prefer. The Clerk-Recorder, Yolo County's chief election official, asked us to look at the voting system and identify any potential problems that she should be aware of.[7] The system consisted of a Hart InterCivic DAU (Disabled Access Unit) eSlate and the controlling JBC (Judge's Booth Controller), a small unit that poll workers use to enable a voter to vote on an eSlate.

The JBC connects to the eSlate and generates a four-digit access code for the voter. The voter enters the code on the eSlate, which confirms with the JBC that the code is active. The voter can then vote, and the cast ballot is stored on both the eSlate and JBC. The eSlate also prints a copy of the ballot so that a sighted voter can verify that the system recorded his or her selections correctly. When the polls close, the memory cards are removed from the JBCs, and the votes are counted (in practice, the printed copies are used).

We organized the class into teams of three or four because the ability to function in teams is critical to the students' careers. This dealt with objective D. We asked each team to examine the system independently, find anything that might pose a problem, and describe the environment in which someone could turn that problem into an attack vector. Initially, we didn't discuss details of the environment in which Yolo County would deploy the system; we also didn't ask the students to mitigate the problem. Because we had no access to the software source code or development environment, this was a black-box penetration test.

Information gathering began with a lecture on elections in Yolo County, as we described earlier. The students wrote a short report about an election's requirements and whether an e-voting system would affect meeting each requirement. The requirements that related to e-voting systems provided the basis for the next step. This emphasized objectives E and H.

The teams then read media reports about election threats, and read other studies, of e-voting systems.[8] None of these reports discussed the Hart InterCivic system being examined. The students learned how Yolo County elections are run, viewed demonstrations of how the system works, and had access to the loaned system. They also could ask Yolo County election officials about the election process in general and how the machines were used. In doing so, they learned to formulate questions for non-computer scientists and for fields that used computers as opposed to developing them (meeting objective G).

Using this information, the students had to suggest possible flaws and justify them by pointing to the specific requirement that the hypothesized flaw would violate. So, if the system froze for a few minutes but accurately kept track of the ballots, the accuracy requirement wouldn't be violated. The usability requirement might be, though. We told the students that asserting a possible flaw when the requirement was imprecise was fine (so, in the previous example, the freezing would be held to violate the usability requirement).

During hypothesis testing, students designed and carried out tests to validate or refute the proposed flaws. In some cases, the students couldn't carry out the tests owing to lack of equipment or time. We encouraged them to record how to test the hypotheses, to allow future teams to benefit from their experience and ideas.

Sometimes, students who found problems could generalize them to find other problems. Perhaps the best example came from the hypothesis that unplugging the JBC and associated eSlate (which draws power through the cable connecting it to the JBC) would drain battery power and shut down the system before the end of election day. Another team wondered whether some other mechanism could cause a power drain. They looked at the eSlate's cables. One such cable, which attaches to another eSlate, allows eSlates to be daisy-chained so that one JBC can run multiple eSlates. The students found that plugging a null terminator onto the trailing, unused cable would inhibit the eSlate from drawing power from the JBC, forcing it to run on batteries.

We asked students who found flaws to suggest ways to fix them. Most problems required vendor fixes. However, teams suggested some procedures the county could adopt. For example, the team that found the second power-draining attack recommended removing the daisy-chaining cable. Because Yolo County only uses one eSlate per polling place, the Clerk-Recorder thought removing the cable was a good idea, and did so.

Throughout the project, and especially at the end, we discussed the ethics of this type of testing, and especially the ethics of handling the vulnerabilities found. The students developed their own ideas about how to handle these problems and, most important, how their actions could affect the people they work

with and society at large. This met objective F.

Evaluation

Each team wrote two reports (but merged them). The first dealt with election requirements, as we mentioned earlier. The second recounted their work using the flaw hypothesis methodology and described in detail their hypotheses and experiments, what worked, what didn't, and the hypotheses they were unable to test. They had to tie each hypothesis to a specific requirement, to show that the hypothesis, if true, caused a violation of the requirement. When we graded the combined paper, we evaluated not only the work but also the writing to ensure it met standard technical-writing norms (objective G). So, we could determine whether the students met all the learning and security-related objectives.

The more a topic relates to practical, real-world issues in a student's life, the more likely that student will be engaged by the material and learn from it. UC Davis students include US citizens participating (or about to participate) in their first election and foreign students who can compare this material to practices in their own countries. The discussions among the students were interesting and informative, and their excitement about the project's real-world nature shone through.

One student joined a group of graduate students in the next quarter to complete the analysis and help write a paper about what they found.[9] All the students learned the flaw hypothesis methodology and applied it to systems whose security was critical to elections' success. They also saw their work impact civic procedures, which heightened their enthusiasm for the project: people listened, and their coursework actually had a practical effect!

References

1. M. Bishop and D. Frincke, "Achieving Learning Objectives through E-voting Case Studies," *IEEE Security & Privacy*, vol. 5, no. 1, 2007, pp. 53–56.
2. S. Peisert, M. Bishop, and A. Yasinsac, "Vote Selling, Voter Anonymity, and Forensic Logging of Electronic Voting Machines," *Proc. 42nd Hawaii Int'l Conf. System Sciences (HICSS), Digital Forensics—Pedagogy and Foundational Research Activity Minitrack*, 2009; www.cs.ucdavis.edu/~peisert/research/2009-PBY-HICSS-Voting.pdf.
3. M. Bishop et al., "E-voting and Forensics: Prying Open the Black Box," *Proc. 2009 Electronic Voting Technology Workshop/Workshop on Trustworthy Computing*, Usenix Assoc., 2009; http://static.usenix.org/event/evtwote09/tech/full_papers/bishop.pdf.
4. B. Simidchieva et al., "Modeling and Analyzing Faults to Improve Election Process Robustness," *Proc. 2010 Usenix/Accurate Electronic Voting Technology Workshop*, Usenix Assoc., 2010; http://static.usenix.org/event/evtwote10/tech/full_papers/Simidchieva.pdf.
5. E. Barr, M. Bishop, and M. Gondree, "Fixing Federal E-voting Standards," *Comm. ACM*, vol. 50, no. 3, 2007, pp. 19–24.
6. R.R. Linde, "Operating System Penetration," *Proc. Am. Federation of Information Processing Societies 1975 Nat'l Computer Conf.*, ACM, 1975, pp. 361–368.
7. M. Bishop, "E-voting as a Teaching Tool," *Proc. World Conf. Information Security Education*, 2007, pp. 17–24; http://nob.cs.ucdavis.edu/bishop/papers/2007-wise5-1/evote.pdf.
8. T. Kohno et al., "Analysis of an Electronic Voting System," *Proc. 2004 IEEE Symp. Security and Privacy*, IEEE, 2004, pp. 27–40.
9. E. Proebstel et al., "An Analysis of the Hart InterCivic DAU eSlate," *Proc. 2007 Usenix/Accurate Electronic Voting Technology Workshop*, Usenix, 2007.

Matt Bishop *is a professor in the Department of Computer Science at the University of California, Davis.*

Contact him at bishop@ucdavis.edu.

Sean Peisert *is an assistant adjunct professor in the Department of Computer Science at the University of California, Davis and is a research scientist at Lawrence Berkeley National Laboratory. Contact him at peisert@cs.ucdavis.edu.*