Rootkit

By Contributors to Wikimedia projects

Published: 2003-05-09 · Archived: 2026-04-05 19:39:22 UTC

A **rootkit** is a collection of computer software, typically malicious, designed to enable access to a computer or an area of its software that is not otherwise allowed (for example, to an unauthorized user) and often masks its existence or the existence of other software.[1] The term *rootkit* is a compound of "root" (the traditional name of the privileged account on Unix-like operating systems) and the word "kit" (which refers to the software components that implement the tool).[2] The term "rootkit" has negative connotations through its association with malware.[1]

Rootkit installation can be automated, or an attacker can install it after having obtained root or administrator access.[3] Obtaining this access is a result of direct attack on a system, i.e., exploiting a vulnerability (such as privilege escalation) or a password (obtained by cracking or social engineering tactics like "phishing"). Once installed, it becomes possible to hide the intrusion as well as to maintain privileged access. Full control over a system means that existing software can be modified, including software that might otherwise be used to detect or circumvent it.

Rootkit detection is difficult because a rootkit may be able to subvert the software that is intended to find it. Detection methods include using an alternative and trusted operating system, behavior-based methods, signature scanning, difference scanning, and memory dump analysis. Removal can be complicated or practically impossible, especially in cases where the rootkit resides in the kernel; reinstallation of the operating system may be the only available solution to the problem. When dealing with firmware rootkits, removal may require hardware replacement, or specialized equipment.

The term *rootkit*, *rkit*, or *root kit* originally referred to a maliciously modified set of administrative tools for a Unix-like operating system that granted "root" access.[4] If an intruder could replace the standard administrative tools on a system with a rootkit, the intruder could obtain root access over the system whilst simultaneously concealing these activities from the legitimate system administrator. These first-generation rootkits were trivial to detect by using tools such as Tripwire that had not been compromised to access the same information.[5][6] Lane Davis and Steven Dake wrote the earliest known rootkit in 1990 for Sun Microsystems' SunOS UNIX operating system.[7] In the lecture he gave upon receiving the Turing Award in 1983, Ken Thompson of Bell Labs, one of the creators of Unix, theorized about subverting the C compiler in a Unix distribution and discussed the exploit. The modified compiler would detect attempts to compile the Unix `login` command and generate altered code that would accept not only the user's correct password, but an additional "backdoor" password known to the attacker. Additionally, the compiler would detect attempts to compile a new version of the compiler, and would insert the same exploits into the new compiler. A review of the source code for the `login` command or the updated compiler would not reveal any malicious code.[8] This exploit was equivalent to a rootkit.

The first documented computer virus to target the personal computer, discovered in 1986, used Helix Cloaking techniques to hide itself: the Brain virus intercepted attempts to read the boot sector, and redirected these to elsewhere on the disk, where a copy of the original boot sector was kept.[1] Over time, DOS-virus cloaking methods became more sophisticated. Advanced techniques included hooking low-level disk INT 13H BIOS interrupt calls to hide unauthorized modifications to files.[1]

The first malicious rootkit for the Windows NT operating system appeared in 1999: a trojan called *NTRootkit* created by Greg Hoglund.[9] It was followed by *HackerDefender* in 2003.[1] The first rootkit targeting Mac OS X, WeaponX/Weapox, appeared in 2004[10] while the Stuxnet worm was the first to target programmable logic controllers (PLC).[11]

Lenovo BIOS Rootkit (Lenovo Service Engine) Incident (2015)

[edit]

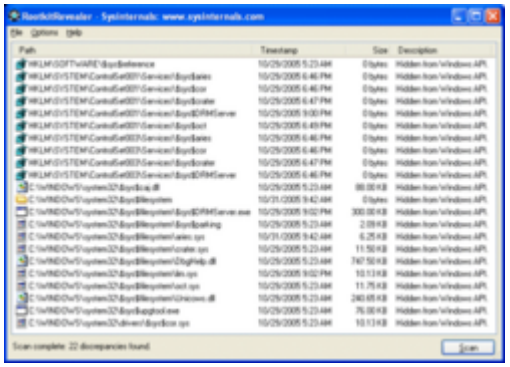
In mid-2015, it was discovered that Lenovo had been shipping certain consumer PCs with firmware that behaved like a built-in rootkit. The feature, called Lenovo Service Engine (LSE), was embedded in the system BIOS and would execute on startup, even before Windows booted. LSE was designed to ensure that Lenovo's system update utility and related pre-installed programs remained installed by automatically reinstalling them if they were removed. Because it resided in firmware, the code was difficult for users to detect or remove; even a clean Windows installation would not eliminate LSE, as it would be reinstalled on the next reboot.

Researchers later discovered that LSE introduced a serious security issue – a vulnerability allowing a privilege escalation attack (via a buffer overflow) to gain administrator-level control. In response, Lenovo released BIOS updates and a removal utility in 2015 to disable and delete the LSE feature. Microsoft also updated its Windows security guidelines to bar such firmware behavior, effectively forcing Lenovo to cease using LSE in new systems. The LSE functionality was removed from subsequent models, and Lenovo urged customers to install the updated firmware to eliminate the risk.[12][13]

Stuxnet, uncovered in 2010, was a highly sophisticated worm developed in a joint U.S.–Israeli intelligence operation targeting Iran's nuclear facilities. It notably included a Windows kernel-mode rootkit that concealed the malware's files and processes, enabling the worm to silently sabotage industrial control systems. Stuxnet is often cited as the first known cyberweapon; it destroyed a significant part of Iran's uranium centrifuges, while remaining difficult to detect.[14][15][16]

Sony BMG copy protection rootkit scandal (2005)

[edit]



Screenshot of RootkitRevealer, showing the files hidden by the Extended Copy Protection rootkit

