MEGAMIND
The Rootkit Epidemic

E. Eugene Schultz, Ph.D., CISSP, CISM

Introduction

Malicious code (also called malware) has become increasingly sophisticated since the time the first virus surfaced in the wild around 1980. Malware such as viruses and worms attack are troublesome, yet they are generally easy to detect and eradicate once they infect a system. Viruses and worms also largely (but not exclusively) target Windows systems, largely leaving other types of systems alone. Other types of malware started to pose a proportionately greater degree of security threat several years ago when the allure and utility of writing and releasing viruses and worms started to fade because malware writers started to deploy more surreptitious malware because they become increasingly motivated by financial gain (SCHU06). Rootkits, in contrast, are designed to help attackers escape being noticed; they have, therefore, in particular proven much more troublesome than other types of malicious code. Rootkits are becoming so prevalent that to refer to the rootkit problem as an "epidemic" is becoming increasingly appropriate. This paper defines rootkits, explains how they work, explicates why they are likely to become even more prevalent, and wrestles with the issue of whether the war against rootkits will ever be won and if so, how.

What is a Rootkit?

A rootkit is a type of Trojan horse tool that if installed on a host modifies the hosts' operating system in a manner such that evidence of attackers' actions (including initial accesses to the systems and changes to the system made during installation of the program) are hidden. Attackers who have installed a rootkit can also generally use the rootkit to achieve remote back door access to the host at will. Rootkits often swap system programs and libraries with versions that look normal, but that in reality compromise the integrity of the victim host.

Two types of rootkits exist, user-mode and kernel-mode rootkits.

- User-mode rootkits substitute executables and system libraries that system administrators and users use. Changes that are made are systematically obfuscated such that if a system administrator lists a directory containing binaries of which one has been changed by the rootkit, no signs such as changes in last modification time and file size are displayed.

- Kernel-mode rootkits change parts of the kernel of the compromised host's operating system or may also even replace the kernel with an entirely new one. Process and other listings as well as in some cases kernel data structures are changed to hide kernel-related processes and other indications of the presence of the rootkit. In kernel-mode rootkits program execution flow is also frequently redirected so that instructions of the rootkit writer's choice, not normal ones, are

run in memory. Because many Linux and Unix systems have Loadable Kernel Modules (LKMs) that allow kernel functions to be modified without the need to edit the kernel, these systems have heightened vulnerability to kernel-level rootkits. Needless to say, kernel-level rootkits pose greater security risk than do user-level rootkits.

A good way to view the functionality and effects of user- and kernel-mode rootkits is to consider the rings of Intel x86 processors. Ring 3 is where unprivileged user-level instructions are run. In Ring 0 only privileged instructions can be run. User-mode rootkits run in Ring 3, whereas kernel-mode rootkits run in Ring 0 (see Figure 1 below).

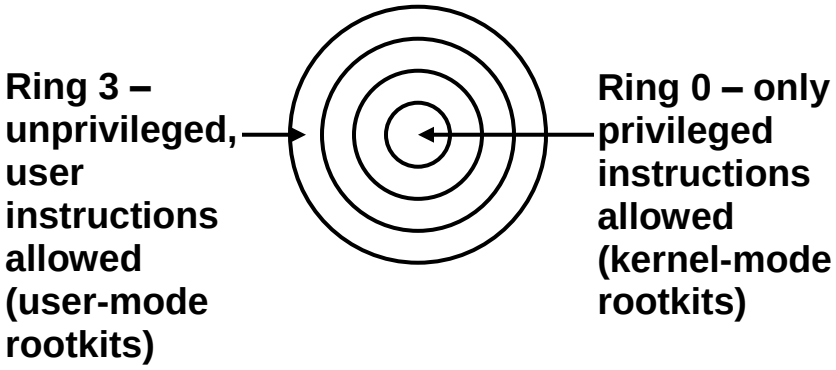

Figure 1 – Rings 0 and 3 in Intel x86 Processor Architectures

How Rootkits Work

As discussed by Shah (SHAH06), rootkits work in a variety of ways:

- Modifying kernel data structures. Modifying data structures, particularly the ones that list currently running processes, is commonplace part of rootkit functionality because modifying them goes far in masquerading the presence of a rootkit when system administrators and users enter commands that display processes.

