Rootkit

From Wikipedia, the free encyclopedia

A **rootkit** is malware which consists of a program (or combination of several programs) designed to hide or obscure the fact that a system has been compromised. Contrary to what its name may imply, a rootkit does not grant a user administrator access as it requires such access to execute and tamper with system files and processes. An attacker may use a rootkit to replace vital system executables which may then be used to hide processes and files the attacker has installed along with the presence of the rootkit itself. Access to the hardware (e.g., the reset switch) is rarely required as a rootkit is intended to seize control of the operating system running on the hardware. Typically, rootkits act to obscure their presence on the system through subversion or evasion of standard operating system security mechanisms. Often, they are Trojans as well, thus fooling users into believing they are safe to run on their systems. Techniques used to accomplish this can include concealing running processes from monitoring programs, or hiding files or system data from the operating system.[1] Rootkits may also install a 'backdoor' in a system by replacing the login mechanism (such as /bin/login) with an executable that accepts a secret login combination which in turn allows an attacker to access the system regardless of changes to the actual accounts on the system.

Rootkits may have originated as regular applications, intended to take control of a failing or unresponsive system, but in recent years have been largely malware to help intruders gain access to systems while avoiding detection. Rootkits exist for a variety of operating systems, such as Microsoft Windows, Linux, Mac OS, and Solaris. Rootkits often modify parts of the operating system or install themselves as drivers or kernel modules, depending on the internal details of an operating system's mechanisms.

Contents

History

The term *rootkit* or *root kit* originally referred to a maliciously modified set of administrative tools for a Unix-like operating system. If an intruder could replace the standard administrative tools on a system with a rootkit, the modified tools would allow the intruder to maintain administrative control over the system while concealing his activities from the legitimate system administrator. The earliest known rootkit was written in about 1990 by Lane Davis and Steven Dake for SunOS 4.1.1. There was an earlier, quite famous, exploit equivalent to a rootkit which was perpetrated by Ken Thompson of Bell Labs against a Naval Laboratory in California to win a bet. Thompson subverted the C compiler in a distribution of Unix to the Lab.

Contrary to the name, rootkits do not give an intruder administrator access. In order to install itself in a system by replacing crucial operating system administration files, it would require the intruder to already have root or administrator access. This is achieved by exploiting a vulnerability in a computer system which would lead to user or administrator level access. Privilege escalation type exploits are normally used to gain administrator access before an attacker installs a rootkit to hide his activities, thus allowing him to maintain administrator access without the knowledge of the system owner.

In 2005, Sony BMG caused a scandal by including rootkit software on music CDs that, in an attempt to enforce DRM,[2] opened a backdoor that allowed root access to anyone aware of the rootkit's installation.[3] The scandal raised the public's awareness of rootkits, while the public relations fallout for Sony was compared by one analyst to the Tylenol scare.[4]

Common use

A successfully-installed rootkit allows unauthorized users to maintain access as system administrators, and thus to take and keep full control of the 'rootkitted' - or 'rooted' - system. Most rootkits typically hide files, processes, network connections, blocks of memory, or Windows Registry entries from other programs used by system administrators to detect specially privileged accesses to computer system resources. However, a rootkit may masquerade as or be intertwined with other files, programs, or libraries with other purposes. It is important to note that while the utilities bundled with a rootkit may be maliciously intended, not every rootkit is always malicious. Rootkits may be used for both productive and destructive purposes. Nonetheless, the use of another person's or organization's computing resources without their consent is unethical - and quite probably illegal - in most cases.

Many rootkits hide utility programs. Those that do so usually aim to abuse a compromised system, and often include a so-called "backdoor" to give the attacker subsequent access at will. A simple example might be a rootkit which hides an application that spawns a command processing shell when the attacker connects to a particular network port on the system. Kernel rootkits may include similar functionality. A backdoor may also allow processes started by a non-privileged user to run as though it were started by a privileged user (including the root user) and to carry out functions normally reserved for the superuser. Rootkits are hard to detect with common antivirus programs and therefore a complete scan of the system is necessary.