In 2005, Sony BMG published CDs with copy protection and digital rights management software called Extended Copy Protection, created by software company First 4 Internet. The software included a music player but silently installed a rootkit which limited the user's ability to access the CD.[17] Software engineer Mark Russinovich, who created the rootkit detection tool RootkitRevealer, discovered the rootkit on one of his computers.[1] The ensuing scandal raised the public's awareness of rootkits.[18] To cloak itself, the rootkit hid any file starting with "sys" from the user. Soon after Russinovich's report, malware appeared which took advantage of the existing rootkit on affected systems.[1] One BBC analyst called it a "public relations nightmare."[19] Sony BMG released patches to uninstall the rootkit, but it exposed users to an even more serious vulnerability.[20] The company eventually recalled the CDs. In the United States, a class-action lawsuit was brought against Sony BMG.[21]

Greek wiretapping case (2004–05)

[edit]

The Greek wiretapping case 2004–05, also referred to as Greek Watergate,[22] involved the illegal telephone tapping of more than 100 mobile phones on the Vodafone Greece network belonging mostly to members of the Greek government and top-ranking civil servants. The taps began sometime near the beginning of August 2004 and were removed in March 2005 without discovering the identity of the perpetrators. The intruders installed a rootkit targeting Ericsson's AXE telephone exchange. According to *IEEE Spectrum*, this was "the first time a rootkit has been observed on a special-purpose system, in this case an Ericsson telephone switch."[23] The rootkit was designed to patch the memory of the exchange while it was running, enable wiretapping while disabling audit logs, patch the commands that list active processes and active data blocks, and modify the data block checksum verification command. A "backdoor" allowed an operator with sysadmin status to deactivate the exchange's transaction log, alarms and access commands related to the surveillance capability.[23] The rootkit was discovered after the intruders installed a faulty update, which caused SMS texts to be undelivered, leading to an automated failure report being generated. Ericsson engineers were called in to investigate the fault and discovered the hidden data blocks containing the list of phone numbers being monitored, along with the rootkit and illicit monitoring software.

Modern rootkits do not elevate access,[4] but rather are used to make another software payload undetectable by adding stealth capabilities.[9] Most rootkits are classified as malware, because the payloads they are bundled with are malicious. For example, a payload might covertly steal user passwords, credit card information, computing

resources, or conduct other unauthorized activities. A small number of rootkits may be considered utility applications by their users: for example, a rootkit might cloak a CD-ROM-emulation driver, allowing video game users to defeat anti-piracy measures that require insertion of the original installation media into a physical optical drive to verify that the software was legitimately purchased.

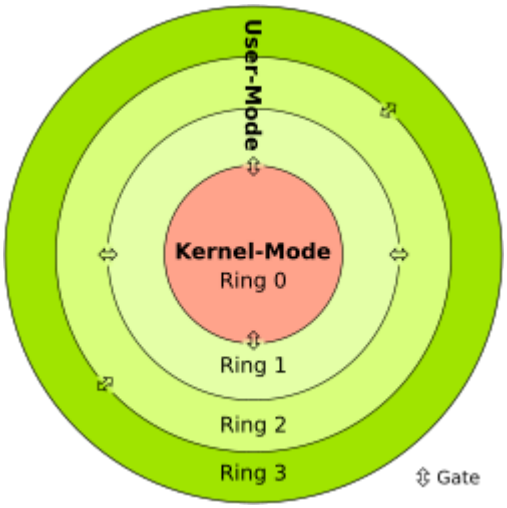
Rootkits and their payloads have many uses:

- Provide an attacker with full access via a backdoor, permitting unauthorized access to, for example, steal or falsify documents. One of the ways to carry this out is to subvert the login mechanism, such as the /bin/login program on Unix-like systems or GINA on Windows. The replacement appears to function normally, but also accepts a secret login combination that allows an attacker direct access to the system with administrative privileges, bypassing standard authentication and authorization mechanisms.
- Conceal other malware, notably password-stealing key loggers and computer viruses.[24]
- Appropriate the compromised machine as a zombie computer for attacks on other computers. (The attack originates from the compromised system or network, instead of the attacker's system.) "Zombie" computers are typically members of large botnets that can–amongst other things–launch denial-of-service attacks, distribute email spam, and conduct click fraud.[25]

In some instances, rootkits provide desired functionality, and may be installed intentionally on behalf of the computer user:

- Detect attacks, for example, in a honeypot.[26]
- Enhance emulation software and security software.[27] Alcohol 120% and Daemon Tools are commercial examples of non-hostile rootkits used to defeat copy-protection mechanisms such as SafeDisc and SecuROM.[28] Kaspersky antivirus software also uses techniques resembling rootkits to protect itself from malicious actions. It loads its own drivers to intercept system activity, and then prevents other processes from doing harm to itself. Its processes are not hidden, but cannot be terminated by standard methods.
- Anti-theft protection: Laptops may have BIOS-based rootkit software that will periodically report to a central authority, allowing the laptop to be monitored, disabled or wiped of information in the event that it is stolen.[29]
- Bypassing Microsoft Product Activation[30]

There are at least five types of rootkit, ranging from those at the lowest level in firmware (with the highest privileges), through to the least privileged user-based variants that operate in Ring 3. Hybrid combinations of these may occur spanning, for example, user mode and kernel mode.[31]



Intel based computer security rings (Note that Ring -1 is not shown.)

User-mode rootkits run in Ring 3, along with other applications as user, rather than low-level system processes.[32] They have a number of possible installation vectors to intercept and modify the standard behavior of application programming interfaces (APIs). Some inject a dynamically linked library (such as a .DLL file on Windows, or a .dylib file on Mac OS X) into other processes, and are thereby able to execute inside any target process to spoof it; others with sufficient privileges simply overwrite the memory of a target application. Injection mechanisms include:[32]

- Use of vendor-supplied application extensions. For example, Windows Explorer has public interfaces that allow third parties to extend its functionality.
- Interception of messages.
- Debuggers.
- Exploitation of security vulnerabilities.
- Function hooking or patching of commonly used APIs, for example, to hide a running process or file that resides on a filesystem.[33]

> ...since user mode applications all run in their own memory space, the rootkit needs to perform this patching in the memory space of every running application. In addition, the rootkit needs to monitor the system for any new applications that execute and patch those programs' memory space before they fully execute.

