;login:

THE MAGAZINE OF USENIX & SAGE

June 2001 • Volume 26 • Number 3

inside:

SECURITY
Musings

by Rik Farrow

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

musings

It is truly amazing how busy one can get.

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>

By the time you get this issue of *;login:*, it will be the beginning of summer. Summer often means that the pace of work slows down, as students go home or coworkers take vacations. And the slowdown in the economy is already helping some people. Although most of the layoffs (firings, in a less gilded age) are in factories, some are in the tech sector. And others who have survived the dot-com craze are now working at a much more reasonable pace.

I was fortunate enough to meet some of the movers and shakers for a two-day meeting that finished two days before I wrote this. You might not consider the Linux Developers conference a relaxing event, and you would be right, but it was certainly stimulating. I had often wondered about the people who write and guide the creation of the Linux kernel. You can find out more about this event, and something about the personalities involved, by reading my summaries in this issue.

I also discovered just the thing to while away the idle hours I don't have, and perhaps you have already heard of it. The Honeynet Project (<*http://project.honeynet.org/challenge/*>) is a collection of security guys who have decided to focus on attack signatures and forensics. They had posted a forensics challenge, and one of their number, Dave Dittrich of the University of Washington, let me know about it in January. I was too buried to even think about it then, but by the middle of February things had calmed down enough for me to take a peek at the challenge, and when I did, I started to get really excited. Well, you really have to be interested in security like I am to get turned on by something like this.

As one of their projects, members of the group monitor networks looking for attack signatures, using the free ID software snort. On the night of November 7, snort detected a scan and then an attack against a Linux system. The victim, a RedHat 6.2 server install, had only been up 2 1/2 days before the successful attack. One of the project members monitored the system, and once things had settled down, took it offline and made a full-image copy (using dd) of the entire hard disk. Good thing it was only a three gigabyte drive.

Conversion

The challenge was to determine how the system had been attacked, which tool was used, who did it, as well as what the attacker did after the attack. As a hint, snort logs from the initial attack were provided, as were the hard disk images. The hard disk had been carefully partitioned, each partition saved and gzipped, along with information about how to mount the partitions using the Linux loopback mount interface. The fact that the victim system was partitioned made the analysis much easier.

I started by mounting the partitions and looking around much like a system administrator might – especially one who had the benefit of an ID system. Well, this was a pretty good waste of time, although not totally fruitless. I started by looking for a service that listened to port 871/udp. I will tell you right away that RFC1700 does not list any service at that port. But, I knew that the attack followed a scan of rpcinfo at port 111. So I started scanning system startup files, looking to see what would have been started. Only the NFS lock services, statd and lockd, would have been started, and statd has a long history of exploits.

I also noticed that the startup script for syslogd had been deleted. The only ordinary user account was named drosen, and in that directory, I found a .bash_history file with a list of commands in it. That sent me off to /usr/man/.Ci, a "hidden" directory just chock

full of stuff. There were scripts here, "hidden" files, as well as directories with scanning and exploit tools.

It was at this point that I decided to download and install The Coroner's Toolkit (TCT) (<*http://www.porcupine.org/forensics/*>) and take advantage of the tools written by Dan Farmer and Wietse Venema. During the challenge, Brian Carrier of Purdue wrote some additional tools to supplement TCT (<*http://www.cerias.purdue.edu/homes/carrier/forensics.html*>), and it would have been handy to have some of these: for example, one that listed the contents of a directory inode (even if it had been deleted) and showed any filenames corresponding to deleted files. I was accustomed to using od -cx on the directory itself, but the Linux kernel balked at letting me read directories as files.

Before I make things look absurdly difficult, the top-rated investigator in the challenge started off with a much different approach. Thomas Roessler mounted the partitions and then used RedHat's rpm tool to determine, in conjunction with MD5 checksums and the still-installed package file, which files had been modified after the install. Another investigator used Tripwire checksums for a database created on an intact system. Both turned up a list of Trojans that had been installed: ls, ps, top, netstat, tcpd, ifconfig, and in.identd. Most of these are part of standard rootkits (that is, similar to one I was given back in 1994). The fake identd provides a root shell to anyone connecting from one of two "magic" source ports.

Autopsy

The standard RedHat 6.2 server install includes 30,000 files and directories, and we know that the attacker has at least attempted to hide things. Also, it is likely that some of the evidence of the attack was deleted. This is where the TCT comes into play. By running graverobber, you create the body file which is grist for the mactime script, which I'll discuss in a moment. What some of the other investigators did (and I wish I had too) was to use the unrm tool to search for deleted inodes, and then to convert the output of ils (inode ls) into a format suitable for inclusion in the body file. Then, when they run mactime, they can see not only the existing files and directories, but ones that have been deleted (and not yet reused) as well.

mactime is my favorite tool of the bunch. It presents a list sorted by access, modify, and/or inode change time. For example, when a program is executed, you see that its access time is modified, as well as the access times of any libraries loaded at that time. Or, when a file is modified, you can see when that happened. Of course, you can only see the last time a program was executed or a file modified. But, you can see things like a script named clean being executed by the attacker, which in turn called the snap script, which then strips lines that match certain patterns from logfiles.