Many other utility tools used for abuse can be hidden using rootkits. These include tools for further attacks against computer systems with which the compromised system communicates, such as sniffers and keyloggers. A possible form of abuse is using a

compromised computer as a staging ground for further abuse (see zombie computer). This is often done to make the abuse appear to originate from the compromised system (or network) instead of the attacker's. Tools for such attacks can include denial-of-service attack tools, tools to relay chat sessions, and e-mail spam distribution. Rootkits are normally used in conjunction with other malicious programs as a means to keep them undetectable from the eyes of the user and antivirus scans.

It has become increasingly popular for virus writers to make use of rootkit technologies. The reason for this is that they make it possible to hide malware from PC users and antivirus programs. Numerous source code samples for ready-made rootkits can be found on the Internet, which inevitably leads to their widespread use in various trojans or spyware programs et cetera.

However, rootkits are not always used to maintain control of a computer. Some software may use rootkit techniques to hide from 3rd party scanners to detect tampering or attempted breakins, for example in a honeypot. Some emulation software and security software is known to use rootkits.[5] Alcohol 120% and Daemon Tools are commercial examples of the use of non-hostile rootkits. Kaspersky antivirus software also uses some techniques somewhat resembling rootkits to protect itself from malicious virus actions. It loads its own drivers to intercept system activity and then prevents other processes from doing harm to itself. So while its processes are not hidden, such processes cannot be terminated by standard methods.

Rootkit is a term now somewhat loosely applied to cloaking techniques and methods.[6]

Types

There are at least five kinds of rootkits: firmware, hypervisor, kernel, library, and application level kits.

Hardware/Firmware

A firmware rootkit uses device or platform firmware to create a persistent malware image. The rootkit can successfully hide in firmware because firmware is not often inspected for code integrity. John Heasman demonstrated the viability of firmware rootkits in both ACPI firmware routines[7] and in a PCI expansion card ROM.[8]

In October 2008, the press reported that European credit card swiping machines had been tampered with while still in the supply chain, and that these devices were intercepting customers' credit card details before transmitting the data to international criminals via the mobile phone network.[9]

Hypervisor level

These rootkits work by modifying the boot sequence of the machine to load themselves as a hypervisor under the original operating system. By exploiting hardware features such as Intel VT or AMD-V, the rootkit is able to load the original operating system as a virtual machine, thereby enabling it to intercept all hardware calls made by the original operating system, which is now a guest. The "SubVirt" laboratory rootkit, developed jointly by Microsoft and University of Michigan researchers, is an academic example of a virtual machine based rootkit (VMBR),[10] while Blue Pill is another.

Kernel level

Kernel-level rootkits add additional code and/or replace portions of an operating system, including both the kernel and associated device drivers. Most operating systems support kernel-mode device drivers, that execute with the same privileges as the operating system itself.[11] As such, many kernel mode rootkits are developed as device drivers or loadable modules, such as loadable kernel modules in Linux or device drivers in Microsoft Windows. This class of rootkit is perceived as dangerous simply because of the unrestricted security access the code has obtained, regardless of the features the rootkit may employ. Any code operating at the kernel level may have serious impacts on entire system stability if bugs are present in the code. The first and original rootkits did not operate at the kernel level, but were simple replacements of standard programs at the user level. One of the first widely known kernel rootkit was developed for Windows NT 4.0 and released in Phrack issue 55 (http://phrack.org/issues.html?issue=55&id=5) in the mid-1990s by Greg Hoglund.[12]

Kernel rootkits can be especially difficult to detect and remove because they operate at the same level as the operating system itself, and are thus able to intercept or subvert any operation made by the operating system. Any software, such as antivirus software, running on the comprised system is equally easily subverted. In a situation such as this, the whole system can no longer be trusted while it is running. One response in such a case is to perform system offline analysis from a second 'trusted' system by mounting the hard drive of the infected system as a secondary resource without executing anything on the untrusted volume, while another is to format the disk and re-install from trusted media. Lastly, a (known good) operating system can be booted from read-only removable media such as a CD-ROM. Investigation and rootkit removal actions can then be performed safely in trusted system without requiring another computer system. So called live CDs booting into read-only OS installation are useful for such tasks. Read-write media like HDD or USB Flash to boot systems while removing rootkits is dangerous due to chances that rootkit may affect this OS installation as well.