—Windows Rootkit Overview, Symantec[4]

Kernel-mode rootkits run with the highest operating system privileges (Ring 0) by adding code or replacing portions of the core operating system, including both the kernel and associated device drivers.[*citation needed*] Most operating systems support kernel-mode device drivers, which execute with the same privileges as the operating system itself. 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 has unrestricted security access, but is more difficult to write.[34] The complexity makes bugs common, and any bugs in code operating at the kernel level may seriously impact system stability, leading to discovery of the rootkit.[34] One of the first widely known kernel rootkits was developed for Windows NT 4.0 and released in *Phrack* magazine in

1999 by Greg Hoglund.[35][36] Kernel rootkits can be especially difficult to detect and remove because they operate at the same security level as the operating system itself, and are thus able to intercept or subvert the most trusted operating system operations. Any software, such as antivirus software, running on the compromised system is equally vulnerable.[37] In this situation, no part of the system can be trusted.

A rootkit can modify data structures in the Windows kernel using a method known as *direct kernel object manipulation* (DKOM).[38] This method can be used to hide processes. A kernel mode rootkit can also hook the System Service Descriptor Table (SSDT), or modify the gates between user mode and kernel mode, in order to cloak itself.[4] Similarly for the Linux operating system, a rootkit can modify the *system call table* to subvert kernel functionality.[39][40] It is common that a rootkit creates a hidden, encrypted filesystem in which it can hide other malware or original copies of files it has infected.[41] Operating systems are evolving to counter the threat of kernel-mode rootkits. For example, 64-bit editions of Microsoft Windows now implement mandatory signing of all kernel-level drivers in order to make it more difficult for untrusted code to execute with the highest privileges in a system.[42]

A kernel-mode rootkit variant called a **bootkit** can infect startup code like the Master Boot Record (MBR), Volume Boot Record (VBR), or boot sector, and in this way can be used to attack full disk encryption systems.[43] An example of such an attack on disk encryption is the "evil maid attack", in which an attacker installs a bootkit on an unattended computer. The envisioned scenario is a maid sneaking into the hotel room where the victims left their hardware.[44] The bootkit replaces the legitimate boot loader with one under their control. Typically the malware loader persists through the transition to protected mode when the kernel has loaded, and is thus able to subvert the kernel.[45][46][47] For example, the "Stoned Bootkit" subverts the system by using a compromised boot loader to intercept encryption keys and passwords.[48][*self-published source?*] In 2010, the Alureon rootkit has successfully subverted the requirement for 64-bit kernel-mode driver signing in Windows 7, by modifying the master boot record.[49] Although not malware in the sense of doing something the user doesn't want, certain "Vista Loader" or "Windows Loader" software work in a similar way by injecting an ACPI SLIC (System Licensed Internal Code) table in the RAM-cached version of the BIOS during boot, in order to defeat the Windows Vista and Windows 7 activation process.[*citation needed*] This vector of attack was rendered useless in the (non-server) versions of Windows 8, which use a unique, machine-specific key for each system, that can only be used by that one machine.[50] Many antivirus companies provide free utilities and programs to remove bootkits.

Rootkits have been created as Type II Hypervisors in academia as proofs of concept. By exploiting hardware virtualization features such as Intel VT or AMD-V, this type of rootkit runs in Ring -1 and hosts the target operating system as a virtual machine, thereby enabling the rootkit to intercept hardware calls made by the original operating system.[6] Unlike normal hypervisors, they do not have to load before the operating system, but can load into an operating system before promoting it into a virtual machine.[6] A hypervisor rootkit does not have to make any modifications to the kernel of the target to subvert it; however, that does not mean that it cannot be detected by the guest operating system. For example, timing differences may be detectable in CPU instructions.[6] The "SubVirt" laboratory rootkit, developed jointly by Microsoft and University of Michigan researchers, is an academic example of a virtual-machine–based rootkit (VMBR),[51] while Blue Pill software is another. In 2009, researchers from Microsoft and North Carolina State University demonstrated a hypervisor-layer anti-rootkit

called Hooksafe, which provides generic protection against kernel-mode rootkits.[52] Windows 10 introduced a new feature called "Device Guard", that takes advantage of virtualization to provide independent external protection of an operating system against rootkit-type malware.[53]

Firmware and hardware

[edit]

A firmware rootkit uses device or platform firmware to create a persistent malware image in hardware, such as a router, network card,[54] hard drive, or the system BIOS.[32][55] The rootkit hides in firmware, because firmware is not usually inspected for code integrity. John Heasman demonstrated the viability of firmware rootkits in both ACPI firmware routines[56] and in a PCI expansion card ROM.[57] In October 2008, criminals tampered with European credit-card-reading machines before they were installed. The devices intercepted and transmitted credit card details via a mobile phone network.[58] In March 2009, researchers Alfredo Ortega and Anibal Sacco published details of a BIOS-level Windows rootkit that was able to survive disk replacement and operating system re-installation.[59][60][61] A few months later they learned that some laptops are sold with a legitimate rootkit, known as Absolute CompuTrace or Absolute LoJack for Laptops, preinstalled in many BIOS images. This is an anti-theft technology system that researchers showed can be turned to malicious purposes.[29]

Intel Active Management Technology, part of Intel vPro, implements out-of-band management, giving administrators remote administration, remote management, and remote control of PCs with no involvement of the host processor or BIOS, even when the system is powered off. Remote administration includes remote power-up and power-down, remote reset, redirected boot, console redirection, pre-boot access to BIOS settings, programmable filtering for inbound and outbound network traffic, agent presence checking, out-of-band policy-based alerting, access to system information, such as hardware asset information, persistent event logs, and other information that is stored in dedicated memory (not on the hard drive) where it is accessible even if the OS is down or the PC is powered off. Some of these functions require the deepest level of rootkit, a second non-removable spy computer built around the main computer. Sandy Bridge and future chipsets have "the ability to remotely kill and restore a lost or stolen PC via 3G". Hardware rootkits built into the chipset can help recover stolen computers, remove data, or render them useless, but they also present privacy and security concerns of undetectable spying and redirection by management or hackers who might gain control.

