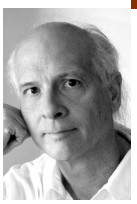
RIK FARROW

musings



Rik is the Editor of *;login:*.

rik@usenix.org

I'VE BEEN ACCUSED, RIGHTLY, OF BEING pessimistic about computer security, and recent events have only increased that pessimism. But rather than tire you with my grumblings, I thought I would take a dispassionate look at computer security as it exists today and make positive suggestions about what you might do, whether in your professional or personal lives.

I'll start out with something you might find surprising, considering the source: if you, or people you know or work with, use Windows XP, convince them to upgrade. The same goes for people using anything earlier than Server 2008.

Microsoft began its Trustworthy Computing Initiative in 2002 and has paid much more attention to security in recent years. Some of the fruits include more reactive security measures, such as DEP (data execution prevention) and ASLR (address space layout randomization), although these are not used in all applications. Internet Explorer 7 prior to SP1 is one of those applications that is not protected with either DEP or ASLR for application compatibility, but later versions are, as is IE8.

Both IE7 and IE8 also rely on Integrity Levels [1], an ACL mechanism where less trusted processes, such as Web browsers, get run with a low integrity level. Processes with low integrity levels have limited or no access to files, processes, or other objects (e.g., registry keys and named pipes) at higher integrity levels—which means, most of the system.

These are good things. I kept hearing from my friends in security that Windows had gotten a lot more secure—but they wouldn't or couldn't provide strong evidence that these mechanisms actually help. Then I learned from Niels Provos, whose Google team searches the Web for malicious sites, that it was much more difficult for most exploits to work with IE7 or IE8. While his team's goal is to find pages that lead to exploits on any version of Windows, I found this interesting news, as they actually test hundreds of millions of pages in their Windows equipped sandboxes (see "The Nocebo Effect," p. 18).

Crispin Cowan, the inventor of stack canaries, also known for Immunix and AppArmor, began working for Microsoft in 2008. Cowan spoke at the 2010 USENIX Security Symposium, allegedly

about the security features of Windows 7 but actually about how Microsoft had sometimes been the first vendor to include new security features. I have it on good authority [2] that such talk is security theater, but you can watch the video of his presentation and decide for yourself [3]. You can also read the summaries of his talk and that of Roger Johnston, the person who describes Cowan's talk as security theater, in this issue.

One point Cowan made that really struck me was this: in 2010, the number of applications that needed administrative privileges to run had been reduced from 900,000 to just 180,000 (49 minutes into the video). I was dumbstruck.

I always knew there were lots of Windows applications, but that there are nearly two hundred thousand that need to run as root just astounded me. Cowan works as a senior project manager on User Access Control, what he called "the moral equivalent of sudo." So running these apps requires sudo to the admin group. You might not need to run any of these apps, but now you know what UAC is doing for you, or allowing these apps to do to your system.

Dark Side

So Windows has its dark side. We've always known that. The need for running apps with privileges has to do with the history of Windows NT, which Cowan also covered earlier in his talk. In 1995, NT had no applications, so by adding the Windows 32-bit API libraries to NT, there were suddenly many thousands of applications. Unfortunately, there were also many, many millions of lines of old code, not written with security in mind.

We still have patch Tuesday, as well as security excitement for all operating systems. None of this will be going away, as the number of programmers capable of writing mostly secure programs is extremely limited. At a past security symposium, a speaker suggested during a WiP that there were only two such programmers, Wietse Venema and Daniel Bernstein. I think this is an exaggeration, leaving out other outstanding programmers. But the point is that most programmers are not particularly good, and certainly not good at security.

At the same time, people are encouraged to write programs. Microsoft's very success is tied to its vast number of applications. But so is Microsoft's greatest weakness: maintaining backward compatibility so that it doesn't lose this asset. This is a problem for all systems today, as adding software—say, a cool PHP-powered Web site—to a server is easy. None of this is news.

