The dangers of rootkits

ROOT TREATMENT

Rootkits are part of the cracker's standard repertoire, allowing them to hide their activities and the results. If you find a rootkit on your machine it is high time for some "root treatment". Boris Schauerte explains

Rootkits are popular aic kiddies. Th make an attacke much easier, as t that once a brea been successful easily regain roo within the syster installing "backc They also disguis cracker's presenc that the adminis won't even notic the uninvited gu controlling his o machine. To top rootkits cover up left during the b you won't even taken place.

Many rootkits information abo environment, su machine or inter network traffic with the help of a "sniffer". This knowledge makes it easier for intruders to spread their attentions to neighbouring machines.

Rootkits most commonly take the form of Trojan horses. These are patched system programs that behave according to the cracker's wishes. However, there are also rootkits whose main part is a kernel module – these don't even require any host programs.

Trust no-one

Rootkits with patched programs work on the assumption that the superuser is going to trust the output of these programs. This assumption is normally correct, as administrators generally have no other choice. The rootkit modifies important system tools in such a way that they no longer output any

could betray the tance, the *ps* command does not show certain e *ls* conceals the ne directories and files. this sort of manipulation hard to detect unless erent tools are used for checking. You could, for example, compare the *ls* listing with the output rom a *find* call. owever, this will only be cessful if the intruder 't also replaced *find* a patched version. The situation is slightly or rootkits that have el modules. Such a kit ly of the module in lesigned to remove any . Tasks, such as hiding nd users or inserting with by these modules ns that all programs are affected.

System break-in

Before attackers can install their favourite rootkits they need to acquire root permissions. The associated break-in normally follows a set pattern. When attacking a system directly, the cracker generally tries to collect as much information as possible about his or her target. The cracker will probe for weak points and then exploit them. Also common are indiscriminate searches for victims supported by network scanners, which may also be used to automate the break-in.

Once inside the target machine the intruder eliminates the traces of the break-in. In order to do this the log entries are removed as well as other evidence pointing to them. The majority of this

The author

Boris Schauerte lives in Dortmund and works mainly in the field of data and Internet security. He programs enthusiastically under BSD and Linux systems. At the moment he is particularly working on the design and implementation of Free operating systems.

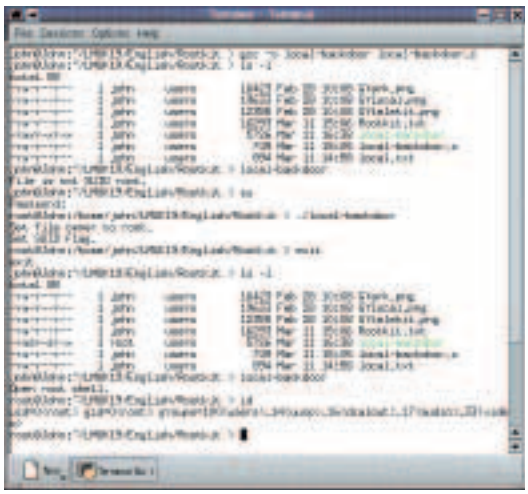

Figure 1: The program in listing 1 assigns SUID root permissions to itself when started by root. After that it can turn any user into root

process is carried out by programs like Zap, which remove all entries in the log files utmp, wtmp, lastlog and messages. Other tools clean additional files in */var/adm* and */var/log*.

The clean-up is usually restricted to the standard files, since it is very time consuming to remove traces manually. This can result in the cracker overlooking some log files, enabling the administrator to detect the break-in after all. A special case is remote logging, where syslog continually transfers its entries to another machine. As long as the attacker has not cracked the log host he will not be able to clean its files.

Once the cracker arrives in the target system, the backdoors provided by the rootkit can be installed, as well as any other programs. The rootkit is often transferred before he covers his tracks as the transfer itself creates new traces. The source of the tools are mostly public servers rather than the attacker's own machine.

