By Jason Kitcat

Source Availability and E-Voting: An Advocate Recants

A former proponent of requiring the availability of e-voting system source code explains why he no longer develops source-available e-voting software.

Source-available software whether proprietary or nonproprietary (such as under open source licenses) has garnered significant attention. The arguments in favor of taking a source-available approach in general are numerous [6], but only those of specific relevance to e-voting systems will be discussed here.

Improved security. A system whose security depends on its design and implementation being secret is likely to have brittle defenses. If a system's security is based on a secret design that becomes public knowledge, then security is compromised and the genie cannot be put back into the bottle. Cryptographers and security professionals use peer review to provide assurance for the quality of their systems. A security scheme whose source code and design is known, yet continues to offer a useful level of protection, is a good one. The secrets such systems usually depend on are tokens such as passphrases or keys. If a secret token is compromised, only a particular instance of the system is compromised—not all systems of that design. A new key or password can be chosen to remedy the situation. In other words, security through obscurity doesn't work.

Unfortunately, the leading e-voting suppliers work on this principle of security through obscurity. At best, they share limited details about their system designs [4] and keep the source code closed. This provides no way for stakeholders in the election process (election administrators, candidates, voters) to verify that the software performs as the suppliers claim. A closed approach is not conducive to building confidence in e-voting systems. It puts an enormous burden of trust on suppliers who have clear motivations to conceal security failings. A source-available e-voting system, where the design and source code are freely available, is likely to be more trusted by stakeholders. Not only can the security of the system design be assessed, but bugs can be spotted by anyone who downloads and checks the code.

Transparency breeds quality. Developers are motivated to provide clean, well-commented code when they know their work will be publicly available. They do this out of pride and, if they want participation, to help others understand and contribute to the system. When errors do creep in (as bugs are inevitable), then the transparency of source availability can allow for anyone to patch holes and correct errors.

Freedom from dependencies. While sponsoring and requiring the use of open protocols such as OASIS's XML-based Election Markup Language might help prevent vendor lock-in, source availability can also play a significant role. Once an authority has purchased proprietary voting machines, open standards or not, the authority becomes dependent on the original supplier to service the machines' hardware and software. With such strong lock-in, voting systems are clearly a lucrative business—with the suppliers very much in control. Yet, with a source-available voting system installed on voting kiosks, authorities would at least have the option to maintain the software themselves or contract its ongoing support to less-expensive alternatives. Additionally, if a vendor chooses not to provide a feature that a customer desires (such as printing a paper ballot image for each vote cast), then the authority has the ability to implement the feature itself.

A source-available approach also helps to prevent municipalities from being abandoned if a provider withdraws from the market. Support and future developments, such as to support legislative changes, are made possible by source availability.

The Arguments Against Source Availability

Although a source-available approach seems promising when compared to the poor security and development practices of many e-voting suppliers today, the benefits it offers are not enough to overcome the risks.

Transparency goes only so far. The source-available rhetoric, particularly regarding nonproprietary development, is one of communally improving code. But while open peer review can improve the quality and security of systems, it requires active participation. The reality is that only a tiny minority of source-available projects attract contributions, the majority languish in obscurity with a single developer caring for the code. Generally, the software most used by programmers is that which gains the most attention. Thus, e-voting is not highly attractive to most potential contributors. My experience is that despite significant press coverage and backing from major organizations, an e-voting project is not attractive to many developers.

Arguments have also been made that source-available code's transparency reduces vulnerability to viruses. While GNU/Linux is less likely to suffer from virus attacks than Windows systems, this has nothing to do with GNU/Linux being a source-available operating system. It is partly due to it being built on a well-designed user-level security architecture derived from Unix. Additionally, the relatively low number of desktop machines running GNU/Linux makes it less attractive to virus writers who prefer the rapid viral self-propagation that a dense monoculture of Windows machines offers [5].

