Lost Treasures

Dan Thomsen | SIFT, LLC
Jeremy Epstein and Peter G. Neumann | SRI International

Is the computer security field really old enough to have lost treasures? Will a granite punch card with ancient Cobol contain some code fragment that produces a better firewall? Hardly. The computing environment changes so much and so radically that implementation details lose relevance quickly. However, concepts and key insights can serve modern developers just as well today as they did builders of ancient systems from the 1980s (for instance, see the "Lost Lessons: Election Systems" sidebar). New security practitioners' conference submissions often reinvent key security concepts, highlighting the need to uncover these lost gems. The future should build on the past, not constantly reinvent it.

Economics and Archaeology

Eras of rapid change foster great economic opportunities as people continually adapt to the changing environment. All this money shifts the industry's focus to getting the next release out the door. In this flurry of change, it's easy to lose track of key concepts. In contrast, look at mathematics. Often, centuries elapsed before someone found a useful way to apply a new theorem.

So what can we do for computer security in this era of rapid change? We could have people pore over old government documents, corporate reports, and papers looking for lost gems. The payoff remains comparable to traditional archaeology: you have to move a lot of dirt before you find something worth keeping. For example, the "Rainbow Series"—a set of government computer security guidelines (http://csrc.nist.gov/publications/secpubs/rainbow)—contains a lot of good computer security science but is encased in the strata of multicolored evaluation criteria. And despite our gentle cajoling, no one wrote an article for this issue on the science of the "Orange Book"—the cornerstone of the Rainbow Series that addresses building a secure

Lost Lessons: Election Systems

by Peter G. Neumann

This special issue features insights on computer security that are particularly relevant to election integrity. Thus, it seems appropriate to consider some of the lost or hidden treasures in that context.

Past elections have illustrated a serious lack of commitment to total-system integrity, accountability, and privacy, as well as weak defenses against insider misuse. We observed many flagrant election irregularities, some of which are directly related to computer systems, others of which are procedural, operational, or seemingly extrinsic. These problems tend to arise owing to the absence or inadequacy of total-system requirements, flawed computer system design and implementation, incomplete or misguided security evaluations, and so on. For example, there was significant tampering in the 1988 Florida Senate race, with a 14 percent drop-off in senate votes compared with presidential votes in the four punched-card counties (there was essentially no drop-off in the non-punched-card counties).[1] In Louisiana, 22 people were indicted in a bribery/kickback scheme in 1999, and nine people in Clay County, Kentucky (including election officials and a federal judge), were indicted for election fraud in 2002, 2004, and 2006.[2] The 2006 fraud involved poll workers intentionally misleading voters about the user interface, informing them that clicking on the cast vote button would cast their votes, whereas a subsequent confirmation button was required. The screen also included an alternative back button, which allowed poll workers to alter the votes. Numerous other problems also arose in Florida in 1988 and 2000, Ohio in 2004, and so on, with registration, voter disenfranchisement and biased authentication, and vote counting. These experiences—including the reports of the 2007 California Top-to-Bottom Review—all suggest that many past lessons have been almost completely ignored.

Various generic security principles in the literature directly apply to beginning-to-end and top-to-bottom election integrity but have been largely ignored in commercial systems. Such treasures include Kerckhoffs's principle (avoiding security by obscurity), the Saltzer-Schroeder-Kaashoek security principles,[3,4] and the Clark-Wilson integrity properties.[5] In particular, many of today's systems ignore the principle of minimizing the extent of components that must be trusted. This can compromise every step in the election process, from standards establishment to voter registration (disenfranchising, gerrymandering, and so forth) to voter authentication (multiple IDs required in certain cases) to vote casting (machines observed switching votes) to vote tabulation and canvassing to dispute remediation (especially in the absence of any meaningful audit trails, much less forensically meaningful nonsubvertible audit trails). Compromises are also possible at each layer of abstraction, including hardware, operating systems, and election-specific software and data formats. Insider attacks remain particularly risky owing to the opportunities presented and the ease of triggering serious irregularities.[6]

There are many other application areas in which these lessons are salient. However, election integrity is a paradigmatic example of why this special issue is more important than ever. Specifically, revisiting some of the lost treasures and problems that might otherwise have been avoided highlights many deeper issues. Consider the risks of outsourcing functionality and responsibility to unaccountable closed source proprietary systems; short-sighted and unprincipled system development; lack of understanding of the risks; and the continued lack of win-win solutions owing to debilitating tradeoffs among features including ease of use, accountability, cost savings, and trustworthiness with respect to security and integrity. Unfortunately, trustworthiness is often the first to be reduced.