Local backdoors

There are two types of backdoors: local backdoors and network backdoors. A local backdoor allows an existing local user account to gain root permissions. In this case the cracker logs into the system as a normal user and executes a program that provides him with a root shell. The backdoor is normally password-protected, so it can't be accessed by any other users.

Suitable host programs for backdoors are login, chsh, passwd or any other program with SUID root permissions. Listing 1 shows how this works (without a host): the user only has to start the program and he immediately becomes root. The file even assigns itself SUID root permissions when it is started by root. The effect can be seen in Figure 1. By the way, this simple example contains a buffer overflow error, which can be ignored for our purposes.

Listing 1: Simple local backdoor

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
  char buf[40];

  /* Is root executing the program? */
  if (getuid () == 0)
  {
    /* Set the file owner to "root" */
    printf ("Set file owner to root.\n");
    sprintf (buf, "chown root %s", argv[0]);
    system (buf);

    /* Set the SUID flag */
    printf ("Set SUID flag.\n");
    sprintf (buf, "chmod +s %s", argv[0]);
    system (buf);
  }
  /* A normal user is executing the program */
  else
  {
    /* Set UID and GID to 0 (root) */
    if ((setuid (0) != 0) || (setgid (0) != 0))
    {
      printf ("File is not SUID root.\n");
      return -1;
    }
    printf ("Open root shell.\n");
    execl ("/bin/sh", "sh", 0);
  }
  return 0;
}
```

The second part of this example, setting the user ID and the group ID to 0 and starting a shell, is similar to what you find in the program patches of rootkits, which may require their own passwords first. Particularly interesting for such amendments are programs that already provide similar functions, for example *su* or *login*, as it is much more difficult to find the changes in these than in other programs.

Network backdoors

The second group of backdoors are network backdoors. These allow the cracker to enter a system he has already cracked at any time via the network without having a normal user account. Here again there are stand-alone backdoors and ones that hide within normal services. Patched versions exist of virtually all Internet daemons that have or can acquire root permissions, such as inetd and sshd.

If the attacker wants to use his or her backdoor the network version generally also requires a password. With well-hidden backdoors this password has to be entered at a point where other data would

normally be expected. If the cracker is recognised at the door it attaches a shell to the port or executes his commands and transfers the output.

There are many varieties of stand-alone network backdoors; the differences lie mostly in the way they are implemented or in the cryptographic method used. Most of these backdoors offer at least simple encryption in order to protect the transferred data from sniffers and therefore from the eyes of the administrator. This provides additional protection for crackers and makes it more difficult to trace their actions.

Network sniffers

The Ethernet sniffer forms an important part of many rootkits, as this allows the cracker to filter out important information from the network traffic and to store it. This will often enable him to crack other systems on the network, or to learn internal and confidential information.

The sniffer is generally a stand-along program that needs to be protected by a number of modified programs. For this reason rootkits containing a sniffer will offer patched versions of ifconfig, netstat and similar utilities, so that these normally dependable tools will no longer be able to find the sniffer.

Despite all protective measures there are many varied strategies for detecting rootkits in your system. With the help of a few tricks most of the rootkits currently in circulation can be tracked down and expelled from the system fairly easily.

Prevention is better

Ideally an administrator should take preventative measures to detect rootkits even before a break-in is suspected. In any case the administrator is going to require the most important system programs on a write-protected medium; it's vital that the administrator can guarantee these programs haven't been tampered with. They also need to be independent of all other files on the machine. For this reason they should ideally be statically linked, as manipulation could also occur within the shared libraries.

However, even these tools are powerless against a manipulation of the kernel. The only defence in this case is to boot up the system from a secure medium (boot disks with a disk operating system or a live CD), to mount the hard disk as read only and then to examine the system.

One important weapon in the fight against rootkits are checksums, which can be used to determine whether files have been changed. Simple CRC sums are not suitable, however, since some tools ensure that the patched file has the same CRC sum as the original. The checksums must be created before any manipulation can take place, ideally directly after the installation of the system. The lists should be stored on a write-protected medium to prevent attackers from tampering with them.

Armed with this type of list the administrator can check the system's programs on a regular basis. This test can also be completely automated using a cron job, provided that attackers can't amend the MD5 list and don't search the cron files for relevant jobs or modify either md5sum or the kernel.

Instead of the rather simple md5sum, more complex tools such as Tripwire or Aide are also suitable. However, in many cases the simpler program will be sufficient. The important thing above all is to back up and check the data often enough, and to ensure that no program is overlooked in this process.

Finding rootkits

Even if no backups and checksums are available, all is not lost; there are still some ways of finding rootkits. A tell-tale sign of many cracker tools stems from the fact that they require their own subdirectory or their own configuration file. Many rootkits write these to */dev* in the hope that no one will look there. This directory usually doesn't contain any normal files, only device files. A simple *find* call is enough to detect any interlopers (see also Figure 3):

