Rebecca T. Mercuri

Computer Security: Quality Rather than Quantity

The challenge of applying protection to systems, software, and networks with intrinsic vulnerabilities is a lofty, but ultimately realizable, one.

Programming (and also secure system design), as Donald Knuth so wisely pointed out decades ago, is an art, as much, and perhaps even more, than it is a science. As such, it should be judged on Quality, and Quality often demands less, not more, in terms of quantity. The larger the software, the more difficult it is to maintain, assure, and protect. Therefore more code (or more hardware) does not necessarily translate to good Quality. Software engineering approaches focusing on code review, development cycles, configuration management, and so on, adding more complexity to the process, and cannot necessarily, in themselves, ensure Quality.

Current practices using a reactive approach to computer security, where firewalls, cryptography, and other add-ons are applied to existing implementations in order to make up for deficiencies, tend to increase code size. Another reactive practice, that of waiting for breaches to occur and issuing corrective patches after detection, is a solution (if one could call it that) requiring ongoing end user or system manager updates in order to maintain product integrity. This "throw more code at it" philosophy is encouraged by advancements in the computer hardware sector, since potential performance compromises are generally offset by faster CPUs and larger storage capacities. Recently, vendors have begun to automate the update process by subjecting networked computers to (often unrequested and unauthorized) automated downloads, a violation of security that occasionally causes complications and has vast potential for nefarious exploitation. Even if this is a viable approach, eventually the resulting patchwork quilt of installations becomes unmaintainable, and customers are pressured to purchase an "upgrade version" with new "security features" promising to resolve the old issues, but typically overlooking, and hence often introducing, new problems.

Of course, one response to the expanding software situation is to add more hardware. The latest of these proposed mechanisms is Palladium, Microsoft's venture with its hardware partners (Intel, AMD, and others), a Windows product intended for release in 2004. According to Mario Juarez, group product manager of the Palladium team: "It's designed to give people greater security, personal privacy and system integrity"[7]. The system plans to include cryptographic and other specialized chip components, functioning through a Trusted Operating System Root. And therein lies the rub.

One might recall this is not the first time a chip-based security

MICHAEL SCHRÖTER

solution has been offered in the Windows context. The earlier Pentium III processor serial number concept, withdrawn slightly over a year after its introduction, was both easy to crack and a privacy concern. These difficulties aside, any go-between system (such as Palladium) is always highly vulnerable at its ends. The FBI still will be able to read your email when they impound your systems on a search-and-seizure warrant for some suspected criminal offense, unless your computer constantly performs a low-level wipe of pertinent memory segments (display, cache, virtual, and otherwise). Plus there's nothing to prohibit legitimate recipients from forwarding your memoranda "in the clear."

Unintended transmission of information can be highly embarrassing, with potential legal consequences. An example of this was when Hewlett-Packard CEO Carly Fiorina's voicemail to HP CFO Bob Wayman regarding the Compaq merger was circulated [2]. Rick Shaw of CorpNet Security was quoted as saying: "The key thing to remember is that voice mail is now digitized. It's just a file out there on a server on a hard drive. And if you don't protect your server and your phone room, it's another hard drive anybody can access." Including those who have been entrusted with permission to access it.

These problems exist in the Unix/Linux/open source world as well as with the proprietary PC/Mac operating systems, since the preponderance of security violations results from an exploitation of vulnerabilities and/or a violation of trust. Trust has become a popular descriptor in the security community that is used, perhaps, too lightly. Who should we trust, and under what circumstances? What mechanisms are used to grant and ensure trust, and how are those monitored? Is Intel CTO Patrick Gelsinger's vision of a billion networked trusted computers operating in a trusted virtual world plausible or even desirable? [3] One is reminded of Ken Thompson's *Reflections on Trusting Trust*: "You can't trust code that you did not totally create yourself. (Especially code from companies that employ people like me.)"[5]. These thoughts are good for openers, and the topic of trust is a broad area I hope to discuss in future columns.

Best Practices for Vulnerability Assessment

- Identify the assets and processes at risk.
- Focus on business risk, not technology.
- Look beyond the IT turf (consider security impact of facility and human resource policies).
- Use available automated tools for technical vulnerability scans.
- Anticipate legal obligations to ward off intruders and prevent involvement in distributed attacks.
- Consider nonelectronic information (shred sensitive input and output forms; evaluate nonmagnetic backups, for example, microfiche).
- Measure what really matters (lost time, not success rate in blocking attacks; intrinsic value of lost or acquired data after a violation).

What must be asked, though, is what amount of trust is essential for security? It often appears the amount of trust we are willing to accept is, in some sense, reciprocal to our security needs.

Sometimes these needs are difficult to identify. A proactive approach, where risk assessments and protection measures are incorporated during the design process, listed in an *eWeek* column [4] suggests a number of best practices (modified slightly in the accompanying box) for vulnerability assessment.

The value of this list is it directs our attention outward, to alternative solutions that aren't always formed from "cutting-edge" technologies. Where paper and pen are better, use them. If your computer contains highly sensitive information, don't connect it to the Internet. What should result from assessment is better overall product construction. Here, are some helpful techniques.