Installation and cloaking

[edit]

Rootkits employ a variety of techniques to gain control of a system; the type of rootkit influences the choice of attack vector. The most common technique leverages security vulnerabilities to achieve surreptitious privilege escalation. Another approach is to use a Trojan horse, deceiving a computer user into trusting the rootkit's installation program as benign—in this case, social engineering convinces a user that the rootkit is beneficial.[34] The installation task is made easier if the principle of least privilege is not applied, since the rootkit then does not have to explicitly request elevated (administrator-level) privileges. Other classes of rootkits can be installed only by someone with physical access to the target system. Some rootkits may also be installed intentionally by the

owner of the system or somebody authorized by the owner, e.g. for the purpose of employee monitoring, rendering such subversive techniques unnecessary.[62] Some malicious rootkit installations are commercially driven, with a pay-per-install (PPI) compensation method typical for distribution.[63][64]

Once installed, a rootkit takes active measures to obscure its presence within the host system through subversion or evasion of standard operating system security tools and application programming interface (APIs) used for diagnosis, scanning, and monitoring.[65] Rootkits achieve this by modifying the behavior of core parts of an operating system through loading code into other processes, the installation or modification of drivers, or kernel modules. Obfuscation techniques include concealing running processes from system-monitoring mechanisms and hiding system files and other configuration data.[66] It is not uncommon for a rootkit to disable the event logging capacity of an operating system, in an attempt to hide evidence of an attack. Rootkits can, in theory, subvert *any* operating system activities.[67] The "perfect rootkit" can be thought of as similar to a "perfect crime": one that nobody realizes has taken place. Rootkits also take a number of measures to ensure their survival against detection and "cleaning" by antivirus software in addition to commonly installing into Ring 0 (kernel-mode), where they have complete access to a system. These include polymorphism (changing so their "signature" is hard to detect), stealth techniques, regeneration, disabling or turning off anti-malware software,[68] and not installing on virtual machines where it may be easier for researchers to discover and analyze them.

The fundamental problem with rootkit detection is that if the operating system has been subverted, particularly by a kernel-level rootkit, it cannot be trusted to find unauthorized modifications to itself or its components.[67] Actions such as requesting a list of running processes, or a list of files in a directory, cannot be trusted to behave as expected. In other words, rootkit detectors that work while running on infected systems are only effective against rootkits that have some defect in their camouflage, or that run with lower user-mode privileges than the detection software in the kernel.[34] As with computer viruses, the detection and elimination of rootkits is an ongoing struggle between both sides of this conflict.[67] Detection can take a number of different approaches, including looking for virus "signatures" (e.g., antivirus software), integrity checking (e.g., digital signatures), difference-based detection (comparison of expected vs. actual results), and behavioral detection (e.g., monitoring CPU usage or network traffic).

For kernel-mode rootkits, detection is considerably more complex, requiring careful scrutiny of the System Call Table to look for hooked functions where the malware may be subverting system behavior,[69] as well as forensic scanning of memory for patterns that indicate hidden processes. Unix rootkit detection offerings include Zeppoo,[70] chkrootkit, rkhunter and OSSEC. For Windows, detection tools include Microsoft Sysinternals RootkitRevealer,[71] Avast Antivirus,[72] Sophos Anti-Rootkit,[73] F-Secure,[74] Radix,[75] GMER,[76] and WindowsSCOPE. Any rootkit detectors that prove effective ultimately contribute to their own ineffectiveness, as malware authors adapt and test their code to escape detection by well-used tools.[Notes 1] Detection by examining storage while the suspect operating system is not operational can miss rootkits not recognised by the checking software, as the rootkit is not active and suspicious behavior is suppressed; conventional anti-malware software running with the rootkit operational may fail if the rootkit hides itself effectively.

Alternative trusted medium

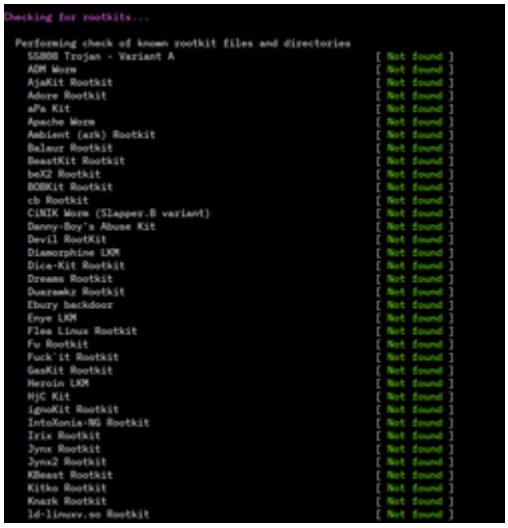
[edit]

The best and most reliable method for operating-system-level rootkit detection is to shut down the computer suspected of infection, and then to check its storage by booting from an alternative trusted medium (e.g., a "rescue" CD-ROM or USB flash drive).[77] The technique is effective because a rootkit cannot actively hide its presence if it is not running.

The behavioral-based approach to detecting rootkits attempts to infer the presence of a rootkit by looking for rootkit-like behavior. For example, by profiling a system, differences in the timing and frequency of API calls or in overall CPU utilization can be attributed to a rootkit. The method is complex and is hampered by a high incidence of false positives. Defective rootkits can sometimes introduce very obvious changes to a system: the Alureon rootkit crashed Windows systems after a security update exposed a design flaw in its code.[78][79] Logs from a packet analyzer, firewall, or intrusion prevention system may present evidence of rootkit behaviour in a networked environment.[31]

Antivirus products rarely catch all viruses in public tests (depending on what is used and to what extent), even though security software vendors incorporate rootkit detection into their products. Should a rootkit attempt to hide during an antivirus scan, a stealth detector may notice; if the rootkit attempts to temporarily unload itself from the system, signature detection (or "fingerprinting") can still find it.[80] This combined approach forces attackers to implement counterattack mechanisms, or "retro" routines, that attempt to terminate antivirus programs. Signature-based detection methods can be effective against well-published rootkits, but less so against specially crafted, custom-root rootkits.[67]

Another method that can detect rootkits compares "trusted" raw data with "tainted" content returned by an API. For example, binaries present on disk can be compared with their copies within operating memory (in some operating systems, the in-memory image should be identical to the on-disk image), or the results returned from file system or Windows Registry APIs can be checked against raw structures on the underlying physical disks[67][81] —however, in the case of the former, some valid differences can be introduced by operating system mechanisms like memory relocation or shimming. A rootkit may detect the presence of such a difference-based scanner or virtual machine (the latter being commonly used to perform forensic analysis), and adjust its behaviour so that no differences can be detected. Difference-based detection was used by Russinovich's *RootkitRevealer* tool to find the Sony DRM rootkit.[1]



The rkhunter utility uses SHA-1 hashes to verify the integrity of system files.

