The Relative Merits of Openness in Voting Systems
California Senate Elections Committee
Hearing, February 8, 2006

Peter G. Neumann
Principal Scientist, Computer Science Lab
SRI International, Menlo Park, California
333 Ravenswood Ave, Menlo Park CA 94025-3493
Telephone: 1-650-859-2375; Fax: 1-650-859-2844
Neumann@CSL.sri.com; http://www.csl.sri.com/neumann

Abstract: In my testimony today, I consider the relative potential advantages and disadvantages of voting systems that can be subjected to objective independent analysis. I emphasize that openness in the design, development, and operation of voting machines can significantly enhance the integrity, transparency, and accountability of the election process. However, openness by itself is not a cure for voting errors and clandestine manipulation, because additional discipline is required throughout the entire process. Nevertheless, openness has significant potential to promote democratic values and enhance reliability and accuracy, which at present are very difficult to ensure with closed proprietary systems (which thus far have not exerted the requisite discipline).

Introduction

To begin, I would like to express my thanks to State Senator Debra Bowen, Chair of the California Senate Elections Committee, for this opportunity to testify on the Relative Merits of Openness in Voting Systems at a hearing that is important both for its consideration of electronic voting systems and for its recognition of the vital needs for greater openness and transparency throughout the election process.

As a computer professional (since 1953), I have been involved with election system integrity, security, and privacy for almost 20 years, specifying requirements and evaluating systems, as well as analyzing system failures, alleged fraud, and human errors. This included extensive participation over a decade ago in New York City's subsequently canceled project to upgrade from lever machines to electronic voting systems.

My testimony to the California Assembly Committee on Elections and Reapportionment and Constitutional Amendments on January 17, 2001 (http://www.csl.sri.com/neumann/calvot01.pdf) stated that "The election process is inherently subject to errors, manipulation, and fraud. It is a process that demands extraordinary integrity of any computerized systems involved, as well as honesty and experience of the people involved in administering elections. Evidently, it may require considerable sophistication on the part of voters as well."

My testimony to the same committee on June 15, 2004 (http://www.csl.sri.com/neumann/calvot04.pdf) reiterated some of the points from the earlier testimony, and concluded that the criticality of the system integrity problems has evidently intensified in the interim rather than abated. This is partly a result of the post-2000 feeding frenzy to acquire all-electronic voting machines that, in the absence of voter-verified audit trails, provide *no substantive assurances* that votes are correctly processed. Vendor claims to the contrary are disingenuous and not credible, for a variety of reasons: voluntary lax evaluation criteria, proprietary software, proprietary interfaces, evaluations that are proprietary and commissioned by the vendors, and the reported presence of convicted felons in company personnel, along with other clear evidence in recent elections of claims that are not justified.

Unfortunately, voting is a process in which there are many weak links throughout; of particular concern here is the reality that all voting systems today are subject to varying degrees of errors, manipulation, and nonaccountability — especially where there is a serious lack of oversight and definitive audit trails. As a technologist, I strive to establish checks and balances not only on the technology but also on the voting process as a whole. As legislators, you have an obligation to ensure that you do not endorse simplistic solutions that could in actuality further weaken the integrity of the election process, with even greater opportunities for fraud, subversion, and undetected errors. Fortunately, government executives and legislators, election commissioners, voters, and system developers are slowly comprehending the extent of the risks involved in having election systems without pervasive systemic attention to overall integrity and operational oversight.

Some important goals are that voting systems should be easy to use by all eligible voters, should be extensively evaluated against rigorous standards, and should be demonstrably capable of satisfying certain critical requirements — not merely for system security but also for voter privacy, system reliability and fault tolerance, and system survivability in the face of a wide range of realistic adversities that include hardware malfunctions, software glitches, inadvertent human actions, coordinated attacks, and acts of God. Also relevant are additional operational requirements such as system usability by communities that are linguistically diverse, disabled, or otherwise disadvantaged, system interoperability among different vendors' products to avoid putting all your eggs in one basket, rapid maintainability, and long-term ease of evolvability, as well as the enforcement of appropriate discipline throughout software development and administration.

Openness in Voting Systems

Despite considerable past research in the design, implementation, and operation of high-assurance trustworthy systems, development of commercial systems is decidedly suboptimal with respect to meeting stringent critical requirements that include security and privacy. Primary concerns include the transparency of system development and use, and especially the extent to which the software itself is open to thorough independent inspection.