Is Windows 7 safe to use? It is safer, but not safe. For example, the ZeuS botnet has been in the news as I write [4], and this criminal tool includes exploits for Vista and 7 [5].

Being safe on the Web today is still difficult. I suggest booting Linux (or a BSD) from a CD or write-protected USB stick and using this for your must-be-secure browsing, such as banking. Next best is to avoid the most popular platforms, such as Windows and Mac, and stick with something obscure like FreeBSD (if you can get the financial site to work with Firefox). Note that malware is starting to appear on smartphones, so banking online using your handheld device may not be safe either. Am I paranoid? Yes, I am paranoid, especially when I recently learned that a security friend lost $35,000 from his bank account.

If you are running a server, consider using tools such as SELinux to sandbox the server's applications. While type-enforcement mechanisms will not protect a server from bugs within itself, such as SQL injection, it will prevent exploits from escaping the application in most instances. Then again, Linux kernel exploits may be designed to bypass, or even abuse, SELinux in the exploit [6].

A Better Sandbox

Robert Watson, a key contributor to the FreeBSD kernel and, by association, to Mac OS X, has created a different way of sandboxing applications. Working with a colleague at the University of Cambridge and two people at Google, Watson developed Capsicum, a capability-based sandbox.

I found a couple of things interesting about Capsicum. First, it attempts to make life simpler for the programmer. Its basic principle involves severely limiting access to the operating system's namespace: files, IPC, shared memory, and even network access. The capabilities used in Capsicum are, for example, open files. Once an application enables Capsicum, only already open files, or files within an already open directory, can be accessed. Capsicum can also work in programs that split privilege levels, such as OpenSSH sshd. The privileged part of a Capsicum-enabled program maintains access to the system namespace and can share capabilities, such as open files or network connections, with the constrained fork of the program.

Capsicum also places the security policy for an application within the application itself. Using SELinux or Microsoft's Integrity Levels and ACLs means that a large portion of an application's security policy exists in system configuration—for example, in Type Enforcement and File Context rules in SELinux. With Capsicum, upgrading a program's security policy is done by upgrading the program, without needing to change system security policy.

The Lineup

Robert Watson, Jon Anderson, Ben Laurie, and Kris Kennaway start off this issue by explaining Capsicum in more detail. Their article compares Capsicum with other forms of sandboxing, as used in Chromium, as well as providing an example of securing tcpdump by dropping privileges after they are no longer needed.

Next up, Moheeb Rajab, working with a team of Google security engineers, updates us on a trend in malware. I've asked Niels Provos to write about his team's activities in the past, and this time Rajab explains that their data clearly shows an increase in the amount of exploits designed to trick people into installing fake antiviral software. I know people who have been fooled by this; I convinced one of them to install Linux instead of reinstalling Windows and buying AV.

Dan Geer, who as Invited Talks co-chair at USENIX Security help to serve up an excellent list of speakers, reprises his own invited talk. Geer ponders the problem created by standards, especially when the protocols they describe are so complex that everyone ports the reference implementation. The result is a form of monoculture, where most, or even all, systems include the same bugs. We have seen this most recently in TLS renegotiation, a protocol that appears in embedded systems as a security feature.

I saved the hardware security article for last, as some may find it a deep dive. Matthew Hicks and co-authors write about their design for working around hardware that includes suspicious circuits. In an earlier paper [7], King et al. showed how they could add circuits to a SPARC CPU that provided a foothold in a system that could easily lead to complete compromise. In this article, Hicks et al. explain how to detect potentially malicious circuits and provide workarounds in software, allowing a system found to include malicious circuits to be patched in the field. I do recommend that you read this article if you wish to learn more about the problem of hardware that may include malicious designs, and a possible solution.

David Blank-Edelman takes us on a utilitarian journey of modules that provide, well, utilities. How practical, and useful as ever.

