Dreams of Security

by M. E. Kabay, PhD, CISSP-ISSMP
Associate Professor of Information Assurance
Program Director, BSc and MSc in Information Assurance
Division of Business & Management
Norwich University, Northfield VT

One of my colleagues and I enjoy having vigorous discussions which cause those listening to turn pale and back off in fear that we will come to blows.

Actually we're good friends and just enjoy a good intellectual tussle. Sometimes we'll switch sides in the middle of the argument for fun.

One of our latest battles practically cleared out the Faculty/Staff dining room in the mess hall at Norwich University last week. The topic was electronic voting systems, and my colleague blew up when I suggested that having electronic voting systems produce a paper ballot to be verified by the voter and then dropped into a secured ballot box in case there was a recall.

Vermont has a tiny population; we have about 600,000 people in the entire state. Because of this small population, people here have many ways of becoming involved in civic affairs. Our state house in Montpelier (the smallest state capital city in the US, with 8,000 people) is open to the public, as are committee meetings. State officials such as the governor often walk about in town where ordinary citizens can chat with them in a friendly and very personal way.

I was recently invited to address the Government Operations Committee as they discussed a pending bill which would require any wholly electronic voting mechanism to be equipped with a means of producing a paper ballot that could be inspected by the voter and which would then be stored safely for official recounts. Given the importance of safeguarding the vote in our nation, I thought it might interest readers to step outside the confines of network security for a moment to consider the security implications of wholly electronic voting.

Today, there are three different forms of voting in place in use in the USA (I won't discuss remote, Internet-based voting in this column): one can mark a piece of paper by hand and have it read by people; one can mark a piece of paper by hand or machine and have it read by an optical-mark reader which tallies the results automatically; or one can use a wholly-electronic system with an input device such as a touch-sensitive screen which stores the results in a database and produces automatic tallies.

Normally, paper ballots, whether read by people or tallied by machines, are stored in sealed containers and can be opened with a court order in cases of judicially-approved recounts when election results are challenged.

In Vermont, the Secretary of State's office allows optical-mark readers to be used for elections; only one such machine is required per voting location, most of which have at most a few thousand voters registered per location. However, most locations still use manual counting of ballots under the supervision of representatives of the various political parties involved in the

election.

In my testimony before the Government Operations Committee, I stressed the following points:

- Any system of vote counting that relies on completely proprietary (secret) programs is potentially vulnerable to abuse. The underlying computer programs controlling how marks on ballots are counted in Vermont are proprietary (they are owned by Diebold Corporation), but the technicians who prepare the configuration tables relating a position on a ballot to a particular name work for an independent consultancy in Massachusetts and their configuration tables are open for inspection.

- Every optical tabulator is tested to see if it reads ballots correctly before the election begins.

- Passing a law that allows the Secretary of State to order a random check on the accuracy of machine tallies in any voting district will help prevent systematic fraud. The tallies in a manual recount must match the machine tallies to within an acceptable error rate (to allow for the inherent difference between machine tallies and human counting methods: machine reject incorrectly-marked ballots whereas people can agree on the intention of the voter).

- Wholly-computer-based voting systems have far more vulnerabilities to tampering than optical-mark sensors. We know that even companies such as Microsoft have allowed Easter Eggs (unauthorized, undocumented code such as flight simulators) to escape quality assurance and be delivered to customers in software such as MS-Excel. We know that microprocessors have been tampered with to cheat clients and evade testing (e.g., gas pump meters in the Los Angeles district were designed to overcharge customers by 10% -- unless they noticed one- or five-gallon deliveries, which were the volumes typically used by inspectors when checking accuracy). We know that production code has been profoundly flawed for years without being caught (e.g., the Colorado lottery's not-very-random-number generator that produced only numbers from zero to eight but never any nines). We know that data stored in databases without careful attention to chained cryptographic checksums involving timestamps, sequence numbers and the previous record's checksum can be modified to misrepresent election results.

- For all these reasons, we should resist the use of wholly-computerized voting machines until there is software that is entirely open to inspection.

- Any wholly-electronic voting machine should be required to produce a paper ballot showing the voter's choices for inspection by that voter (only). The voter should then be required to place the ballot in a ballot box for use in judicial recounts and random testing of the accuracy of the computerized voting system.

What fascinated me is his attitude toward the trustworthiness of electronic systems: "That's ridiculous," he said. "Surely you should be able to devise a foolproof electronic system impervious to tampering?? Otherwise we're all in deep trouble, because we've been replacing manual systems by electronic systems for years now in all aspects of business. Why should we

go to the expense of keeping old manual systems such as ballot boxes and hand recounts – which are vulnerable to abuses anyway – when we can – or ought to be able to – implement completely secure electronic systems?"

This charming confidence in the power of software engineering is undermined by several well-established principles of the field:

- Security is an emergent property (much like usability or performance) and cannot be localized to specific lines of code.

- Testing for security is one of the most difficult kinds of quality assurance procedures known; it is inherently difficult because failures can occur from such a wide range of sources.

- Security failures can come from design errors (e.g., failing to include identification and authentication measures to restrict access to confidential or critical data); programming errors (e.g., failing to implement a security measure because the source code uses the wrong comparison operator in a comparison); run-time errors resulting from poor programming practice (e.g., failing to prevent bounds violations that result in buffer overflows and the consequent execution of data as instructions); and malicious misprogramming (e.g., Trojan horses, logic bombs, and back doors).

- Worse, quality assurance is often sloppy, with poorly-trained people who don't want to be doing the work assigned to the job in spite of their protests. These folks often believe that manual testing (punching data in via a keyboard) is an acceptable method for challenging software (it isn't); they focus on showing that the software works (instead of trying to show that it doesn't); they don't know how to identify the input domains and boundaries for data (and thus fail to test below, at and above boundaries as well as in the middle of input ranges); and they have no systematic plan for ensuring that all possible paths through the code are exercised (thus allowing many ways of using the program to be wholly untested).

- The principles of provably-correct program design have not yet been applied successfully to most of the complex programming systems in the real world. Perhaps some day we will see methods for defining production code as provably secure, but we haven't gotten there yet.

How ironic that (someone incorrectly perceived as) a computer-science geek should thus be in the position of arguing for the involvement of human intelligence in maintaining security. I firmly believe that having independent measures to enforce security is a foundation principle in preventing abuse. Involving skeptical and intelligent people to keep an eye on voting machines is just one example of that principle, and it's worth the money to prevent our democracy from being hijacked.

For more along these lines, see Bruce Schneier's crisp and clear exposition of fundamental security principles in his brilliant book, *Beyond Fear: Thinking Sensibly About Security in an Uncertain World.* The famed cryptographer and general security boffin proposes a systematic

framework for analyzing security issues and proposed solutions:

Step 1: What assets are you trying to protect?
Step 2: What are the risks to those assets?
Step 3: How well does the security solution mitigate those risks?
Step 4: What other risks does the security solution cause?
Step 5: What trade-offs does the security solution require?

This framework applies not just to computer security or information security; it applies to all security decisions.*

* Schneier, B. (2003). *Beyond Fear: Thinking Sensibly About Security in an Uncertain World.* Copernicus Books. ISBN 0-387-02620-7. P. 231 ff. AMAZON link: < http://tinyurl.com/kqgft >