References

1. P.G. Neumann, "Computers in Elections," *RISKS Forum*, vol. 7, no. 78, 1988; http://catless.ncl.ac.uk/Risks/7.78.html#subj1.
2. P.G. Neumann, "Kentucky Election Fraud Indictments," *RISKS Forum*, vol. 25, no. 76, 2009; http://catless.ncl.ac.uk/Risks/25.76.html#subj7.
3. J.H. Saltzer and M.D. Schroeder, "The Protection of Information in Computer Systems," *Proc. IEEE*, vol. 63, no. 9, 1975, pp. 1278–1308.
4. J.H. Saltzer and F. Kaashoek, *Principles of Computer System Design*, Morgan Kauffman, 2009.
5. D.D. Clark and D.R. Wilson, "A Comparison of Commercial and Military Computer Security Policies," *Proc. Symp. Security and Privacy*, IEEE CS, 1987, pp. 184–194.
6. P.G. Neumann, "Combatting Insider Threats," *Insider Threats in Cybersecurity—and Beyond*, M. Bishop and C.W. Probst, eds., Springer, 2010.

operating system—possibly because it would require significant effort.[1]

In traditional archaeology, people look for artifacts to determine how people lived in the past. By looking back at security papers, we can determine how people thought about computer security. Paul Karger and Roger Schell pointed out,[2] and Ken Thompson made famous in his article "Reflections on Trusting Trust,"[3] that a Trojan horse can latch itself onto a compiler and duplicate itself in all the applications compiled with the compiler, even new versions of the compiler. This was far-fetched thinking at the time. In addition, David Elliot Bell and Leonard LaPadula determined the need for the operating system to enforce properties so users'

agents can't copy information to a lower classification and learned that although we can trust humans not to downgrade information because they're cleared, a Trojan horse working on a user's behalf could.[4] When Bell and LaPadula discovered this key property, they didn't even name it; they identified it with "* property," where the asterisk represents a placeholder for some future name. Yet this unnamed property still drives high-assurance systems. Both these articles shifted the way we thought about computer security, which was much simpler before these articles were published. Yet, today we still have trouble getting new security practitioners to believe just how tricky attackers can be. If you work in computer security long enough, you'll hear people say, "no one would do that" or "where does this paranoia end?" However, if you can imagine a way to compromise a system, attackers can implement it.

In This Issue

If we can recreate that "Eureka!" moment in new security practitioners, we can foster a computer security science that builds on the past rather than reinventing it for the latest new fad. The articles in this issue attempt to do this.

In "A Contemporary Look at Saltzer and Schroeder's 1975 Design Principles," Richard E. Smith tracks the use and evolution of the concepts from Jerome Saltzer and Michael Schroeder's seminal 1975 article, "The Protection of Information in Computer Systems."[5] As the technology and political climate changed, the way people applied and restated these principles likewise changed.

Computer security science is really a science of engineering, constantly trading functionality and security to mitigate system risk. In "Lessons from VAX/SVS for High-Assurance VM Systems," Steve Lipner, Trent Jaeger, and Mary Ellen Zurko give us a blow-by-blow account of the engineering insights and decisions from building a high-assurance operating system in the early 1990s.

What if security research got a "do-over"? Howard Shrobe and Daniel Adams ask and answer this question in "Suppose We Got a Do-Over: A Revolution for Secure Computing," looking at DARPA programs designed to examine security without the economic shackles of compatibility requirements that current systems face.

How do we know when we achieve security? Evaluation and certification are supposed to be the yardsticks for measuring it, but they can fail just like any other system component. Steven J. Murdoch, Mike Bond, and Ross Anderson examine the certification standards from the 1970s on and trace their evolution and impact on deployed security in "How Certification Systems Fail: Lessons from the Ware Report."

Finally, Jeffrey T. McDonald and Todd R. Andel look at how key insights can improve information assurance education in "Integrating Historical Security Jewels in Information Assurance Education." Starting from 1883, they examine system design principles that every security practitioner today should know.

To round out this issue, we sent Earl Boebert, a long-term computer security practitioner, out with his trusty shovel to comb the sands of information security for lost gems of computer security. He filled his sack up in no time. Boebert's *The Fox Herders Guide: How to Lead Teams That Motivate and Inform Organizational Change* (Bitsmasher, 2011) contains many of these gems. Others come from Rick Proto, who was director of research at the National Security Agency early in Boebert's career. Rather than providing a single list of these gems, we've scattered them throughout the issue. The reader will benefit from deeply considering each of them.

Happy hunting!

Each of these treasures presents a key shift in a security practitioner's thinking. We need to pinpoint these shifts, not as a matter of historical documentation, but because new security practitioners need to undergo those same shifts in thinking to produce the next wave of computer security engineers. ■

References

1. *Trusted Computer System Evaluation Criteria*, US Dept. Defense, Nat'l Computer Security Center, report 5200.28-STD, Dec. 1985
2. P.A. Karger and R.R. Schell, *Multics Security Evaluation: Vulnerability Analysis*, ESD-TR-74-193, vol. II, HQ Electronic Systems Division, June 1974; http://csrc.nist.gov/publications/history/karg74.pdf.
3. K. Thompson, "Reflections on Trusting Trust," *Comm. ACM*, vol. 27, no. 8, 1984, pp. 761–763.
4. D.E. Bell and L. LaPadula, *Secure Computer Systems: Unified Exposition and Multics Interpretation*, tech. report, MTR-2997, MITRE, July 1975.
5. J.H. Saltzer and M.D. Schroeder, "The Protection of Information in Computer Systems," *Proc. IEEE*, vol. 63, no. 9, 1975, pp. 1278–1308.

Dan Thomsen is a principal researcher at SIFT, LLC. Contact him at d.j.thomsen@ieee.org.

Jeremy Epstein is a senior computer scientist at SRI International. Contact him at jeremy.j.epstein@gmail.com.

Peter G. Neumann is principal scientist at SRI International's Computer Science Lab. Contact him at neumann@csl.sri.com.