Code signing uses public-key infrastructure to check if a file has been modified since being digitally signed by its publisher. Alternatively, a system owner or administrator can use a cryptographic hash function to compute a "fingerprint" at installation time that can help to detect subsequent unauthorized changes to on-disk code libraries.[82] However, unsophisticated schemes check only whether the code has been modified since installation time; subversion prior to that time is not detectable. The fingerprint must be re-established each time changes are made to the system: for example, after installing security updates or a service pack. The hash function creates a *message digest*, a relatively short code calculated from each bit in the file using an algorithm that creates large changes in the message digest with even smaller changes to the original file. By recalculating and comparing the message digest of the installed files at regular intervals against a trusted list of message digests, changes in the system can be detected and monitored—as long as the original baseline was created before the malware was added.

More-sophisticated rootkits are able to subvert the verification process by presenting an unmodified copy of the file for inspection, or by making code modifications only in memory, reconfiguration registers, which are later compared to a white list of expected values.[83] The code that performs hash, compare, or extend operations must also be protected—in this context, the notion of an *immutable root-of-trust* holds that the very first code to measure security properties of a system must itself be trusted to ensure that a rootkit or bootkit does not compromise the system at its most fundamental level.[84]

Forcing a complete dump of virtual memory will capture an active rootkit (or a kernel dump in the case of a kernel-mode rootkit), allowing offline forensic analysis to be performed with a debugger against the resulting dump file, without the rootkit being able to take any measures to cloak itself. This technique is highly specialized, and may require access to non-public source code or debugging symbols. Memory dumps initiated by the operating system cannot always be used to detect a hypervisor-based rootkit, which is able to intercept and subvert the lowest-level attempts to read memory[6]—a hardware device, such as one that implements a non-maskable interrupt, may be required to dump memory in this scenario.[85][86] Virtual machines also make it easier to analyze the memory of a compromised machine from the underlying hypervisor, so some rootkits will avoid infecting virtual machines for this reason.

Manual removal of a rootkit is often extremely difficult for a typical computer user,[32] but a number of security-software vendors offer tools to automatically detect and remove some rootkits, typically as part of an antivirus suite. As of 2005, Microsoft's monthly Windows Malicious Software Removal Tool is able to detect and remove some classes of rootkits.[87][88] Also, Windows Defender Offline can remove rootkits, as it runs from a trusted environment before the operating system starts.[89] Some antivirus scanners can bypass file system APIs, which are vulnerable to manipulation by a rootkit. Instead, they access raw file system structures directly, and use this information to validate the results from the system APIs to identify any differences that may be caused by a rootkit.[Notes 2][90][91][92][93] There are experts who believe that the only reliable way to remove them is to re-install the operating system from trusted media.[94][95] This is because antivirus and malware removal tools running on an untrusted system may be ineffective against well-written kernel-mode rootkits. Booting an alternative operating system from trusted media can allow an infected system volume to be mounted and potentially safely cleaned and critical data to be copied off—or, alternatively, a forensic examination performed.[31] Lightweight operating systems such as Windows PE, Windows Recovery Console, Windows Recovery Environment, BartPE, or Live Distros can be used for this purpose, allowing the system to be "cleaned". Even if the type and nature of a rootkit is known, manual repair may be impractical, while re-installing the operating system and applications is safer, simpler and quicker.[94]

System hardening represents one of the first layers of defence against a rootkit, to prevent it from being able to be installed in the first place.[96] Applying security patches, implementing the principle of least privilege, reducing the attack surface and installing antivirus software are some standard security best practices that are effective against all classes of malware.[97] New secure boot specifications like UEFI have been designed to address the threat of bootkits, but even these are vulnerable if the security features they offer are not utilized.[55] For server systems, remote server attestation using technologies such as Intel Trusted Execution Technology (TXT) provide a way of verifying that servers remain in a known good state. For example, Microsoft Bitlocker's encryption of data-at-rest verifies that servers are in a known "good state" on bootup. PrivateCore vCage is a software offering that secures data-in-use (memory) to avoid bootkits and rootkits by verifying servers are in a known "good" state on bootup. The PrivateCore implementation works in concert with Intel TXT and locks down server system interfaces to avoid potential bootkits and rootkits.

Another defense mechanism called the Virtual Wall (VTW) approach, serves as a lightweight hypervisor with rootkit detection and event tracing capabilities. In normal operation (guest mode), Linux runs, and when a loaded LKM violates security policies, the system switches to host mode. The VTW in host mode detects, traces, and classifies rootkit events based on memory access control and event injection mechanisms. Experimental results demonstrate the VTW's effectiveness in timely detection and defense against kernel rootkits with minimal CPU overhead (less than 2%). The VTW is compared favorably to other defense schemes, emphasizing its simplicity in implementation and potential performance gains on Linux servers.[98]

- Computer security conference
- Host-based intrusion detection system
- Man-in-the-middle attack
- *The Rootkit Arsenal: Escape and Evasion in the Dark Corners of the System*

1. ^ The process name of Sysinternals RootkitRevealer was targeted by malware; in an attempt to counter this countermeasure, the tool now uses a randomly generated process name.
2. ^ In theory, a sufficiently sophisticated kernel-level rootkit could subvert read operations against raw file system data structures as well, so that they match the results returned by APIs.

