RIK FARROW

musings

rik@usenix.org

I RECENTLY READ A SUMMARY OF new vulnerabilities in 2006 which stated that, for the first time, buffer overflows had fallen from number one to number four. The new "leaders" in the vulnerability and exploit race were cross-site scripting (XSS), SQL-injection, and PHP file inclusion. Many pundits drew from this the conclusion that buffer overflow vulnerabilities are in decline. If only that were true, perhaps we would be seeing some light at the end of this tunnel.

Developers, security companies, and vendors have done considerable work to retire buffer overflows as security issues. The early work includes scripts that simply scan for offending string manipulation functions, such as strcpy() in source code. Replacing strcpy() with strncpy() adds bounds checking when copying a string, and as long as the length of the destination string is used as the bounding value, this actually helps a lot. In the security business, we call this "low-hanging fruit," as finding and fixing these vulnerabilities is as simple as running a tool.

Other pathways to uncovering buffer overflows include tracing input flows. An attacker manipulates buffer overflows by providing some malicious input, so the defending coder needs to follow the flow and see how any input gets used. I don't want to make this sound easy, because it's not. Take the Sendmail vulnerability in the address checking function that missed decrementing a counter while parsing email addresses. That bug had been there for many years, unnoticed—until it was found in 2003. A second, similar bug was found later in 2003, also in code that checks email addresses.

Code reviews that can find bugs like these are tedious. After the first bug was found, I downloaded the patches for Sendmail, then used them to find the buggy routine (although there were multiple functions patched, looking for ones related to user input helped to shorten the search). Then I walked through the crackaddr() function in the header.c file, certain the mistake would be found there. The error was there, I guessed correctly how it worked, but I didn't actually uncover the crucial place in this function where a variable should have been decremented. It just isn't that easy.

A hacking group didn't find the missing decrement either. They debugged vulnerable versions of Sendmail and found a way to overwrite a FILE object so that a read callback would point to their own, handcrafted code. This code makes an outgoing connection to port 25/tcp at the IP address of the attacker's choosing [1] and runs a root-owned shell.

Invoking the Hardware Mantra

Since code reviews are so difficult, coming up with a solution that bypasses code reviews is imperative. And there are many such solutions, some done in software, and better ones manifested through appropriate use of hardware.

Crispin Cowan et al. [2] created the idea of placing a value in memory that would be overwritten during a buffer overflow, and testing that value works as a check for buffer overflows. This idea, known as the "stack canary," forms the basis for many software protection schemes, including one used by Microsoft. If this worked reliably, there would no longer be exploitable buffer overflows in Microsoft code, as evidenced by the emergency patch released on September 26 to fix a buffer overflow in the Vector Markup Language (vgx.dll) code in IE versions 5 through 7 [3].

There are, of course, other techniques that can be used to thwart buffer overflows. RedHat, in its versions of the Linux kernel, uses several software techniques, most notably by changing the layout of memory during process creation. By changing the location of the stack within an 8K window, getting an exploit to work correctly becomes much more difficult (but could succeed eventually).

Sun Microsystems has long included software support in its kernel for hardware protection. Sun's SPARC chip architecture makes it easy to make the stack nonexecutable, and this has been an option since the mid-1990s in Solaris. Other prominent CPU vendors lagged way behind on this feature. Intel processors, until more recent versions, could have nonexecutable stacks only by using a kludge that involved segment registers. New AMD and Intel processors now make it much simpler to enable hardware stack protection by no longer lumping write and execution permission bits in memory management together.

Kiss of Death

Even if programmers and chip designers had solved stack-based buffer overflows, we still have to deal with other related programmer errors, including format string, integer overflow, and double free bugs. None of these ever attained the number 1 status of buffer overflows, but all make it into the top 40 vulnerabilities [4]. Protecting the stack does make exploitation more difficult, but not impossible, as people are still writing both buffer overflow and other bug-related exploits.

When I posted some of my thoughts about buffer overflows being toppled from number 1 by Web scripting bugs, I got an even more thoughtful response from Chris Wysopal. Chris pointed out that a buffer overflow that gets prevented by *any* of the methods I've mentioned here has been converted into a Denial of Service attack. All the mechanisms designed to defend against buffer overflow to date halt the execution of the offending program. With the exception of multiply threaded servers, such as Apache, or ones that watch for untimely server death, such as Postfix, the server

has died, denying service. Remember, if your Web browser mysteriously dies while following a link, it may be a victim of an "unsuccessful" buffer overflow attack.

"We stopped the buffer overflow attack, but the patient died." How sad.

Just as significant, Chris pointed out that buffer overflow attacks *have not declined statistically*. To me, this is the most damning point of all. Buffer overflows have fallen to number 4 in the top 40 CVE vulnerability list only because Web scripting bugs have increased in popularity. The absolute number of buffer overflows has remained almost constant over the past five years.

Web Services

Web scripting/programming errors now make up four of the top five reported vulnerabilities. The "dot" category stands for vulnerabilities that rely on the use of ".." to view or execute files that should be protected. I would have thought that, like SQL-injection and XSS, these attacks should have fallen to careful inspection of client-supplied input. Sadly, I am mistaken. People continue to make the same mistakes year after year.

I don't want to trivialize this problem. XSS, in particular, is difficult to deal with, although scrubbing <script> from user input to Web scripts would go a long way toward curing this issue.