Library level

Library rootkits commonly patch, hook, or replace system calls with versions that hide information about the attacker. They can be found, at least theoretically, by examining code libraries (under Windows the term is usually DLL) for changes or against the originally distributed (and so presumably rootkit free) library package; this approach may not succeed however if the code is patched in memory only. Digital signatures help to detect unauthorised changes to code libraries.[13]

Application level

Application level rootkits may replace regular application binaries with Trojan fakes, or they may modify the behavior of existing applications using hooks, patches, injected code, or other means.

There are unscrupulous companies whose business consists of disseminating rootkits for the purpose of generating paid involuntary page referrals. One frequent victim is users of Google which has the largest number of deliberate visits and so can give them income by misdirecting searches.

Detection

Rootkit binaries can often be detected by signature or heuristics based antivirus programs, at least until they're run by a user and are able to attempt to conceal themselves. There are inherent limitations for any program that attempts to detect

rootkits while the program is running under the suspect system. Rootkits are suites of programs that modify many of the core system tools and libraries upon which all programs on the system depend. Some rootkits attempt to modify the running operating system via loadable modules on Linux (and some other UNIX varieties), and through VxDs, virtual external device drivers on MS Windows platforms. The fundamental problem with rootkit detection is that if the operating system currently running has been subverted, it cannot be trusted, including to find unauthorized modifications to itself or its components. In other words, actions such as requesting a list of all running processes, or a list of all files in a directory, cannot be trusted to behave as intended by the original designers. Rootkit detectors running on live systems currently only work because the rootkits they can detect have not yet been developed to hide themselves fully against these detectors. A reasonable analogy would be asking a brainwashed person if they had been brainwashed; obviously their answer could not be trusted.

The best, and most reliable, method for rootkit detection is to shut down the computer suspected of infection, and then check its secondary storage by booting from an alternative medium (e.g., a rescue CD-ROM or USB flash drive). A non-running rootkit cannot actively hide its presence, and most established antivirus programs will identify rootkits armed via standard OS calls (which are often tampered with by the rootkit) and lower level queries, which ought to remain reliable. If there is a difference, the presence of a rootkit infection should be assumed. Running rootkits attempt to protect themselves by monitoring running processes and suspending their activity until the scanning has finished; this is more difficult if the rootkit is not allowed to run.

Security software vendors have attempted a solution by integrating rootkit detection into traditional antivirus products. Should a rootkit decide to hide during scanning, it will be identified by the stealth detector. If it decides to temporarily unload from the system, the traditional antivirus will find it using fingerprint detection. Since anti-virus products are almost never entirely capable of catching all viruses in public tests, this approach may be doubted on past behavior. But this combined approach may force attackers to implement counter-attack mechanisms (so called retro routines) in their rootkit code that will forcibly remove security software processes from memory, effectively killing the antivirus program. As with computer viruses, the detection and elimination of rootkits will be an ongoing struggle between tool creators on both sides of this conflict.

There are several programs available to detect rootkits. On Unix-based systems, three of the most popular are chkrootkit, rkhunter and OSSEC. For Windows, there are many free detection tools such as avast! antivirus, Sophos Anti-Rootkit (http://www.sophos.com/products/free-tools/sophos-anti-rootkit.html) , and F-Secure Blacklight. Another Windows detector is RootkitRevealer from Microsoft (formerly Sysinternals) which detects rootkits by comparing results from the OS against expected results obtained by bypassing the operating system and analysing the raw underlying structures in the file system (**cross-checking**). However, some rootkits started to add RootkitRevealer to a list of files it does not hide from, so in essence, they remove differences between the two listings, and the detector doesn't report them (most notably the commercial rootkit *Hacker Defender Antidetection*). Rootkit Revealer has apparently fixed this problem as they stated on their official page: "The reason that there is no longer a command-line version is that malware authors have started targetting RootkitRevealer's scan by using its executable name. We've therefore updated RootkitRevealer to execute its scan from a randomly named copy of itself that runs as a Windows service. This type of execution is not conducive to a command-line interface. Note that you can use command-line options to execute an automatic scan with results logged to a file, which is the equivalent of the command-line version's

behavior."[14]

Another method is to compare content of binaries present on disk with their copies in operating memory — however some valid differences can be introduced by operating system mechanisms (e.g., memory relocation or shimming), but some can be very likely classified as system call hooks introduced by a running rootkit (**System Virginity Verifier**). Zeppoo is another software product which detects rootkits under Linux and UNIX systems.

