Examining the art of computer forensics

TO CATCH A THIEF

You don't need expensive proprietary tools to practice the craft of computer forensics.

BY NILS MAGNUS, ACHIM LEITNER, AND JOE CASAD

Crime scene: the server room… The thief doesn't need a key card or the protection of darkness – an intruder can use the Internet to come and go. But despite the secret entrance, the attacker still leaves behind some telltale traces. Finding and interpreting this evidence is the top priority of criminal investigators.

This month's cover story explores the world of computer forensics. We'll show you some tools the experts use to find clues, recover deleted files, and root out hidden evidence. We start with a study of the open source Sleuth Kit forensics toolkit. We also look at Foremost and Scalpel – a pair of tools for finding and restoring deleted files. We show you how to examine Windows disks with Linux tools, and we end with a look at the Open Computer Forensics Architecture, a freely available collection of forensics tools and libraries developed for the Dutch police.

But if you're not really going to trial and you just want to catch the intruder on your system, you might not want to go to all the trouble of launching a full forensics investigation. The following sections describe some tips for finding intruders on the system using standard Linux utilities.

In Your Face, or Clandestine?

One of the first questions a forensics investigator must ask is whether the investigation should be performed openly – which means that it will be visible to the attacker, too – or the attacker should be kept unaware of the investigation.

A computer under forensic investigation is very similar to a particle in quantum mechanics: just looking at it changes the state.

An attacker might see a *ps* command, and running *find* against the hard disk typically overwrites valuable *atime* records on the filesystem, removing evidence of a user's last access.

Despite the possible complications of working in the open, the need to get to the bottom of illicit activities is sometimes more important than taking elaborate steps to avoid notice.

Also, keep in mind that most attacks are launched through automated scripts and programs, thus, it is unusual to catch an attacker red-handed at the console. The following tips are primarily for cases in which you don't really care about concealing your activities or keeping a paper trail.

To avoid overlooking details, a systematic approach is useful. The idea of following a

hot trail is often very seductive, but if it takes you to a dead end, you will be disappointed.

For example, if you investigate a list of processes using the command

```
ps gauxwww
```

you should store and work through the full list. The command shows all of the active processes and their command-line arguments, including full options.

Of course, if your system has been compromised, it is always possible an intruder has installed trojaned versions of system utilities, such as *ps*, to conceal the break-in.

A small shell script does the same job by reading the */proc* data (see Listing 1).

Individual extensions are easily added to a script like this and can be particularly useful if you can't trust *ps*.

For a good sanity check, you need to crosscheck the results using a tool similar to the popular *pstree*. Forensic investigators will also remember that programs can change *argv[]*, their argument list, programmatically (see Listing 2).

Kernel

A simple trick like searching for processes is powerless against a kernel rootkit. Rootkits modify the kernel to prevent it from delivering information about certain processes to the */proc* filesystem or other information mechanisms.

On the other hand, it is very surprising how little trouble some attackers take to cover their tracks, so it might be worth trying.

Network Connections

Besides processes, network connections can also reveal clues, such as the attack vector and the address the attacker used to connect to the system.

Issuing *netstat --ip -pan* on Linux shows you all the local IP sockets, their protocols (TCP or UDP), and possibly the communication partners for connected sockets – unless the command or the kernel happen to have been manipulated.

Setting the *-n* option in *netstat* prevents DNS from resolving IP addresses and giving you the matching hostnames. This is a good idea because it avoids unnecessary and suspicious network traffic to the name server.

If necessary, you can always resolve the IP addresses later.

The *whois* and *traceroute* commands display more information about IP addresses. *whois* queries one of several Internet databases to reveal the registration data for the network scope.

Typically, these details are very trustworthy and difficult for attackers to spoof without the cooperation of an Internet service provider.

Listing 1: DIYS Replacement for ps

```
#!/bin/sh

cd /proc
for p in [0-9]*
do
    proc=$(cat $p/cmdline)
    user=$(ls -ld $p | cut -d\  -f3)

    echo "$user $p $proc"
done
```

Connection Origin

One final thing that forensic investigators should not forget is that the origin of a TCP or UDP connection does not necessarily match the attacker's location. Some attackers use hijacked systems as a starting point for their work.

If the connection originates with a system that is very close to your own network, you should be extremely cautious. A short list of hops in the output from *tcpdump target-address* will tell you more. If you can rule out a regular user, you have fairly convincing evidence that

Listing 2: changecommand.c

```
/*
  Build as follows
  gcc -o changecommand changecommand.c
  Waits for three seconds, then changeas its
  command line, waits another three seconds
  and terminates. The specified commands
  are not executed, but they do frighten
  the administrator running the ps command.

*/

void overwrite(char *arg, char *new) {
    char w;

    while (*arg)
    {
        if (*new)
            w = *new++;
        else
            w = 0x00;
        *arg++ = w;
    }
}

int main(int argc, char **argv)
{
    char a0[] = "/bin/rm";
    char a1[] = "-fr";
    char a2[] = "*";

    usleep(3000000);

    overwrite(argv[0], a0);
    overwrite(argv[1], a1);
    overwrite(argv[2], a2);

    usleep(3000000);

    return 0;
}
```

an attacker has made inroads into your network. An attacker who is close is much more dangerous than one who is far away. (Sniffing passwords off the same subnet is much easier than off the Internet.)

Looking for Clues

If the forensics expert has discovered an unknown process or program running on the system, the next question is: "What does the malware actually do?"

Is it just a jumping-off point for more attacks? If so, an unknown process will probably have created an entry in the socket list.

Does it sniff data off the network? If so, look for a network interface in promiscuous mode.

Typically, this creates an audit entry in the kernel ring buffer, and you can run *dmesg* to find out:

```
device eth0 entered ↩
promiscuous mode
audit(1408381411.504:2): ↩
dev=eth0 prom=256 ↩
old_prom=0 auid=4267295
device eth0 left ↩
promiscuous mode
audit(1408381413.144:3): ↩
dev=eth0 prom=0 ↩
old_prom=256 auid=4297295
```

What are you supposed to do if you see an active process but don't know what it does? To start, it makes sense to start by backing up the process itself. To locate the executable, type *ps gauxwww* or check */proc/PID/cmdline*.

What are you supposed to do if an attacker has launched a tool, immediately deleted it, and overwritten the disk sectors? While the program is running, there is hope – the kernel keeps a virtual symlink to the executable in */proc/PID/exe*, even if the attacker has deleted it from the filesystem. If the response team saves this file somewhere safe, analysis is often possible at a later stage.

Poking in the Binary Trash

One simple but effective approach is to take a closer look at the binary itself. The *strings -a binary* command searches a file for printable characters. If the malware connects to an FTP or web server that requires a password, you might be able to find the password in the program code. But you will need a modicum of intuition to distinguish digital bread crumbs from binary trash.

A Last Resort

If you are considering running binutils tools – for example, to extract the symbol table (with < *nm*), or even disassemble the machine code (*objdump* might help) – your mileage will vary. Usually, this technique is a last resort.

Conclusion

The simple strategies we've described might help you catch a thief in the act, but if the intruder is a seasoned professional, or if you need to worry about maintaining a formal, documented process for collecting evidence, you'll need something more.

Read on for more about the tools and techniques of computer forensics. ■

FREE SOFTWARE FOUNDATION

Freedom:

the freedom to run the software as you wish.

Join the FSF as an associate member.
Your donation will support our 20-year-old commitment to community.