1. ^ Jump up to: **a** **b** **c** **d** **e** **f** **g** **h** *"Rootkits, Part 1 of 3: The Growing Threat" (PDF). McAfee. 2006-04-17. Archived from the original (PDF) on 2006-08-23.*
2. ^ *Evancich, N.; Li, J. (2016-08-23). "6.2.3 Rootkits". In Colbert, Edward J. M.; Kott, Alexander (eds.). Cyber-security of SCADA and Other Industrial Control Systems. Springer. p. 100. ISBN 9783319321257 – via Google Books.*
3. ^ *"What is Rootkit – Definition and Explanation". www.kaspersky.com. 2021-04-09. Retrieved 2021-11-13.*
4. ^ Jump up to: **a** **b** **c** **d** *"Windows Rootkit Overview" (PDF). Symantec. 2006-03-26. Archived from the original (PDF) on 2010-12-14. Retrieved 2010-08-17.*
5. ^ *Sparks, Sherri; Butler, Jamie (2005-08-01). "Raising The Bar For Windows Rootkit Detection". Phrack. **0xb** (x3d).*
6. ^ Jump up to: **a** **b** **c** **d** **e** *Myers, Michael; Youndt, Stephen (2007-08-07). An Introduction to Hardware-Assisted Virtual Machine (HVM) Rootkits (Report). Crucial Security. CiteSeerX 10.1.1.90.8832.*
7. ^ *Andrew Hay; Daniel Cid; Rory Bray (2008). OSSEC Host-Based Intrusion Detection Guide. Syngress. p. 276. ISBN 978-1-59749-240-9 – via Google Books.*
8. ^ *Thompson, Ken (August 1984). "Reflections on Trusting Trust" (PDF). Communications of the ACM. **27** (8): 761. doi:10.1145/358198.358210. Archived (PDF) from the original on 2007-09-24. Retrieved 2010-06-09.*
9. ^ Jump up to: **a** **b** *Greg Hoglund; James Butler (2006). Rootkits: Subverting the Windows kernel. Addison-Wesley. p. 4. ISBN 978-0-321-29431-9 – via Google Books.*
10. ^ *Ferrie, Peter (2005-07-01). "Got [Mac]Root?" (PDF). Virus Bulletin. Archived (PDF) from the original on 2022-05-28. Retrieved 2025-10-03.*
11. ^ *"Stuxnet Introduces the First Known Rootkit for Industrial Control Systems". Symantec. 2010-08-06. Archived from the original on August 20, 2010. Retrieved 2010-12-04.*
12. ^ *"CAUGHT: Lenovo crams unremovable crapware into Windows laptops – by hiding it in the BIOS". Archived from the original on 2025-09-06. Retrieved 2025-10-26.*
13. ^ *Hern, Alex (2015-08-14). "Lenovo does it again as LSE component removed after security fears". The Guardian. ISSN 0261-3077. Retrieved 2025-10-26.*
14. ^ *"Is Stuxnet the 'best' malware ever?". Computerworld. Retrieved 2025-10-26.*
15. ^ *"The Real Story of Stuxnet - IEEE Spectrum". spectrum.ieee.org. Retrieved 2025-10-26.*
16. ^ *Weinberger, Sharon (2011-06-01). "Computer security: Is this the start of cyberwarfare?". Nature. **474** (7350): 142–145. doi:10.1038/474142a. ISSN 1476-4687. PMID 21654779.*
17. ^ *"Spyware Detail: XCP.Sony.Rootkit". Computer Associates. 2005-11-05. Archived from the original on 2010-08-18. Retrieved 2010-08-19.*
18. ^ *Russinovich, Mark (2005-10-31). "Sony, Rootkits and Digital Rights Management Gone Too Far". TechNet Blogs. Microsoft. Archived from the original on 2016-01-01. Retrieved 2010-08-16.*
19. ^ *"Sony's long-term rootkit CD woes". BBC News. 2005-11-21. Archived from the original on 2026-02-20. Retrieved 2026-02-20.*