As used here, the word *openness* has a straightforward natural meaning, implying the ability to conduct impartial and objective oversight. Various needs for openness exist throughout the entire election process, from voter registration to voter authentication to voting itself and the collection and canvassing of the results, and — in particular in the present context — throughout the entire life cycle of the computer systems and other equipment used in the process.

To be precise about our terminology, we distinguish here between *black-box* (that is, closed-box) systems in which source code is not available for inspection, and *open-box* systems in which source code is available under a set of specified conditions that may apply to its use, modification, reuse, and further distribution, as well as to the relative openness of the development process itself. Black-box software is often considered advantageous by vendors and believers in *security by obscurity* ("just trust us"). However, black-box software makes it much more difficult for anyone other than the original developers to discover vulnerabilities and surmount them. It also hinders open analysis of the development process (which is something many developers are usually happy to hide). Overall, it can be a serious obstacle to having unbiased confidence in the ability of a system to fulfill its requirements for integrity, security, privacy, reliability, correctness, and so on, as applicable.

We also distinguish here between *proprietary* and *nonproprietary* software. As noted above, open-box software can come in various proprietary and nonproprietary flavors, with various licensing agreements. Examples of nonproprietary open-box software are found in the Free Software Movement (such as the Free

Software Foundation's GNU system with Linux, and its General Public License) and the Open Source Movement [1], although discussions of the distinctions between those two movements and their respective nonrestrictive licensing policies are beyond the scope of this brief testimony. In essence, both movements believe in and actively promote unconstrained rights to modification and redistribution of open-box software, and use of nonproprietary interfaces. However, internal confusions among the different communities tend to mask the more important issues discussed here. Furthermore, the question of who can do what to the source code is less important to the voting process than the question of what is hidden from view.

The potential benefits of open-box software include the ability of external experts to carry out peer reviews, identify flaws, get them fixed rapidly, and perhaps even add improved functionality — for example, through collaborative efforts of a wider community. Open specifications and nonproprietary interfaces are particularly relevant when critical responsibilities are outsourced, although then oversight of trusted (but potentially untrustworthy) outsourcees becomes critical as well. Of course, the risks include increased opportunities for evil-doers to discover flaws that can be exploited, and to insert malicious code into flawed software. Clearly, certain controls must be exerted over software developers, certified source-code repositories, and trustworthy software distribution, Especially important are issues of trustworthy configuration management — for example, with rigorous assurances that any software that is actually used in elections is precisely the same software that has been subject to scrutiny. This must include operating systems, compilers, source code, and executable code for the entirety of any voting system.

Of particular interest is the potential for the use of disclosable software in extremely trustworthy systems, in light of (for example) the Internet, typically flawed operating systems, vulnerable system embeddings of strong cryptography, and the presence of internal potentially hostile code and remotely insertable malware. A system architecture question involves where trustworthiness must be placed to minimize the amount of critical code and to achieve robustness in the face of specified adversities. For example, it is possible to develop systems in which vote preparation systems do not have to be trusted, because of other checks and balances. This follows the architectural approach described in [2] of isolating a minimal disclosable portion of a system that must be trustworthy, which is vastly preferable to a nondisclosed proprietary system that must be blindly trusted.

A frequently asked question is "Can open-box software really improve system security?" The answer is that *it does not necessarily do so by itself, although the potential is considerable.* Many other factors must also be considered. Indeed, many of the problems of black-box software can also be present in open-box software, and *vice versa* (for example, flawed designs, the risks of mobile code, difficulties in finding gifted system administrators, and so on). In the absence of significant discipline and inherently better system architectures, opportunities may be even more widespread for insertion of malicious code in the development process, and for uncontrolled subversions of the operational process. But perhaps most important is the need for both openness and discipline throughout the election process — particularly with respect to disclosure and independent analysis of source code and even object code, and irrespective of any particular open-box licensing model.

We face a basic conflict between (a) *security by obscurity* that is supposed to slow down the adversaries, and (b) *security by open design* that allows for independent analysis [3] and collaborative improvement of critical systems – as well as providing a forcing function to inspire improvements in the face of discovered attack scenarios. Ideally, if a system has been designed carefully to be meaningfully secure, an open development process, open specifications, disclosable source code, and even open evaluations should not be a significant benefit to attackers, and the defenders should be able to maintain a competitive advantage! For example, this is the principle behind using strong openly published cryptographic algorithms — for which open analysis of algorithms and their implementations is very valuable, and where only the private keys need to be hidden. Other examples of obscurity include tamper-resistance and obfuscation. (Note that tamper*proofing* is basically impossible, especially in the presence of untrustworthy insiders and outsourcees.)