As always, prevention is better than cure, for being certain you have removed a rootkit typically involves re-installation of all software. If the integrity of the system install disks is trusted, cryptography can be used to monitor the integrity of the system. By "fingerprinting" the system files immediately after a fresh system install and then again after any subsequent changes made to the system (e.g., installing new software), the user or administrator will be alerted to any dangerous changes to the system's files. In the fingerprinting process a message digest is used to create a fixed-length "digest" dependent on every bit in the file being fingerprinted. By calculating and comparing message digest values of files at regular intervals, changes in the system can be detected.

Detection in firmware can be achieved by computing a cryptographic hash of firmware and comparing hash values to a whitelist of expected values, or by extending the hash value into TPM (Trusted Platform Module) configuration registers, which are later compared to a whitelist of expected values. Code that performs hash, compare, and/or extend operations must itself not be compromised by the rootkit. The notion of an immutable (by a rootkit) root-of-trust, if implementable, ensures that the rootkit does not compromise the system at its most fundamental layer. A method of rootkit detection using a TPM is described by the Trusted Computing Group[15]

Removal

Many hold this to be forbiddingly impractical. Even if the nature and composition of a rootkit is known, the time and effort of a system administrator with the necessary skills or experience would be better spent re-installing the operating system from scratch. Since drive imaging software makes the task of restoring a “clean” OS installation almost trivial, there is no good reason to try to dig a rootkit out directly.

' "*I suppose traditional rootkits could be made to be as hard to remove as possible even when found, but I doubt there is much incentive for that, because the typical reaction of an experienced sysadmin on finding a rooted system is to save the data files, then reformat [and reinstall]. This is so even if the rootkit is very well known and can be removed 100%.*" [16] While most Anti-Virus and Malware Removal tools remain ineffective against rootkits, tools such as BartPE and other Preinstallation Environment(PE) or Live Distros allow users to boot their computer with a fresh (presumably) "un-rooted" copy of the operating system. This allows users to examine and replace affected system files and delete offending rootkits of most types while keeping the underlying systems intact. Since most rootkits hook system files needed at the lowest level of the OS, booting into Safe Mode will not usually allow removal of the rootkit process. In contrast, PE's do not rely on the infected underlying system structure but instead load a clean read-only copy of the Operating System allowing full control and detection of the rootkit. While most Administrators prefer a clean reinstall, a skilled Administrator using a PE can often delete the rootkit and clean a rooted system if a reinstall is not a viable option.

Symantec Veritas VxMS (Veritas Mapping Service), on the other hand, lets the antivirus

scanner bypass Windows File System APIs, which are controlled by the operating system and therefore vulnerable to manipulation by a rootkit. VxMS directly accesses raw Windows NT File System files. In effect, VxMS can map data, compare it to the Windows file structure, and isolate any discovered mismatches. VxMS technology has been extended to Symantec's Norton product line from the 2007 product year. [17] [18] [19] [20] [21]

Comparison with computer viruses and worms

The key distinction between a computer virus and a rootkit relates to propagation. Like a rootkit, a computer virus modifies core software components of the system, inserting code which attempts to hide the "infection" and provides some additional feature or service to the attacker (i.e., the "payload" of a virus).

In the case of the rootkit the payload may attempt to maintain the integrity of the rootkit (the compromise to the system) --- for example every time one runs the rootkit's version of the *ps* command, it may check the copies of *init* and *inetd* on the system to ensure that they are still compromised, "re-infecting" as necessary. The rest of the payload is there to ensure that the intruder continues to control the system. This generally involves having backdoors in the form of hard-coded username/password pairs, hidden command-line switches or 'magic' environment variable settings which subvert normal access control policies of the uncompromised versions of the programs. Some rootkits may add port knocking checks to existing network daemons (services) such as *inetd* or the *sshd*.

A computer virus can have any sort of payload. However, the computer virus also attempts to spread to other systems. In general, a rootkit limits itself to maintaining control of one system.

A program or suite of programs that attempts to automatically scan a network for vulnerable systems and to automatically exploit those vulnerabilities and compromise those systems is referred to as a computer worm. Other forms of computer worms work more passively, sniffing for usernames and passwords and using those to compromise accounts, installing copies of themselves into each such account (and usually relaying the compromised account information back to the intruder through some sort of covert channel).

