musings

by Rik Farrow

Rik Farrow provides UNIX and Internet security consulting and training. He is the author of UNIX System Security and System Administrator's Guide to System V.

rik@spirit.com

Many years ago, I heard Michelle Crabb, a sysadmin at NASA Ames in the San Francisco Bay Area at that time, tell a story about discovering an intrusion there. Ames was the site of the satellite feed that connected the nascent Internet in Australia to the continental US. A NASA manager wanted Internet access added to his own desktop Sun workstation, and eventually Crabb agreed to do so. Within a week, the manager came to her and mentioned that the login prompt on his Sun had changed. Instead of "login" with a small "l", it now appeared as "Login".

This subtle change was the overt signature of a trojaned login program. Had the Australian teenage attacker been a bit more careful, his successful intrusion might have gone unnoticed much longer.

I have always wondered about how many really skilled attackers there are. People who can break into a system, cleverly trojan/rootkit it, and only visit that system occasionally. With little traffic and no overt signs of an intrusion, such as the usual port scanners or IRC relays like psyBNC generating traffic, a skillfully completed intrusion might not be noticed for months, perhaps years.

Little, if any, data about the really skillful attacks seems to exist. But a very big hint did emerge in late May, when new overflow exploits, as root, for CVS server software were divulged. Some people claimed that they had been using the CVS exploits since 2001 and had owned many well-known open source CVS servers. All that was needed was anonymous read-only access in order to exploit the server.

I found these assertions somewhat hard to believe. Partially because I didn't want to believe them. And largely because so few attackers (apparently) have the discipline to take over a system and behave in such a low-key manner as not to be noticed. Also, one would hope that any changes to open source software, such as the addition of back doors, would have been noticed by now (after the claimed three years).

There were the trojaned configure scripts that were discovered in 2003. Each configure script included an addition designed to look as though it belonged in a configure script. The trojan was not rocket science, as it used a hardwired address to connect to and failed to loop properly (at least the version I played with). The trojaned configure script would hardly be the work of a skillful attacker.

The attacks on supercomputers at Stanford University and in the TeraGrid at SDSC, NCSA, and other locations were closer to a skillful assault, but they were not actually successful. The attackers, instead of treading lightly, just kept abusing more accounts and taking over more systems, making it only a matter of time before their intrusions were discovered. The attackers were certainly persistent, returning even after being discovered. I am hoping that one or more of the defenders will discuss their experiences with these intrusions in the Security edition of *;login:*.

Open Source in Danger?

If open source (OS) CVS servers were owned for a long time, what effect will this have on the security of open source software? There have been attempts to install back doors in OS software before, such as the slight modification to the wait() system call late last year. But that attempt was noticed because the attackers bypassed the code

management system, drawing attention to the change. Two other attempts, on CVS servers, were also noticed.

One reason for these changes being noticed is that any time someone submits a change, many eyes will scrutinize the code. It would be one thing to make a change that would subvert the entire CVS system, and quite another to slip something strange past people who are competing (to a degree) to produce the best code. Still, the reputation of OS code is at stake, and weaknesses in CVS code, responsible for maintaining the integrity of OS, certainly do not look good.

Other Dangers

Problems with CVS have a much greater impact, I would hope, than the latest bit of tomfoolery from the Alexis de Tocqueville Institution (ADTI). Ken Brown, president of ADTI, paid a visit to Andy Tanenbaum at Vrije University in Amsterdam, where he teaches computer science. Tanenbaum is the author of MINIX, a UNIX-like operating system created mainly as a teaching tool. Brown had traveled from Washington, DC, to the Netherlands, apparently convinced that there was some ill will between Linus Torvalds, creator of Linux, and Professor Tanenbaum, creator of the not nearly as popular MINIX. You can read Tanenbaum's words for yourself at *http://www.cs.vu.nl/~ast/brown/*.

Brown has written a book which may have appeared by the time this column gets published. His book will apparently make the claim that Linus did *not* write Linux, because it is impossible for one person to write an operating system. Imagine making that claim after interviewing a college professor who had himself done exactly that.

There are more incredibly bogus claims floating about, such as the one by Dan O'Dowd, CEO of Green Hills Software, who said in a speech, "The very nature of the open source process should rule Linux out of defense applications." O'Dowd used as his example Ken Thompson's famous trojan code in early versions of AT&T UNIX. How Thompson's wonderful hack would affect an entirely different code tree (it wasn't actually a *magical* hack) is beyond most thinking individuals. Perhaps O'Dowd was trying to recapture Linux business his company had lost to OS versions of Linux?

The closed source world is just as vulnerable to back doors, if not more so, than OS. Some security software vendors have had difficulty staying on NSA's approved software list because when NSA attempted to build their software from the source, it didn't match the distributed version. Did this signify back doors or merely different compiler flags and configuration options? And Thompson's back door was in closed source, not OS.

Cash Registers and Voting

The most egregious example of closed source software appears in US voting systems. Electronic voting has been successfully used in Australia and India. The source code is available for public inspection, not kept as a trade secret as is the code in the most widely used electronic voting systems in the US. (See *http://www.elections.act.gov.au/Elecvote.html* for information about an Australian system, *http://www.theregister.co.uk/2004/06/23/open_source_voting_software/* for information about a Dutch system.)

What gets me about companies like Diebold is this: Diebold also makes touch-screen cash registers, like the ones you have probably seen in bars and restaurants. Similar touch-screen cash registers can be purchased from Internet sites for about $1200, with

printers included. Diebold's touch-screen voting systems cost over $3000, without printers. And Diebold has a well-known reputation for installing last-minute patches to their voting systems, a practice that resulted in Diebold machines being barred from use in several counties in California.

Now what is so hard about a touch-screen voting machine that it would require last-minute patches? Diebold doesn't have this problem with their cash registers. Cash registers have a paper trail to prevent fraud. But not voting systems? Something is fishy here—actually, it stinks to high heaven.

I think the problem with some US voting machine companies is so serious that people prefer not to think about it.

Open source, or at least publicly audited code, is more secure than proprietary code. If you have to use a touch-screen system to vote, vote by mail so that there will be at least as much of a paper record as there would be when you buy lunch.