```
find /dev -type f
```

The patched programs naturally try to open their config files. Consequently many modified files contain the string */dev*. This is also easy to find with the help of the *strings* command:

```
strings patched-file | grep /dev
```

A somewhat more complicated method of finding rootkits is the system call-trace. All system calls made by a program can be monitored using strace. The

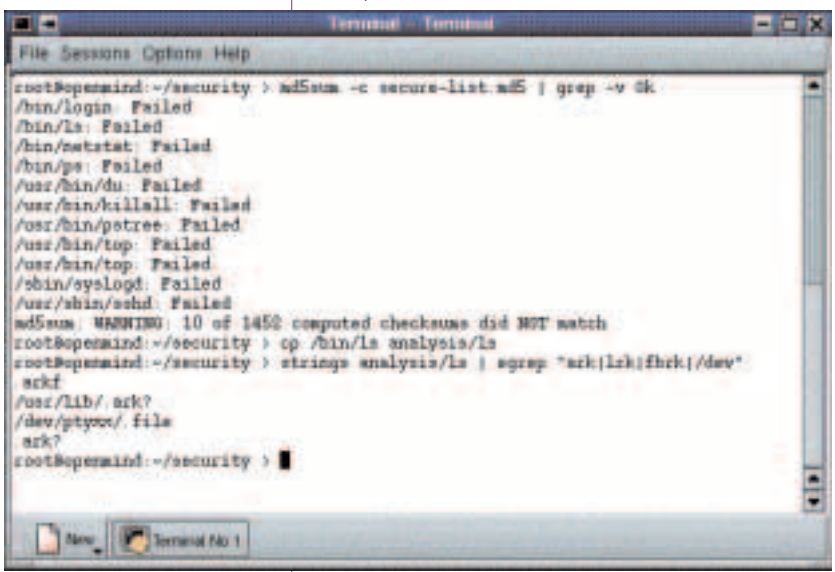

Figure 2: The program md5sum compares the current MD5 checksum with the stored values: 10 files have been modified. Using *strings* we can determine that we are dealing with Ambient's Rootkit (ARK)

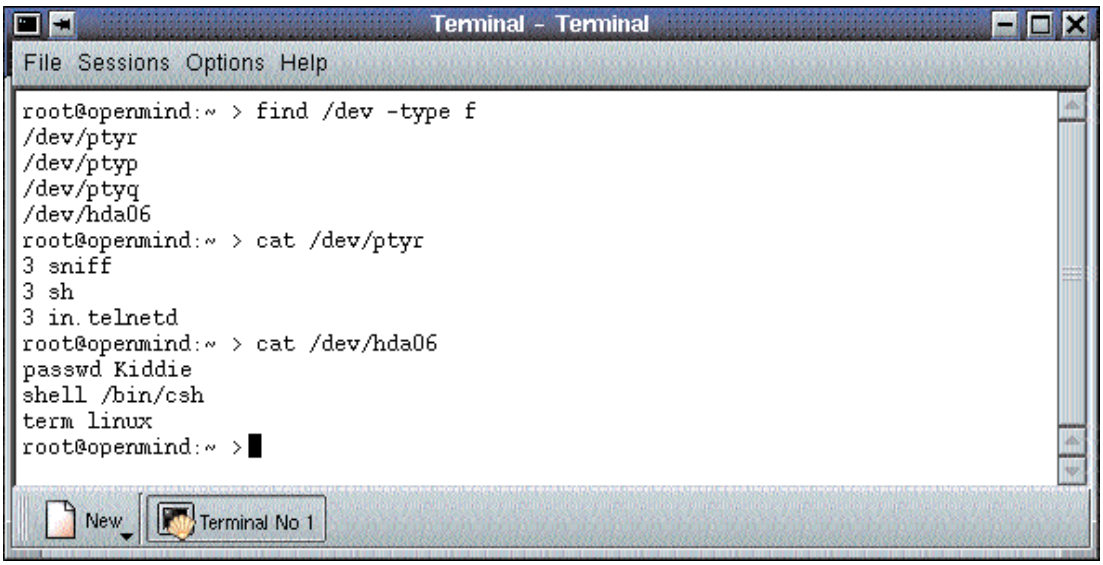

Figure 3: Telekit hides its configuration files in *dev*. A simple *find* call is enough to track them down

output of this can be quite sizeable, but also very revealing because when a rootkit wants to open one of its configuration files this is done using the relevant sys call.

/proc, the administrator's friend

When looking for rootkits the */proc* directory plays a very significant role. A lot of important information that rookits are trying to hide can be found here. The programmers of cracker tools are aware of this, and some rootkits do hide some of the information from */proc*, but this directory is still almost always worth a look.

Amongst other things, */proc* lists all processes that are running along with their process ID. If *ls* and *find* have not been manipulated in such a way that they will hide some of these files then a comparison of the output from */proc* with that of *ps* may already be enough to detect a rootkit. For hidden processes a look in */proc/PID/stat* or in the easier to understand */proc/PID/status* is well worth it. It is also possible that *ps* and *top* remain unchanged but that */proc* no longer shows every process.

Entries with network information can also be informative. The most important task is to check the open ports and to compare them to the output of *netstat* or a similar tool. This can be done in the folder */proc/net*, which contains files that will give you the open sockets for every protocol.