What really struck me was the "new" PHP bug, file inclusion, involving simply appending a bit of text to a request to a PHP script. PHP has a reputation for making it easy to write insecure Web scripts, and this flaw certainly bolsters that impression.

The Lineup

Team CYMRU leads off the 2006 Security focus issue with an article about the computer underground. Rob Thomas gave the keynote at SRUTI [5] on this very topic. If you have ever worried about sharing your personal information online, or with anyone, whether he or she is a bank clerk or just some anonymous person on the phone, you will want to read this article. It's not just the level of fraud but, as the authors write, the lack of any attempt to hide criminal activities that is just astounding. No wonder we haven't caught bin Laden by this time if the United States and other countries continue to permit blatant identity trading to go on (not that identity traders are never caught; see [6]).

Next, a trio of German researchers share information about how they detect and quarantine infected Windows systems on a university network. Through the use of Nepenthe, a low-interaction honeypot, they can both capture malware and detect without any false positives infected systems on their network. I know that this is an issue for many organizations, and the approach described here appears worth pursuing. Also, Göbel, Hektor, and Holz describe the Haxdoor malware and the vast cache of identity information it captured in just nine days.

I so enjoyed Andy Ozment's Security '06 presentation that I asked him to write for *;login:*. Andy and Stuart Schechter have statistically analyzed the bugs found in OpenBSD, relating each bug not just to the code patched but to when that code appears in OpenBSD. Perhaps there is some light at the end of this security tunnel after all.

Nick Weaver and Dan Ellis carefully explain why white worms, those that attempt good works rather than exploits, are not a good idea. If you have ever considered writing a white worm, this article is certain to dissuade you.

Mike Scher, past *;login:* contributor and winner of the "legal counsel of this issue" award, entertains us with musings about tort law and negligence. A recent case grabbed Mike's attention, and he uses a similar, but fictitious, case to describe just how an organization might be liable for negligence. If you write policy or have anything to do with administering public servers, you need to read this article.

Finally, Mike Howard argues that changing passwords too often can be more harmful than simply using strong passwords. I believe Mike makes his case well.

In the Sysadmin section, Mark Burgess continues his excellent series about the nature of configuration management. After reading (and editing) Anderson's "Configuration Management" [7], I thought I knew it all. But Mark has a different way of viewing things and a very convincing way of getting the reader to look at configuration management issues in a new light.

Dave Josephsen's article borders on security, but it really does belong in the Sysadmin section. Dave got me interested when he told me about BGP hijacking attacks designed to support spammers. Stopping spam is Dave's real focus, but the BGP attack is interesting in itself. Dave narrates the story of the spam wars, reaching a conclusion you might recognize if you have read Dave's writing before.

Our columnists have done their work as well, with David Blank-Edelman deciding to write about security in the same sense that TSA protects those flying in U.S. airspace. Robert Haskins writes about the use of wireless by ISPs, including solutions that may apply to those who have chosen to live remotely. Heison Chak considers the security-related aspects of using VoIP. Robert Ferrell enlightens us on the uses of sledgehammers in computer security.

In the Book Reviews section, Elizabeth Zwicky continues her search for good Windows security books, finding two, and tells us about reading *Security and Usability*, an interesting collection of papers that came out last year. Sam Stover reports on yet another Syngress book that contains some new material and loads of repeated chapters. Finally, I managed to write a couple of reviews myself.

Nick Stoughton has produced another article in his series about standards. Nick has been the USENIX representative for standards for many years, and these articles allow you to get an insider's view of standards processes (some of which will affect you).

We have three summaries in this issue, all related to security. The Security '06 conference summaries belong in this issue, of course. Dan Geer has produced excerpts from a much longer summary of MetriCon, the first workshop on security metrics. And the organizers of the New Security Paradigms Workshop have produced a very concise summary of hours of discussion.

I would like to leave you with some parting thoughts related to the CYMRU article about stolen identity. During my Guru presentation in Boston at USENIX Annual Tech, I pondered aloud about how it is that in this supposedly modern age our identity can be described in approximately 160 bytes of information. This small amount of data covers everything you would need to present to get a car or home loan—and every-

thing an identity thief would need as well. I think this is absurd, but I will confess that I haven't come up with (and patented) a workable solution.

In the meantime, I suggest that you watch your own credit and banking reports closely. Someone else's jackpot could easily be your own misfortune.

REFERENCES

[1] "Technical Analysis of the Remote Sendmail Vulnerability": http://lwn.net/Articles/24292/.

[2] "StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks": http://www.usenix.org/publications/library/proceedings/sec98/cowan.html.

[3] "Microsoft Internet Explorer VML Buffer Overflow": http://www.kb.cert.org/vuls/id/416092.

[4] "Vulnerability Type Distribution in CVE," posting about the CVE top 38 (not 40, but that sounds better) vulnerabilities of 2006: http://www.attrition.org/pipermail/vim/2006-September/001032.html.

[5] 2nd Workshop on Steps on Reducing Unwanted Traffic on the Internet: http://www.usenix.org/events/sruti06/.

[6] "Hacker Hunters," *Business Week* (May 30, 2005): http://www.businessweek.com/print/magazine/content/05_22/b3935001_mz001.htm?chan=tc.

[7] P. Anderson, "System Configuration": http://www.sage.org/pubs/14_sysconfig/.