There are also hybrids. A worm can install a rootkit, and a rootkit might include copies of one or more worms, packet sniffers or port scanners. Also many of the e-mail worms are commonly referred to as "viruses." So all of these terms have somewhat overlapping usage and are often conflated.

Public availability

Like much malware used by attackers, many rootkit implementations are shared and are easily available on the Internet. It is not uncommon to see a compromised system in which a sophisticated publicly available rootkit hides the presence of unsophisticated worms or attack tools that appear to have been written by inexperienced programmers.

Most of the rootkits available on the Internet are constructed as an exploit or "proof of concept" to demonstrate varying methods of hiding things within a computer system and of taking unauthorized control. Since these are often not fully optimized for stealth, they sometimes leave unintended evidence of their presence. Even so, when such rootkits are used in an attack they are often very effective.

See also

- Hacker con
- Computer virus
- Host-based intrusion detection system
- The SANS Institute
- 2005 Sony BMG CD copy protection scandal

References

1. ^ Brumley, David (1999-11-16). "invisible intruders: rootkits in practice". USENIX. http://www.usenix.org/publications/login/1999-9/features/rootkits.html.
2. ^ Mark Russinovich (2005-10-31). "Sony, Rootkits and Digital Rights Management Gone Too Far". http://blogs.technet.com/markrussinovich/archive/2005/10/31/sony-rootkits-and-digital-rights-management-gone-too-far.aspx. Retrieved on 2008-09-15.
3. ^ "New backdoor program uses Sony rootkit". Kaspersky Lab. 2005-11-10. http://www.kaspersky.com/news?id=173737204. Retrieved on 2008-09-15.
4. ^ "Sony's long-term rootkit CD woes". BBC News. 2005-11-21. http://news.bbc.co.uk/2/hi/technology/4456970.stm. Retrieved on 15 September 2008.
5. ^ Russinovich, Mark (2006-02-06). "Using Rootkits to Defeat Digital Rights Management". *Winternals*. SysInternals. Archived from Using Rootkits to Defeat Digital Rights Management the original on 2006-08-31. http://blogs.technet.com/markrussinovich/archive/2006/02/06/using-rootkits-to-defeat-digital-rights-management.aspx. Retrieved on 2006-08-13.
6. ^ Mark Russinovich (June 2005). "Unearthing Root Kits". Windows IT Pro. http://www.windowsitpro.com/Article/ArticleID/46266/46266.html.
7. ^ Implementing and Detecting an ACPI Rootkit (http://www.blackhat.com/presentations/bh-europe-06/bh-eu-06-Heasman.pdf) by John Heasman, presented at BlackHat Federal, 2006.
8. ^ Implementing and Detecting a PCI Rootkit (http://www.ngssoftware.com/research/papers/Implementing_And_Detecting_A_PCI_Rootkit.pdf) by John Heasman, 15 November, 2006.
9. ^ Austin Modine (2008-10-10). "Organized crime tampers with European card swipe devices: Customer data beamed overseas". The Register. http://www.theregister.co.uk/2008/10/10/organized_crime_doctors_chip_and_pi Retrieved on 2008-10-13.
10. ^ "SubVirt: Implementing malware with virtual machines" (PDF). University of Michigan, Microsoft. 2006-04-03. http://www.eecs.umich.edu/virtual/papers/king06.pdf. Retrieved on 2008-09-15.
11. ^ The 64-bit version of Windows XP and Server 2008 are a notable exception "Driver Signing Requirements for Windows". Microsoft. http://www.microsoft.com/whdc/winlogo/drvsign/drvsign.mspx. Retrieved on 2008-07-06.
12. ^ Rootkit Evolution (http://www.net-security.org/article.php?id=1173&p=2) by Alisa Shevchenko (1 September 2008), An Overview of Unix Rootkits (http://www.megasecurity.org/papers/Rootkits.pdf) by Anton Chuvakin (February 2003)
13. ^ "Signing and Checking Code with Authenticode". Microsoft. http://msdn.microsoft.com/en-us/library/ms537364(VS.85).aspx. Retrieved on 2008-09-15.
14. ^ "RootkitRevealer v1.71". Microsoft Technet. 2006-11-01. http://technet.microsoft.com/en-us/sysinternals/bb897445.aspx. Retrieved on 2008-10-10.
15. ^ "Stopping Rootkits at the Network Edge" (PDF). Trusted Computing Group. 2007-01-04. https://www.trustedcomputinggroup.org/news/Industry_Data/Whitepaper_Rootkit_Strom_v3.pdf. Retrieved on 2008-07-11.
16. ^ http://forums.spywareinfo.com/lofiversion/index.php/t52360.html

