



Rootkits: Avoiding Detection and Maintaining Control

Ferner Cilloniz Bicchi

Department of Computer Sciences
The University of Texas at Austin

Overview

- **Project Description**
 - Objective and Motivation.
- **History and Rootkit Basics**
 - User-level, library-level, kernel-level.
 - Infection.
- **Detection**
 - Removal.
- **Case Study**
 - LRK-5 (Linux Rootkit 5), Adore-ng.
 - Infection, control, and session using FreeKit.
- **Conclusion**

Project Description

- **Objective**

- Present what a rootkit is.
- Understand what rootkit designers attack in order to achieve their agenda.
- Compare the different types and levels of abstraction that rootkits have been designed and implemented in.
- Provide a real life example and interactive session.

- **Motivation**

- Victims to computer worms, viruses, exploits, etc, are growing by the numbers. However, not only do hackers use rootkits, but we have seen major corporations as Sony deploy similar malware on computers across the globe.
- Understanding what is carried out in this area must be met.

Rootkit Basics

- **What is a rootkit?**

- Hides the presence of the intruder.
- Main objective: keep all intruder activities **HIDDEN**.
 - Tricky: Mask all intruder activities as a normal execution of the system. Why? (hint: think anomaly detection).

- **What can a rootkit do?**

- **Remove processes** from the queue (keeps them hidden from reporting tools: ps, top, etc). Can also remove entries from the /proc file system.
- **Hide users** logged into the system (log wipers can achieve similar goals but can be detected as they are userland programs. More on the later).
- **Privilege escalation** (very easy once kernel code has been compromised).

Can anyone think of anything else?

Rootkit Basics (contd.)

- **Binary Rootkits**
 - First generation rootkit.
 - Replaces crucial system binaries with their backdoored counterpart.
 - `passwd`, `login`, `ps`, `pidof`, `ls`, etc.
- **Infection**: replacing the original binaries with their trojaned counterparts.
- These backdoored binaries allow for the attacker to hide processes, network connections, hide the fact that the network interface is in promiscuous mode (used to sniff network traffic), and escalate user privileges.
- Example: LRK-5 aka Linux Rootkit 5. (More on this later).

Backdoored ls binary.
How would you
detect this?



```
if( !strcmp(current_dirent, HIDE_DIRENTRY) ) {  
    ... // do some logic  
    return NOT_FOUND;  
}  
... // else proceed normally with ls
```

Rootkit Basics (contd.)

- **Library-level Rootkits**

- Replaces library function calls (libc in UNIX) with a backdoored version.
- **getuid(), setuid(), getdirenents(), etc.**
- **Infection:** replacing the original library modules (/lib) with their trojaned counterparts.
 - *Can anyone think of a way to have a user use a backdoored library?*

```
int setuid(uid_t uid)
{
    if( getuid() == HACKER_UID) {
        ... // do some logic
        return 0; // man setuid
    }

    return original_setuid(uid);
}
```

← What does this do?
How would you detect this?

Rootkit Basics (contd.)

- **Kernel-level Rootkits**

- Patch the kernel by supplying additional code, or removing existing code.
- **Attack Vectors:** hook system calls, modify system queues, modify the VFS (Virtual File System), etc.
- **Infection:** through loadable kernel modules (LKM) and/or inline byte code patching.

```
int hooked_write(unsigned int fd, char *buf, unsigned int count)
{
    if( !strcmp("give me r00t", buf) && fd == 1)
        current->uid = 0;

    return original_write(fd, buf, count);
}
```

Enough to do: echo "give me r00t!"
and the kernel will give you root
privileges in a Linux system!

How can you protect against this?

Detection

- Detection **does not** work the same for all types of rootkits.
- Detection methods only detect known attack techniques (Chicken-Egg problem).
 - **User-level:** suffice to periodically compute the hash of each system critical binary and compare for changes (a change flags a modification to the binary).
 - *Example: find / --exec sha1sum {} \;*
 - Special programs such as Check Rootkit, Tripwire, and Rootkit Hunter use this approach.
 - **Library-level:** can be done similarly as with user-level because the libraries are getting modified.