Roughly speaking, here is what the attacker did. I will use the datestamps taken from the victim system, whose clock was set incorrectly (just to make things more interesting, no doubt).

The initial attack followed a couple of probes. First port 111/tcp was checked to see if it was open and reachable, then an RPC request was made, most likely checking for statd and the port it was listening to (871/udp). Forty-five seconds later, the attack used statd to write a new entry in /etc/inetd.conf that would run a root-owned shell at port 4545/tcp for anyone who connected to it. This part of the attack appeared to be automatic, which was very likely, considering some of the other stuff found on the victim. This happened shortly after 11 p.m.

At about 7:25 the next morning, the attacker connected to the magic port, and cleared /etc/host.deny, a control file for TCP wrappers. It was probably at this point that the attacker deleted the init script that starts syslogd, leaving many orphaned symlinks pointing nowhere. The attacker then used FTP to download some tools and scripts. One of these tools was used to add two new user accounts, own and adm1, and the attacker then disconnected and logged back in as adm1. The own account had a UID of zero and no password, so the attacker could immediately become root. Soon, the attacker cleaned up the inetd.conf file so no one else could take advantage of the original backdoor at port 4545.

Although the attacker deleted or cleaned up many logfiles, lastlog was forgotten. Linux has no tools that will display the content of any lastlogfile except one in the default location, so I wrote one (something that Roessler did as well) and discovered that the attacker had logged in at 7:31 as adm1 from c871553-b.jffsn1.mo.home.com (the initial attack came from an address within home.net).

The very next activity was to install tpack, an IRC bot used to keep control over IRC channels. The attacker later installed a recent version of BitchX (bx), an IRC client. This indicates a possible motive for the attack – to gain another platform for the IRC wars.

The attacker next started running scripts that backup programs, then install and configure the various Trojans mentioned earlier. Most of these Trojans are part of the lurker four (lrk4) package that you can find at sites like <*ftp://ftp.technotronic.com*>, although a couple are apparently from a later version, lrk5. The scripts are not perfect: for example, they use the wrong pathnames for a replacement telnetd and a Trojanned syslogd (that ignore log messages that match certain patterns).

The attacker's script also launches snif, a version of LinSniffer (also part of lrk4), leaving behind snif.pid and an empty logfile (no passwords were sniffed).

Good Sysadmin

I found what the attacker did next remarkable, although I have heard that this has become increasingly common. The attacker downloaded a set of rpms and patched the most commonly used security holes on Linux systems: amd, lpr, nfs-utils, wu-ftpd, and BIND. Another script ruthlessly stripped away set-user-id permissions on many programs, apparently to prevent anyone else from using an exploit to gain root access.

The reason for this paranoia became more apparent when I looked through the /usr/man/.Ci directory. I found scanning and exploit tools for exploiting each of the patched server programs, as well as z0ne and strobe for scanning. In general, each directory contained a program that generated IP addresses when given a 16-bit prefix (e.g., 206.1 to create 206.1.1.1, 206.1.1.2, up to 206.1.254.254), that got passed to a tool that probed for a particular port, BIND version, or RPC service, and finally to an automatic exploit. In other words, the system now became yet another automatic attack platform that, in turn, discovers more vulnerable systems. No wonder it took only 2 1/2 days before this system was exploited.

Like any security-conscious individual, the attacker installed a version of SSH. Unlike most versions, this one not only logs passwords, but it also has a magic password. The attacker logged out of the adm1 account, and logged back in using SSH and the magic password. The Trojan sshd dutifully logged the magic password, tw1Lightz0ne, in the logfile (/usr/tmp/nap).

By the time the attacker logged out, he had spent about 36 minutes downloading files, installing rpms, and running scripts. The adm1 and own accounts were deleted, and Trojans hid most (but not everything) that had been done. The attacker had two backdoors, sshd and identd, to use for future visits.

What the attacker took about a half hour to accomplish takes the average investigator about 34 hours to uncover. I spent close to 30 hours myself (I did not enter the challenge), simply because I did feel the challenge posed by this thoroughly hacked system.

In the usual case, this system would simply have been taken offline, and the OS reinstalled. If security patches had not also been installed, it is very likely that the system would have been hacked again in very short order. At the time I write this, the port being scanned most frequently is 53/tcp, the port used by BIND. If you have systems, even non-Linux ones, that use versions of BIND prior to 8.3 or 9.1, I suggest that you upgrade them soon.

The attack used by the Honeynet Project was not particularly subtle. Swift, a little messy, obviously not someone who was totally clueless, but (by the same token) someone who obviously was into taking over as many systems as possible. Not what I consider a serious attacker, who would have been considerably more subtle and left much less in the way of traces. If all the attacker had wanted was a copy of a certain file, the initial attack could have used a script that would have done everything, including cleaning up afterwards.

All in all, trying The Challenge was a very worthwhile learning experience. I suggest it as a fine way to pass the copious free time that you might have. On a more serious note, if you do take the time to read a couple of the analyses, you will learn a lot about how one would investigate a hacked system. You might also consider scanning your own networks (you can use the tools found in The Challenge) and see how many of your own systems are vulnerable.

And if you don't find any at all, that could be good or it could be bad. If you installed the patches, it is good. If the attacker installed the patches, well, you have a lot of work ahead of you.