20. ^ *Felton, Ed (2005-11-15). "Sony's Web-Based Uninstaller Opens a Big Security Hole; Sony to Recall Discs".*
21. ^ *Knight, Will (2005-11-11). "Sony BMG sued over cloaking software on music CD". New Scientist. Archived from the original on 2011-01-15. Retrieved 2010-11-21.*
22. ^ *Kyriakidou, Dina (March 2, 2006). ""Greek Watergate" Scandal Sends Political Shockwaves". Reuters. Retrieved 2007-11-24.* [dead link]
23. ^ Jump up to: **a** **b** *Vassilis Prevelakis; Diomidis Spinellis (July 2007). "The Athens Affair". Archived from the original on August 1, 2009.*
24. ^ *Russinovich, Mark (June 2005). "Unearthing Root Kits". Windows IT Pro. Archived from the original on 2005-11-03. Retrieved 2026-02-21.*
25. ^ *Marks, Joseph (July 1, 2021). "The Cybersecurity 202: DOJ's future is in disrupting hackers, not just indicting them". The Washington Post. Archived from the original on February 7, 2022. Retrieved July 24, 2021.*
26. ^ *Steve Hanna (September 2007). "Using Rootkit Technology for Honeypot-Based Malware Detection" (PDF). CCEID Meeting.*
27. ^ *Russinovich, Mark (6 February 2006). "Using Rootkits to Defeat Digital Rights Management". Winternals. SysInternals. Archived from the original on 14 August 2006. Retrieved 2006-08-13.*
28. ^ *"Symantec Releases Update for its Own Rootkit". HWM (March): 89. 2006 – via Google Books.*
29. ^ Jump up to: **a** **b** *Ortega, Alfredo; Sacco, Anibal (2009-07-24). Deactivate the Rootkit: Attacks on BIOS anti-theft technologies (PDF). Black Hat USA 2009 (PDF). Boston, MA: Core Security Technologies. Archived (PDF) from the original on 2014-10-16. Retrieved 2014-06-12.*
30. ^ *Kleissner, Peter (2009-09-02). "Stoned Bootkit: The Rise of MBR Rootkits & Bootkits in the Wild" (PDF). Archived from the original (PDF) on 2011-07-16. Retrieved 2010-11-23.*
31. ^ Jump up to: **a** **b** **c** *Anson, Steve; Bunting, Steve (2007). Mastering Windows Network Forensics and Investigation. John Wiley and Sons. pp. 73–74. ISBN 978-0-470-09762-5.*
32. ^ Jump up to: **a** **b** **c** **d** *"Rootkits Part 2: A Technical Primer" (PDF). McAfee. 2007-04-03. Archived from the original (PDF) on 2008-12-05. Retrieved 2010-08-17.*
33. ^ *Kdm. "NTIllusion: A portable Win32 userland rootkit". Phrack. **62** (12). Archived from the original on 2026-02-20. Retrieved 2026-02-21.*
34. ^ Jump up to: **a** **b** **c** **d** *"Understanding Anti-Malware Technologies" (PDF). Microsoft. 2007-02-21. Archived from the original (PDF) on 2010-09-11. Retrieved 2010-08-17.*
35. ^ *Hoglund, Greg (1999-09-09). "A *REAL* NT Rootkit, Patching the NT Kernel". Phrack. **9** (55). Archived from the original on 2026-02-20. Retrieved 2026-02-21.*
36. ^ *Chuvakin, Anton (2003-02-02). An Overview of Unix Rootkits (PDF) (Report). Chantilly, Virginia: iDEFENSE. Archived from the original (PDF) on 2011-07-25. Retrieved 2010-11-21.*
37. ^ *Butler, James; Sparks, Sherri (2005-11-16). "Windows Rootkits of 2005, Part Two". Symantec Connect. Symantec. Retrieved 2010-11-13.*
38. ^ *Butler, James; Sparks, Sherri (2005-11-03). "Windows Rootkits of 2005, Part One". Symantec Connect. Symantec. Archived from the original on 2021-01-21. Retrieved 2010-11-12.*
39. ^ *Burdach, Mariusz (2004-11-17). "Detecting Rootkits And Kernel-level Compromises In Linux". Symantec. Archived from the original on 2020-08-10. Retrieved 2010-11-23.*

40. ^ *Osborne, Charlie (September 17, 2019). "Skidmap malware buries into the kernel to hide illicit cryptocurrency mining". ZDNet. Archived from the original on July 25, 2021. Retrieved July 24, 2021.*
41. ^ *Marco Giuliani (11 April 2011). "ZeroAccess – An Advanced Kernel Mode Rootkit" (PDF). Webroot Software. Archived (PDF) from the original on 25 August 2011. Retrieved 10 August 2011.*
42. ^ *"Driver Signing Requirements for Windows". Microsoft. 2017-01-06. Archived from the original on 2026-02-20. Retrieved 2026-02-20.*
43. ^ *Salter, Jim (July 31, 2020). "Red Hat and CentOS systems aren't booting due to BootHole patches". Ars Technica. Retrieved July 24, 2021.*
44. ^ *Schneier, Bruce (2009-10-23). "'Evil Maid' Attacks on Encrypted Hard Drives". Archived from the original on 2026-02-20. Retrieved 2026-02-20.*
45. ^ *Soeder, Derek; Permeh, Ryan (2007-05-09). "Bootroot". eEye Digital Security. Archived from the original on 2013-08-17. Retrieved 2010-11-23.*
46. ^ *Kumar, Nitin; Kumar, Vipin (2007). Vbootkit: Compromising Windows Vista Security (PDF). Black Hat Europe 2007.*
47. ^ *"BOOT KIT: Custom boot sector based Windows 2000/XP/2003 Subversion". NVlabs. 2007-02-04. Archived from the original on June 10, 2010. Retrieved 2010-11-21.*
48. ^ *Kleissner, Peter (2013-03-16). "Stoned Bootkit". Peter Kleissner. Archived from the original on 2014-12-09. Retrieved 2026-02-20.* [*self-published source*]
49. ^ *Goodin, Dan (2010-11-16). "World's Most Advanced Rootkit Penetrates 64-bit Windows". The Register. Archived from the original on 2010-11-21. Retrieved 2010-11-22.*
50. ^ *Francisco, Neil McAllister in San. "Microsoft tightens grip on OEM Windows 8 licensing". www.theregister.com. Archived from the original on 2021-10-08. Retrieved 2021-10-08.*
51. ^ *King, Samuel T.; Chen, Peter M.; Wang, Yi-Min; Verbowski, Chad; Wang, Helen J.; Lorch, Jacob R. (2006-04-03). "SubVirt: Implementing malware with virtual machines" (PDF). 2006 IEEE Symposium on Security and Privacy (S&P'06). Institute of Electrical and Electronics Engineers. pp. 14 pp.-327. doi:10.1109/SP.2006.38. ISBN 0-7695-2574-1. S2CID 1349303. Archived (PDF) from the original on 2008-12-07. Retrieved 2008-09-15.*
52. ^ *Wang, Zhi; Jiang, Xuxian; Cui, Weidong; Ning, Peng (2009-08-11). "Countering Kernel Rootkits with Lightweight Hook Protection" (PDF). In Al-Shaer, Ehab (General Chair) (ed.). Proceedings of the 16th ACM Conference on Computer and Communications Security. CCS 2009: 16th ACM Conference on Computer and Communications Security. Jha, Somesh; Keromytis, Angelos D. (Program Chairs). New York: ACM New York. doi:10.1145/1653662.1653728. ISBN 978-1-60558-894-0. Archived (PDF) from the original on 2009-12-29. Retrieved 2009-11-11.*
53. ^ *"Device Guard is the combination of Windows Defender Application Control and virtualization-based protection of code integrity (Windows 10)". 11 July 2023.*
54. ^ *Delugré, Guillaume (2010-11-21). Reversing the Broacom NetExtreme's Firmware (PDF). hack.lu. Sogeti. Archived from the original (PDF) on 2012-04-25. Retrieved 2010-11-25.*
55. ^ Jump up to: **a** **b** *"Hacking Team Uses UEFI BIOS Rootkit to Keep RCS 9 Agent in Target Systems - TrendLabs Security Intelligence Blog". 2015-07-13. Archived from the original on 2015-07-23. Retrieved 2015-07-15.*
56. ^ *Heasman, John (2006-01-25). Implementing and Detecting an ACPI BIOS Rootkit (PDF). Black Hat Federal 2006. NGS Consulting. Archived (PDF) from the original on 2011-02-27. Retrieved 2010-11-21.*