- Modifying kernel memory. In Unix and Linux hosts /dev/mem is the file containing the kernel's memory image. Kernel-level rootkits access this file and then use it to edit the kernel in some manner such as swapping the system call table with a new version that executes malicious instructions. Modifying kernel memory has certain inherent dangers, however—one mistake can render a system completely dysfunctional. Additionally, any changes made are lost if the system is rebooted.

- Subverting virtual memory. Subversion of virtual memory is another mechanism that rootkits often use. The goal is often limiting what commands, applications, and kernel drivers are able to read from memory so that rootkit-caused changes and processes cannot be detected.

- Bypassing normal system call sequences. System call sequences are another frequent target of rootkits. A rootkit can either alter entries in the system call table to run malicious instructions or can alter the system call hander code itself to cause initial instructions to modify the execution flow by jumping to the rootkit call handler code.

- Altering interrupt handling. A rootkit can also alter interrupt handling, often by changing the interrupt descriptor table (IDT) so that the IDT entry containing the interrupt handler address runs rootkit instructions. Alternatively, a rootkit may change the interrupt handler code itself, usually by altering the first few instructions.

- Intercepting virtual file system calls. Some rootkits intercept virtual file system calls by swapping handler routines with new ones that conceal and/or filter information that might indicate the presence of a rootkit.

Prevalence of Rootkits

How prevalent are rootkits? It is difficult to obtain accurate statistics concerning rootkit prevalence because of the extreme difficulty to identifying rootkits. Despite this barrier, various statistics concerning rootkit prevalence have emerged over the last few years. Without exception they show a growth in the number of rootkits discovered in real-world settings. In 2006 Trend Micro found that the number of reported rootkits increased during that year and that rootkits were the most frequently found type of malware (TREN06). McAfee Labs has announced similar results, reporting that the number of rootkits given to these labs from the first quarter of 2005 to the first quarter of 2006 increased by nearly 700 percent. Additionally, McAfee has reported that the number of Windows-based stealth elements in malicious code grew 2300 percent from 2001 to 2005. Microsoft researchers also conducted a study in which they compared the number of rootkits found on Windows systems over a 15-month period (MICR06). They reported that the number of rootkits that were found increased over this period, although they credited the creation and use of the Microsoft Malicious Software Removal Tool for limiting the spread of rootkits. They also reported that although roughly 14 percent of the systems that were infected with some kind of malware had rootkits installed, although it is likely that this figure grossly underestimates the prevalence of rootkits—Microsoft's study included only Windows 2000, pre-SP2 versions of Windows XP, and Windows Server 2003 hosts. For some reason, Microsoft did not analyze hosts running SP2 versions of Windows XP, the by far most prevalently used versions of Windows operating systems at the time.

What particular rootkits are most prevalent? As would be expected because of the sheer prevalence of Windows systems, there are more Windows-based rootkits in the wild than any other type. Trend Micro has reported that the TROJ_ROOTKIT variant, one that runs in Windows systems, is the most frequently found family of rootkits (TREN06). These rootkits cover up indications that processes are running and other signs of the presence of malware by altering assigned functions in NTOSKRNL.EXE, the file on Windows

systems that is used in controlling a variety of kernel functions. They are often inserted in the Windows folder as .SYS files and then are registered as a service. Panda software recently announced that the most prevalent rootkits found in the wild are Beagle.FU and Adware/NaviPromo (BUST07). According to Panda, these two types of rootkits comprise almost 64 percent of all detected rootkits. Beagle.FU, of which there are many variants, targets Windows hosts. It is a kernel-mode rootkit that hides processes that run by altering kernel data structures. It also unlinks EPROCESS blocks (which are used in starting processes in Windows systems) from the list of active processes in the kernel. NaviPromo also targets Windows hosts and is in reality adware. Adware/NaviPromo copies itself into the system folder with the name mstmpreg32.dll, runs explorer.exe, inserts itself in the explorer.exe process, and deletes the original explorer.exe file. This rootkit also creates several .exe files and then adds them to the Windows folder. Finally, it uses MSClock32.dll to kill the functionality of certain system functions that could otherwise result in detection of this rootkit.