Unfortunately, many existing systems tend to be poorly designed and poorly implemented, with respect to incomplete and inadequately specified requirements, and are therefore susceptible to insider misuse and undetected errors, and perhaps also to outsider misuse. Developers are then at a decided disadvantage, even with black-box systems.

Unfortunately, in poorly designed and poorly implemented systems, the flaws may be identifiable whether or not the system is open for external analysis, in which case security by obscurity is essentially a sham. The most glaring deficiency in today's direct-recording (e.g., touch-screen) voting systems is the total absence of any ability for voters to verify that their votes are correctly recorded and processed throughout an election. (The most obvious short-term approach to overcome this fundamental defect is the so-called voter-verified audit trail [4], which can detect inconsistencies between cast ballots and recorded ballots, and can in effect independently overcome failures in the rest of the system.) For this reason, researchers and developers with significant expertise in computer security and voting systems vastly prefer inherently sound system designs, open disclosure of the design process and of the source code, extensive open analysis of the software, and avoidance of misguided security by obscurity.

Development, Evaluation, and Operation of Trustworthy Voting Systems

Behavioral application requirements such as reliability, real-time performance, and usability cannot be realistically achieved unless the systems are adequately secure. It is very difficult to build robust applications based on proprietary black-box software that is not sufficiently trustworthy. It is even more difficult to do so when the full set of requirements is not considered from the outset.

Several 1956 papers, by Edward F. Moore, Claude Shannon, and John von Neumann, showed how to construct reliable components out of less reliable components. Later work on correct behavior despite some number of arbitrarily perverse faults followed along those lines. In that context, building a fault-tolerant silk purse out of less robust sow's ears is indeed possible in some cases. But constructing more trustworthy secure systems out of less trustworthy subsystems does not seem realistic when the underlying components are fundamentally compromisible, despite various palliative technological efforts.

There are many arguments in favor of allowing various forms of open review — for example, oversight and analysis of requirements, designs, specifications, source code, executable code, and operational practices. Even when some obscurity or proprietary nondisclosure is deemed acceptable, wider-community approaches to openness should be considered. For software and system applications in which security can be assured by other means and is not compromisible within the application itself, openness has particularly great appeal. In any event, it is always unwise to rely *solely* on security by obscurity.

So, what else is needed to achieve robust systems that are predictably trustworthy? The first-level answer is the same for open-box systems as well as closed-box systems: serious discipline throughout the development cycle and operational practice, use of good system architectures that inherently reduce the critical dependence on potentially untrustworthy components (or otherwise compensate for inconsistencies in those components), good software engineering practices, rigorous evaluations of systems in their entirety throughout their life cycles, and enlightened management can all help.

A second-level and rather more technological answer involves inherently robust and secure evolvable architectures with open interfaces that enable interoperability among different vendor products, that avoid excessive dependence on potentially untrustworthy components, and that ensure that the critical components are indeed trustworthy (for example, servers, firewalls, code distribution paths, nonspoofable provenance for critical software, cryptographic coprocessors, tamper-resistant embeddings, preventing denial-of-service attacks, runtime detection of malicious code and deviant misuse, and so on). Of particular relevance to both black-box and open-box software in that context is a report [2] on how to develop complex systems as predictable compositions of well-analyzed components. A recent book [5] provides a compendium of diverse chapters on “Free and Open-Source Software” and is recommended reading. Also relevant is the Open Voting Consortium (openvotingconsortium.org), which is dedicated to transparent democracy through accountable election systems.

A third-level answer is that there is still much research yet to be done (such as on inherently sound architectures that starkly minimize what must be trustworthy [2], realistic predictable composition of systems out of carefully evaluated components [2], trustworthy configuration management, rigorous vulnerability and risk analysis, and open-box business models), as well as more efforts to bring that research into practice. It must be more widely recognized that every step in the process is a potential weak link, and that aggressive measures must be taken to ensure adequate oversight. In the long run, effective technology transfer seems more likely to happen in systems that can be subjected to independent scrutiny.

Conclusion

Nonproprietary open-box systems are by themselves certainly not a panacea, either in general or as applied specifically to electronic voting systems. However, openness has many potential benefits throughout the process of developing and operating critical systems. Impressive beginnings already exist in communities that pursue various forms of open-box software. Nevertheless, much effort remains in providing the necessary development discipline, adequate controls over the integrity of the emerging software, system architectures that can evolvably satisfy critical requirements, and well documented demonstrations of the benefits of openness in the real world. If nothing else, openness successes may have a forcing function on closed-box developers, who could if suitably motivated rapidly adopt the best of the results. In any event, openness will remain the ultimate need if we are to attain correct, usable, reliable, auditable, and transparent elections.