A portscan of your own machine from a secure source can also be very revealing. If this throws up open ports not shown by *netstat* this is a pretty definite indication of a rootkit. Unexpected open ports should set an administrator's alarm bells ringing in any case.

Rootkit found: what now?

Should your investigations actually turn up a rootkit, the first thing is to separate the system from the network. This enables you to examine the machine properly and to clean it. During your analysis it is not only important to find out who has broken in, more significant is how the cracker gained access to the system and which weak point he used to do it. This is where log files once again play a crucial role.

One important question is whether the system needs to be completely re-installed or whether it is sufficient to remove the modified programs and backdoors. There is no easy answer to this. It depends, amongst other things, on how much you can find out about the break-in and whether you can really trace all the cracker's actions. Otherwise it is possible that you might overlook a backdoor and the intruder could break in again as soon as the machine is back on the network. In that case nothing would have been gained from an administrator's point of view.

You can use the fact that crackers normally return to your advantage, however. Provided the administrator knows all of the cracker's access points, but not his identity, an administrator can set a trap for his adversary. A "honeypot" allows him to watch every action of the attacker, to track him and to study his behaviour.

Such studies enable the administrator to draw conclusions about the cracker's motives and to anticipate his future behaviour. If he has only cracked this machine by accident it is unlikely that he will bother it again. After all, he must assume that the administrator has now plugged the gaps and will be reading his log files with renewed interest. However, if the intruder was searching for internal information or had other motives for breaking into this host specifically then he probably continues to pose a threat.

You can use the fact that crackers normally return to your advantage

Info

"Rootkits – How Intruders Hide": *http://www.theorygroup.com/Theory/rootkits.html*
"Know Your Enemy": *http://project.honeynet.org/papers/*