The Future of Rootkits

A number of factors greatly increase the probability that rootkits will become even more prevalent over time. These include:

- The sheer number of vulnerabilities in hosts that connect to the Internet and the rate at which these vulnerabilities surface. Rootkits do not exploit vulnerabilities. To install a rootkit, an attacker must first exploit a vulnerability to gain access to the targeted system and often must exploit another to gain superuser access such as root access on Unix and Linux hosts. Vulnerabilities are surfacing at an unprecedented rate; many of them are zero-day vulnerabilities. It is thus nearly impossible to keep hosts and applications properly patched; in fact, many organizations and individuals do not even try. This creates a target rich environment for rootkits; the same will be even more true in the future when vulnerabilities surface at an even faster rate if the present trend continues.

- Web, chat and instant messaging collaboration among rootkit writers. Writing a rootkit requires a very high level of programming proficiency. However, the widespread availability of collaborative methods that enable rootkit writers who cannot leap one or more hurdles in authoring a new rootkit greatly compensates for any lack of programming proficiency.

- The creation, proliferation and distribution of highly effective rootkits. Rootkits have grown considerably in both number and sophistication from year-to-year. Some of them are nearly impossible to detect, even by the most proficient technical staff. Obtaining rootkits has on the other hand also become increasingly easy to the point where even an amateur perpetrator can readily download and install highly effective rootkits. Rootkits of the future will almost certainly include considerably more functionality, making them even more difficult to detect.

- Lack of tools that effectively detect and eradicate rootkits. AUSCERT, the Australian CERT Team, found that commonly used anti-virus software fails to detect up to 80 percent of Trojan horse programs that reside in systems. This finding is easy to explain; anti-virus software is designed more than anything else to discover viruses and worms by using signatures. But signature-based detection, rootkit tools very much included, is much less likely to work when malware is purposefully clandestine. Furthermore, rootkits are much more surreptitious than are "normal" Trojan programs. The inescapable conclusion, therefore, is that any statistics concerning the prevalence of rootkits must almost certainly seriously underestimate the actual prevalence of this type of malware. Tools specifically designed to detect rootkits (such as chkrootkit in Linux and Rootkit Revealer in Windows systems) are generally somewhat better in their detection rates than is anti-virus software, but they nevertheless miss a substantial proportion of rootkits in systems.

- Lack of privilege control in widely used systems and applications. As mentioned previously, rootkit installation requires superuser privileges. The fact that such a large proportion of users as well as system administrators who engage in computing functions not related to system management (e.g., Web surfing) engage in these activities with superuser privileges makes their hosts much more conducive to being infected by rootkits when they visit malicious Web sites, are infected by viruses and worms, and so on.

- The incorporation of rootkit functionality into spyware. Spyware routinely makes its way into hosts, particularly Windows hosts; no anti-spyware tools are anywhere near being 100 percent effective in preventing spyware infections. An increasing number of spyware programs are incorporating rootkit functionality, thereby not only infecting systems, but also hiding any evidence of an infection and any malicious activities such as capturing all traffic that goes in and out of the victim host. The likely result of this trend is once again for rootkits to become more prevalent over time.

- Lack of security knowledge among system administrators and users. Lack of security knowledge among system administrators and users will also contribute to the target-rich environment that will help rootkits to become even more prevalent than they are now.

Is there Hope in the Fight against Rootkits?

Is there hope in the fight against rootkits? The answer is "maybe," but definitely "no" if the fight against rootkits continues to keep going in the direction it has gone until now. Simply put, anti-rootkit technology has not kept pace with the sophistication of rootkits. If this trend continues the gap between what is needed in this fight and the capabilities of rootkits will continue to grow larger. Rootkit detection and eradication tools are almost without exception not up to the task. Signature-based tools are most inadequate, but even more effective tools suffer from the problem that they must run on systems in which the

integrity of the processes and files that these tools must examine may possibly be compromised.