Now you see it, now you don't. Having the ability to review software design and source code does tend to improve its security. But while the design and code may appear sound, we have no guarantees this is what is actually used on Election Day. This is a genuine risk, illustrated with the use of uncertified software by Diebold in 17 Californian counties [1] and by Elections Systems and Software in 41 Indiana counties [2], both during 2003 primaries. Computers are black boxes where proving to voters that the software audited is that being run is a challenging problem. Code signing and certificates might have a role but, on large scales and for extensive deployments, they are not enough.

Last-minute fixes are likely, and so any update process presents the possibility that malicious code can also be inserted. Given the inherent pressures of Election Day, what would administrators do if code signatures didn't validate on all machines? A smart attacker would probably either subvert the code signing system or create an exploit that didn't require altering code that is checked.

Design control. Changes to source code are difficult to prevent in an e-voting system, whether or not it uses source-available code. The same applies at the design level. A collaboratively designed source-available system that embodied the best practices in e-voting could be built. But on installing the system a government could easily choose to modify the system or implement it poorly, thereby rendering the system less secure and less reliable. This has already occurred in Australia, where an open source system implemented a voter verifiable paper trail that the government chose to remove, thereby cutting out the costs of providing printers [7]. Of course governments have enough buying power to get any software changed, but source availability makes such a possibility only that much more likely.

Source availability doesn't change the fundamentals. Making an e-voting system's source code available doesn't alter the fundamental challenges that e-voting presents. Creating a secure, private, reliable and anonymous system that provably records voters'

intentions accurately is an extremely difficult technical problem. Source availability doesn't change the problems presented in preventing insider attacks, correctly identifying voters while protecting their identities, and in creating trustworthy audit trails that don't undermine voter anonymity. Electronic voting is challenging in terms of usability, the scale on which it needs to be implemented, and on the levels of trust we must have in the outcome. Source availability does not change any of these essential factors.

Conclusion

This article has shown that the source-available approach can offer some security and transparency benefits to the development of e-voting systems. Conversely, source-available software fails to address the fundamental challenges involved in adding technology to the voting process. The gains in system quality that source availability might offer are not sufficient to outweigh the considerable risks to the voting process that all forms of e-voting present. It was for these reasons that, after three years of extensive effort, I ceased development of GNU.FREE, the world's first open source e-voting software [3]. More recent open source initiatives such as the Open Voting Consortium in California are highly likely to encounter similar difficulties. No amount of source availability, clever design, or ingenious code can prevent poor implementation or malicious source changes. C

ELECTRONIC VOTING IS CHALLENGING IN TERMS OF USABILITY, THE SCALE ON WHICH IT NEEDS TO BE IMPLEMENTED, AND ON THE LEVELS OF TRUST WE MUST HAVE IN THE OUTCOME. SOURCE AVAILABILITY DOES NOT CHANGE ANY OF THESE ESSENTIAL FACTORS.

References

1. Ackerman, E. E-voting probe criticizes vendor. *The Mercury News*, 2004; www.mercurynews.com/mld/mercurynews/news/politics/8491288.htm.
2. Fritze, J. New voting equipment didn't pass state muster. *The Indianapolis Star*, 2004; www.indystar.com/articles/2/139985-5092-092.html.
3. Kitcat, J.P. *GNU.FREE...A Free Software Odyssey*. 2003; www.j-dom.org/h/n/WRITING/nonfict/ALL/41/.
4. Mohen, J. and Glidden, J. The case for Internet Voting. *Commun. ACM 44*, 1 (Jan. 2001), 72–85.
5. Peeling, N. and Satchell, J. Analysis of the impact of open source software. *QinetiQ*, 2001; www.govtalk.gov.uk/documents/QinetiQ_OSS_rep.pdf.
6. Stallman, R.M. *Why Software Should Not Have Owners*. Free Software Foundation, 1994; www.fsf.org/philosophy/why-free.html.
7. Zetter, K. Aussies do it right: E-Voting. *Wired News*, 2003; www.wired.com/news/ebiz/0,1272,61045,00.html.

Jason Kitcat (jeep@j-dom.org) is an e-democracy consultant and a doctoral candidate researching online government consultations at SPRU, University of Sussex, U.K.; www.j-dom.org.