Rootkit Question
17. ^ http://techloop.blogspot.com/2007/04/rootkits-next-big-enterprise-threat.html
18. ^ http://reviews.cnet.com/4520-3513_7-6686763-1.html
19. ^ http://www.pcworld.com/businesscenter/article/137821/six_ways_to_fight_back_against_botnets.
20. ^ https://forums.symantec.com/t5/Malicious-Code/Handling-Today-s-Tough-Security-Threats-Rootkits/ba-p/305215;jsessionid=7D537BC23D36E
21. ^ http://www.infoworld.com/article/07/04/30/18FErootkit_3.html

Further reading

- Robert S Morris, Sr. (1984). "UNIX Operating System Security". *Bell Systems Technical Journal* (BSTJ) **Vol. 62, No. 8**.
- Greg Hoglund and James Butler (2005). *Rootkits: Subverting the Windows Kernel*. Addison Wesley. ISBN 0-321-29431-9. http://books.google.com/books?id=fDxg1W3eT2gC.
- Nancy Altholz and Larry Stevenson (2006). *Rootkits for Dummies*. John Wiley and Sons Ltd. ISBN 0-471-91710-9. http://books.google.com/books?id=MTcep7V6heUC.
- Ric Veiler (2007). *Professional Rootkits*. Wrox. ISBN 978-0-470-10154-4. http://books.google.com/books?id=OSaVF_jzsJgC.

External links

- Even Nastier: Traditional RootKits (http://www.informit.com/articles/article.aspx?p=23463)
- Sophos Podcast about rootkit removal (http://podcasts.sophos.com/en/sophos-podcasts-004.mp3)
- Featured Article: Rootkit Unhooker v3.8 It's Past, Present and Future of the NTx86 Rootkit Detection by DiabloNova (http://www.rootkit.com/newsread.php?newsid=902)
- Rookit research in Microsoft (http://research.microsoft.com/Rootkit/)
- White paper on new-generation rootkit detection (http://northsecuritylabs.com/downloads/whitepaper-html/)
- antirootkit.com (http://www.antirootkit.com)
- Testing of antivirus/anti-rootkit software for the detection and removal of rootkits (http://www.anti-malware-test.com/?q=taxonomy/term/7) made by Anti-Malware Test Lab, January 2008
- Testing of anti-rootkit software (http://www.informationweek.com/news/showArticle.jhtml?articleID=196901062) made by InformationWeek, January 2007
- Sony, Rootkits and Digital Rights Management Gone Too Far (http://blogs.technet.com/markrussinovich/archive/2005/10/31/sony-rootkits-and-digital-rights-management-gone-too-far.aspx) (Mark Russinovich's first blog entry about the Sony DRM rootkit, from which the scandal ensued)
- Designing BSD Rootkits (http://www.oreilly.com/catalog/1593271425/) An Introduction to Kernel Hacking (book by Joseph Kong)
- How to remove spyware from your PC: rid yourself of rootkits (http://www.pcworld.ca/news/column/23a7f73b0a010408001a024ccab0dd5e/pg1.htm)
- Glossary of malware terminology (http://www.antispywarecoalition.org/documents/glossary.htm) ("Rootkit" has a negative connotation)
- White paper on hypervisor rootkit technology (http://www.crucialsecurity.com

/documents/hvmrootkits.pdf)
- Review: Six Rootkit Detectors Protect Your System (http://www.informationweek.com/news/software/reviews/showArticle.jhtml?articleID=196901062)

Retrieved from "http://en.wikipedia.org/wiki/Rootkit"
Categories: Malware | Rootkits | WikiProject Computer Security articles
Hidden categories: All articles with unsourced statements | Articles with unsourced statements since August 2008 | Articles with unsourced statements since September 2007