One potential future technology solution that has been widely considered is running rootkit detection and eradication processes in a virtual environment that cannot be affected by any rootkit. In theory this is a brilliant solution, but in reality rootkits can even subvert the integrity of virtual environments. Joanna Rutkowska has developed what she has termed a completely undetectable rootkit (called "Blue Pill") that evades the Vista operating system's integrity-checking during loading unsigned code into the Vista kernel using AMD's secure virtual machine to conceal itself (RUTK06),. Even virtual machines and environments are subject to rootkit-causes integrity compromises; their potential value in detecting and eradicating rootkits thus appears to be limited.

The best rootkit detection and eradication solution to date is hardware-based. Komoku has produced a rootkit detection product, the Copilot PCI Card™, a PCI card that has its own CPU and memory. It inspects the physical memory of systems that are potentially compromised by rootkits. Because it has its own CPU and memory, its processes cannot potentially be affected by a rootkit's subversive mechanisms. The main limitation to this solution presently is the financial cost, which is high. Hopefully, in time the cost will come down, as typically has happened with other new technologies.

Another promising and more cost-effective solution that is considerably better than most detection and eradication solutions available today and that is likely to become even better in time is Security Event Management (SEM) technology. SEM tools use sophisticated event correlation algorithms to identify symptoms of a rootkit infection as are visible on the network. Rootkits may be able to hide themselves on *hosts*, but they at various times give indications of their presence on the *network*, e.g, when the attacker who installed the rootkit gains backdoor access to the rootkit. Indications such a certain port on a rootkit-compromised host temporarily listening for input and the presence of encrypted traffic between an internal and external host after the initial traffic between the hosts was unencrypted can provide enough information for a SEM's event correlation algorithms to detect the presence of a rootkit.

Ultimately, however, the best solution is to build rootkit resistant mechanisms directly into operating systems and applications. Add-on solutions such as rootkit detection and eradication tools are invariably less effective than are ones that are built-in from the start. Operating systems in particular should recognize and halt attempts to engage in integrity-compromising actions such as those that result in modification of critical kernel data structures. It is well time for operating system and application vendors to wake up to the fact that they hold the real key in the fight against rootkits and other kinds of malware.

References

AUSC06 AUSCERT. Eight-percent of new malware defeats anti-virus. Web posting, 2006. www.zdnet.com.au/.../soa/Eighty-percent-of-new-malware-defeats-antivirus/0,130061744,139263949,00.htm

BUST07 Bustamante, P. Rootkits in the mist. Web posting, 2007. http://research.pandasecurity.com/archive/Rootkits-in-the-mist.aspx

MCAF06 McAfee. McAfee warns about increasing prevalence of rootkits. Web posting, 2006. http://www.playfuls.com/news_02100_McAfee_Warns_About_Increasing_Prevalence_of_Rootkits.html

MICR06 Microsoft, 2006. Windows Malicious Software Removal Tool: Progress made, trends observed. Web posting, 2006. www.microsoft.com/downloads/details.aspx?FamilyId=47DDCFA9-645D-4495-9EDA-92CDE33E99A9

PHIF06 Phifer, L. Rootkits 201: Countermeasures and defenses. Web posting, 2006. http://www.corecom.com/external/livesecurity/rootkits201.htm

SCHU06 Schultz, E.E. Where have viruses and worms gone? New trends in malware. Computer Fraud and Security. July, 2006, pp. 2 – 7.

SHAH06 Shah, A. Analysis of rootkits: Attack approaches and detection mechanisms. Web posting, 2006. http://www-static.cc.gatech.edu/~salkesh/files/RootkitsReport.pdf

TREN06 Trend Micro. The trend of threats today: 2006 annual roundup and 2007 forecast. Web posting, 2006. http://us.trendmicro.com/imperia/md/content/us/pdf/threats/securitylibrary/trend_annualreport06.pdf