Peter Galvin begins to explore alternatives to Solaris in the first of a two-part column. In this issue, Galvin compares and contrasts what he considers the most likely contenders to Solaris in enterprise-level computing: Linux and AIX. Yes, I wrote AIX, and don't write off this unusual UNIX variant without a closer look.

Dave Josephsen explores Ganglia, a tool for monitoring clusters of systems. Josephsen obviously likes Ganglia (enough), partially for the ease of configuring clients and for its lightweight footprint.

Robert Ferrell ponders the intent of a hardware manufacturer who is selling CPUs with key features disabled—but will enable them if you are willing to pay a ransom.

Elizabeth Zwicky gets into the Christmas spirit, including reviews of two cooking books and a LEGO book, as well as two technical books. I had also read *Cooking for Geeks* and would have called it *Cooking for Hackers*, as it is full of the type of details I wanted to find out years ago. I also wrote a review on a book you may consider buying as a gift for someone, *Your Money: The Missing Manual*. Sam Stover reviewed our only security book this time, *Inside Cyber Warfare*, and it sounds like an interesting and quick read.

We have reports on the 2010 USENIX Security Symposium, as well as reports for three of the seven workshops that were co-located with Security. We also have a summary of NSPW, an interesting security workshop with very limited attendance.

I am not really worried about depressing you when it comes to news about security. If you aren't depressed, something is wrong with you, or you just haven't been paying attention.

Stuxnet, a bit of very competently designed malware aimed specifically at Siemens S7 control systems used in Iran, has been in the news as I muse [8]. Stuxnet, spread via USB sticks, includes *four* Windows zero-day exploits and two signed device drivers, using keys stolen from two companies in Taiwan. The malware is carefully written, so that it never crashes the systems it infects, never communicates with its creators, and only causes havoc when it detects it is running on the S7 systems installed in very specific applications.

In other words, Stuxnet appears to be the first shot in a "cyber war"—a term I hate, but I don't know what else fits. And now that the cat is out of the bag, I expect we will begin to see copycat attacks take down other SCADA-controlled systems, with the developed world, particularly the United States, being particularly vulnerable.

When computer systems were first used, they were terribly expensive and carefully isolated systems. As this changed in the 1980s, people were just

happy to have computers they could afford. In the 1990s, prices of systems began to plummet, with the first under-$1000 system appearing around 2000. Now you can buy a netbook for under $400 and smartphones more powerful than a 1980 Cray. None of this history includes a mandate for secure computing.

Building secure computer systems requires a complete redesign of both software and hardware, and this isn't going to happen overnight. I do see some things I like, such as SeL4, type-safe languages, and experimental multicore designs such as the Single Chip Cloud. But restarting computer science, where security is built in and unavoidable, instead of an added-later feature, is still years away.

REFERENCES

[1] Integrity Levels, used in Vista and Win2008 and later: https://secure.wikimedia.org/wikipedia/en/wiki/Mandatory_Integrity_Control.

[2] Roger Johnston, Editor's Notes, *Journal of Physical Security*, vol. 4, no. 2 (2010): http://jps.anl.gov/v4iss2.shtml.

[3] Crispin Cowan, "Windows 7 Security from a UNIX Perspective": http://www.usenix.org/events/sec10/stream/cowan/index.html.

[4] ZeuS bot installed using LinkedIN spam: http://www.thestar.com/news/gta/crime/article/869704--email-attack-targeting-linkedin-users-termed-largest-ever.

[5] ZeuS for Windows 7: http://www.secureworks.com/research/threats/zeus/?threat=zeus.

[6] Cheddar Bay Linux kernel exploit: http://lwn.net/Articles/341773/.

[7] Samuel T. King et al., "Designing and Implementing Malicious Hardware," Proceedings of the First USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '08), April 2008.

[8] Stuxnet May Be Targeting Iran's Nuclear Sites: http://www.bloomberg.com/news/2010-09-24/stuxnet-computer-worm-may-be-aimed-at-iran-nuclear-sites-researcher-says.html.