References

[1] The Free Software Foundation website is `http://www.fsf.org`, and contains software, projects, licensing procedures, *etc.* The Open Source Movement website is `http://www.opensource.org/`, which includes Eric Raymond’s *The Cathedral and the Bazaar* and the Open Source Definition.

[2] Peter G. Neumann, *Principled Assuredly Trustworthy Composable Architectures,* Computer Science Laboratory, SRI International, Final report, SRI Project 11459, Menlo Park, California, December, 2004. (This report enumerates various approaches to developing predictably dependable systems. It also includes discussion of the potential risks and benefits of outsourcing, which are relevant to voting systems.) (`http://www.csl.sri.com/neumann/chats4.html`, .pdf, and .ps).

[3] Representative analytic tools applicable to source code analysis include Crispin Cowan’s StackGuard (`http://immunix.org`), David Wagner’s buffer overflow analyzer (`http://www.cs.berkeley.edu/~daw/papers/`), Cigital’s ITS4 function-call analyzer for C and C++ code (`http://www.cigital.com/its4/`), and @Stake’s L0pht security review analyzer slint.

[4] Rebecca Mercuri, Electronic Vote Tabulation Checks and Balances, Department of Computer Science, University of Pennsylvania, 2001 (`http://www.notablesoftware.com/evote.html`).

[5] Joseph Feller, Brian Fitzgerald, Scott A. Hissam, and Karim R. Lakhani, editors, *Perspectives on Free and Open Source Software,* MIT Press, Cambridge, Massachusetts, 2005.

Biographical and Organizational Information

Peter G. Neumann (Neumann@CSL.sri.com) has doctorates from Harvard and Darmstadt. After 10 years at Bell Labs in Murray Hill, New Jersey, in the 1960s, during which he was heavily involved in the Multics development jointly with MIT and Honeywell, he has been in SRI International's Computer Science Laboratory since September 1971. (SRI is a not-for-profit R&D institute founded in 1946.) Neumann has long been concerned with computer systems and networks, trustworthiness/dependability, high assurance, security, reliability, survivability, safety, and many risks-related issues such as voting-system integrity, crypto policy, social implications, and human needs including privacy. He moderates the ACM Risks Forum, edits CACM's monthly Inside Risks column, chairs the ACM Committee on Computers and Public Policy, and chairs the National Committee for Voting Integrity, NCVI (http://www.epic.org/privacy/voting). He created ACM SIGSOFT's Software Engineering Notes in 1976, and was its editor for 19 years — and still contributes the RISKS section. His 1995 book, Computer-Related Risks, is still in print. He is a Fellow of the ACM, IEEE, and AAAS, and is also an SRI Fellow. He received the National Computer System Security Award in 2002 and the ACM SIGSAC Outstanding Contributions Award in 2005. He is a member of the U.S. Government Accountability Office Executive Council on Information Management and Technology, and the California Office of Privacy Protection advisory council. He has taught courses at Darmstadt, Stanford, U.C. Berkeley, and the University of Maryland. See his website (http://www.csl.sri.com/neumann) for further background, U.S. Senate and House testimonies, papers, bibliography, etc.

Neumann is currently one of the principal investigators of the NSF-sponsored ACCURATE project (A Center for Correct, Usable, Reliable, Auditable and Transparent Elections), which also includes faculty and students at the University of California at Berkeley, the University of Iowa, Johns Hopkins University, Rice University, and Stanford University as well as SRI International. The ACCURATE effort is considering a wide range of fundamental research directly relevant to voting systems, but also much more widely applicable to accountable trustworthy systems.

The National Committee for Voting Integrity (NCVI) brings together experts on voting issues from across the country to promote constructive dialogue among computer scientists, elections administrators, voting rights advocates, policymakers, the media and the public on the best methods for achieving in practice: fair, reliable, secure, accessible, transparent, accurate, accountable, and auditable public elections. NCVI is a project of the Electronic Privacy Information Center (EPIC), which is a public interest research center in Washington, D.C. EPIC was established in 1994 to focus public attention on emerging civil liberties issues and to protect privacy, the First Amendment, and constitutional values. One fundamental constitutional value is the right to vote, and for this reason EPIC began the NCVI project in 2003 when discussions about remedies for the failures of the nation's election process during the 2000 election rested solely on the adoption of electronic voting systems.