Detection (contd.)

- **Kernel-level:** much harder to detect.
 - Compare the entries in the **system call table** with the address of the system calls. If they don't match, then system call hooking is in play. **However, some legitimate kernel patches have this effect.**
 - Similar to intrusion detection systems, training kernel daemon to **detect anomalies** in the system can reveal suspicious behaviour.
 - ***What if rootkit disabled/modified the system calls and/or binaries the IDS is using?***

Detection (contd.)

- Removal
 - Unfortunately, there is no easy way to remove a rootkit.
 - **System binaries can be replaced** just as they were by the attacker, but there runs a risk of not replacing all backdoored binaries.
 - **Libraries can also be replaced** but might not find them all.
 - **Kernels can be updated...** but as with all before, there might be a corner in the system not found and hence the infection is still there.
- **Most Valuable Rule: don't get infected!**

Case Study

- **LRK-5: User-level Linux rootkit**
 - Published in November of 1998.
 - Provides backdoored counterparts for critical system binaries.
 - **Networking Binaries:** bindshell, tcpd, linsnir, inetd, snichk, netstat, rshd, ifcong.
 - **User Account Information:** chfn, chsh, passwd.
 - **Process Information:** ps, pidof, top, killall, crontab.
 - **Filesystem Binaries:** du, nd, ls, x.
 - **Log Wipers:** wted, z2.
 - **System Binaries:** login, syslogd.

Case Study (contd. LRK-5)

- **Installation:**

- Begins with replacing the original binaries with their trojaned counterparts. Only installs their backdoored version if the original binary can be found on the host machine.
 - Allows for the attacker to hide processes, network connections, hide the fact that the network interface is in promiscuous mode (what for?), and escalate user privileges.

- **Interaction:**

- Interaction by sending a series of commands to a file located in the /dev filesystem. However, because this is a user-level and not a kernel-level rootkit, the file being written to is not a device file but just a normal text file. Thus, any casual user can output its contents and reveal the configuration of the installed rootkit.

Case Study (contd.)

- Adore-ng: Kernel-level Linux and BSD rootkit
 - Adore-ng does not modify the system call table but hijacks the routines used by the Virtual File System (VFS) and, therefore, it is able to intercept and modify calls that access les in both the /proc le system and the root le system.
- **Interaction:**
 - Inserts itself into the running operating system as a kernel module.
- **Interaction:**
 - Interacting with Adore-ng is achieved by sending commands to special files in the /proc filesystem. As opposed to LRK-5, the files that Adore-ng uses to read the attacker's requests are not normal files but virtual files, created by special kernel functions. Therefore, it does not necessarily mean that the content sent to the files will be sent as output if any user, including the attacker, tries to open the les for reading.

Case Study (contd.)

- FreeKit: Modern Kernel-level Linux and BSD rootkit
 - Kernel-level rootkit.
 - Around 20KB in total.
 - Runtime kernel patching and LKM.
 - It allows for the attacker to infect a host machine and remain undetected for an undetermined amount of time.
 - Designed with detection and capabilities in mind, it achieves the following flawlessly:
 - Communication: through a character device `le` in the `/dev` filesystem.
 - Arbitrary assignment of user id.
 - Completely hide any process given either a name or process id.
 - Hide arbitrary TCP and UDP connections.
 - Hide arbitrary files and directories.
 - Record all keystrokes (keylogger).
 - Hide its presence and avoid detection.

Case Study (contd. FreeKit)

Lets look at how FreeKit can be used in an SSH session.



Conclusion

- **Don't let your system be vulnerable to attacks.**
- **After an attack, the intruder will install a rootkit.**
- **Understand what can be done once a system has been compromised.**
- **Know all the potential that backdoored binaries, backdoored libraries, and patched kernels can achieve.**
- **Left to the curious: what novel ideas can be used for future rootkits and detection?**

Questions?