At the high end of the secure system design spectrum is the ISO Common Criteria [6], a proactive methodology that, in its extreme, full provability, is theoretically

impossible to apply except in severely limited cases. The Common Criteria does provide a framework of accountability for security claims and establishes a level of confidence in the assurance of security functions. In this way, products can be compared as to their effectiveness and applicability to scenarios with varying security needs.

Unfortunately, due to industry dependence on legacy systems and products requiring backward compatibility, the strongest aspects of the Common Criteria are difficult to satisfy, since they involve integration at the time of design inception. However, this does not preclude its use as a benchmark standard for system evaluation. Other, less stringent, security design guidelines exist, such as the International Information Security Foundation's Generally Accepted System Security Principles [1].

Before design principles can be applied, an understanding of the nature of security is required. Can the presence of adequate security be determined and its value ascertained? Does security involve direct actions in order to provide assurances about the resources being protected? Dictionary definitions describe security in terms of measures adopted to guard against theft, espionage, sabotage and attack, or those applied in order to prevent losses from occurring. Another set of definitions speaks of security as a state of freedom (from the negative aspects of risk, danger, doubt, anxiety, fear) or in the affirmative, as a sense of safety, confidence, and certainty.

In the wake of the 9/11 terrorist attacks, heightened attention to security in the U.S. and other affected countries has been perceived as a way of maintaining or guaranteeing freedom. Yet many of the security controls applied thus far have translated into losses of personal freedom through reduced mobility and increased invasion of privacy. One must ponder how to weigh freedom from terrorism (if such is even possible) against freedom of activity, and the benefits and demerits of such tradeoffs.

At the 2001 RSA conference in San Francisco, Microsoft Vice President Dave Thompson said: "Security is a journey, not a destination." But how far is that journey? After all, humankind brings the double-edged sword to the table; people want security to protect themselves from the people who are intent upon violating it.

One might assert that until we have a firm understanding of the complex machinations of the human mind, we can never hope to eliminate the Trojan Horses of our cyberfuture. Indeed, although there are some who would like to use technology to observe our very thoughts, one shudders at the Orwellian implications of a society that would consider the implementation of such drastic measures in the name of security.

Some relief from these gloomy prospects for the future can be found in the words of Robert Pirsig, from his classic volume *Zen and the Art of Motorcycle Maintenance*: "…what really causes technological hopelessness is absence of the perception of Quality in technology by both technologists and antitechnologists." Pirsig goes on to say, "When traditional rationality divides the world into subjects and objects it shuts out Quality, and when you're really stuck it's Quality, not any subjects or objects, that tells you where you ought to go."

Of course the nature of Quality is rather elusive, but he lends us a clue in the statement: "By returning our attention to Quality it is hoped that we can get technological work out of the noncaring subject-object dualism and back into craftsmanlike, self-involved reality again, which will reveal to us the facts we need when we are stuck." When we look at computers and their peripheral networks and data stores as objects requiring control or protection, and programming and design as subjects, or vice versa, we lose our sense of focus on the essence of the security we attempt to provide. By encouraging artistry and applying craftsmanship to our security problems, viable solutions will emerge.

One way of starting this process is by defining computer security with respect to need. Since need is a Quality (not a subject or object), it requires the consideration of balances and tradeoffs. The application of pedantic heuristics simply will not suffice. Our present security challenges have heretofore largely

stemmed from the need to apply protection to systems, software, and networks that contain intrinsic vulnerabilities. It has been perceived as more expedient and cost-effective to modify and attempt to adapt existing systems, but in practice this approach has not provided adequate results.

Only by taking the time to consider our needs in a broader, proactive way, will we be able to envision different, innovative system models. The future must allow us to engage in the back-to-scratch development of entirely new paradigms and infrastructures, from the ground up, if we are to have any hope of success in advancing technology's ability to adapt to our various and changing security requirements. This is a lofty challenge, but one I believe ultimately can be overcome. C

References

1. Best Practices. Vulnerability assessment. *eWeek*, Dec. 19, 2001.
2. Common Criteria Implementation Board. *Common Criteria for Information Security Evaluation*. ISO IS 15408, 1999; csrc.nist.gov/cc.
3. Gelsinger, P. A billion trusted computers. RSA Data Security Conference Keynote, (Jan. 20, 1999); www.intel.com/pressroom/archive/speeches/pg012099.htm.
4. Hachman, M. and Rupley, S. Microsoft's Palladium: A new security initiative. *PC Magazine*, June 25, 2002.
5. International Information Security Foundation. Generally-accepted system security principles, June 1997; web.mit.edu/security/www/GASSP/gassp021.html.
6. Searching for votes. *San Jose Mercury News*, (Apr. 9, 2002).
7. Thompson, K. Reflections on trusting trust. *Commun. ACM 27*, 8 (Aug. 1984); www.acm.org/classics/sep95.

Rebecca Mercuri (mercuri@acm.org) is a professor of computer science at Bryn Mawr College and the president of Notable Software, Inc., Princeton, NJ.