57. ^ *Heasman, John (2006-11-15). "Implementing and Detecting a PCI Rootkit" (PDF). Next Generation Security Software. CiteSeerX: 10.1.1.89.7305. Retrieved 2010-11-13.*
58. ^ *Modine, Austin (2008-10-10). "Organized crime tampers with European card swipe devices: Customer data beamed overseas". The Register. Situation Publishing. Archived from the original on 2008-10-13. Retrieved 2008-10-13.*
59. ^ *Sacco, Anibal; Ortéga, Alfredo (2009). Persistent BIOS infection (PDF). CanSecWest 2009. Core Security Technologies. Archived from the original (PDF) on 2011-07-08. Retrieved 2010-11-21.*
60. ^ *Goodin, Dan (2009-03-24). "Newfangled rootkits survive hard disk wiping". The Register. Situation Publishing. Retrieved 2009-03-25.*
61. ^ *Sacco, Anibal; Ortéga, Alfredo (2009-06-01). "Persistent BIOS Infection: The Early Bird Catches the Worm". Phrack. **66** (7). Archived from the original on 2026-02-20. Retrieved 2026-02-20.*
62. ^ *Ric Vieler (2007). Professional Rootkits. John Wiley & Sons. p. 244. ISBN 9780470149546.*
63. ^ *Matrosov, Aleksandr; Rodionov, Eugene (2010-06-25). "TDL3: The Rootkit of All Evil?" (PDF). Moscow: ESET. p. 3. Archived from the original (PDF) on 2011-05-13. Retrieved 2010-08-17.*
64. ^ *Matrosov, Aleksandr; Rodionov, Eugene (2011-06-27). "The Evolution of TDL: Conquering x64" (PDF). ESET. Archived from the original (PDF) on 2015-07-29. Retrieved 2011-08-08.*
65. ^ *Gatlan, Sergiu (May 6, 2021). "New Moriya rootkit used in the wild to backdoor Windows systems". Bleeping Computer. Retrieved July 24, 2021.*
66. ^ *Brumley, David (1999-11-16), "Invisible Intruders: rootkits in practice" (pdf), Login, USENIX Association, pp. 27–29, retrieved 2007-08-27*[dead link]
67. ^ Jump up to: ***a*** ***b*** ***c*** ***d*** ***e*** *Davis, Michael A.; Bodmer, Sean; LeMasters, Aaron (2009-09-03). "Chapter 10: Rootkit Detection" (PDF). Hacking Exposed Malware & Rootkits: Malware & rootkits security secrets & solutions. New York: McGraw Hill Professional. ISBN 978-0-07-159118-8. Archived from the original (PDF) on 2012-03-08. Retrieved 2010-08-14.*
68. ^ *Trlokom (2006-07-05). "Defeating Rootkits and Keyloggers" (PDF). Trlokom. Archived from the original (PDF) on 2011-07-17. Retrieved 2010-08-17.*
69. ^ *Dai Zovi, Dino (2011). "Kernel Rootkits". Archived from the original on September 10, 2012. Retrieved 13 Sep 2012.*
70. ^ *"Zeppoo". SourceForge. 18 July 2009. Retrieved 8 August 2011.*
71. ^ *Cogswell, Bryce; Russinovich, Mark (2006-11-01). "RootkitRevealer v1.71". Microsoft. Archived from the original on 2017-07-01. Retrieved 2010-11-13.*
72. ^ *"Rootkit & Anti-rootkit". Retrieved 13 September 2017.*
73. ^ *"Sophos Anti-Rootkit". Sophos. Archived from the original on 2012-11-19. Retrieved 2026-02-20.*
74. ^ *"BlackLight". F-Secure. Archived from the original on 2011-01-01. Retrieved 2026-02-21.*
75. ^ *"Radix Anti-Rootkit". usec.at. Archived from the original on 2022-07-03. Retrieved 2026-02-21.*
76. ^ *"GMER". Archived from the original on 2026-02-16. Retrieved 2026-02-21.*
77. ^ *Harriman, Josh (2007-10-19). "A Testing Methodology for Rootkit Removal Effectiveness" (PDF). Dublin, Ireland: Symantec Security Response. Archived from the original (PDF) on 2009-10-07. Retrieved 2010-08-17.*
78. ^ *Cuibotariu, Mircea (2010-02-12). "Tidserv and MS10-015". Symantec. Retrieved 2010-08-19.*
79. ^ *"Restart Issues After Installing MS10-015". Microsoft. 2010-02-11. Retrieved 2010-10-05.*

80. ^ *Steinberg, Joseph (June 9, 2021). "What You Need To Know About Keyloggers". bestantivirus.com. Archived from the original on June 4, 2023. Retrieved July 24, 2021.*
81. ^ *"Strider GhostBuster Rootkit Detection". Microsoft Research. 2010-01-28. Archived from the original on 2016-07-07. Retrieved 2026-02-21.*
82. ^ *"Signing and Checking Code with Authenticode". Microsoft. Archived from the original on 2008-12-29. Retrieved 2008-09-15.*
83. ^ *"Stopping Rootkits at the Network Edge" (PDF). Beaverton, Oregon: Trusted Computing Group. January 2017. Retrieved 2008-07-11.*
84. ^ *"TCG PC Specific Implementation Specification, Version 1.1" (PDF). Trusted Computing Group. 2003-08-18. Archived (PDF) from the original on 2011-09-28. Retrieved 2010-11-22.*
85. ^ *"How to generate a complete crash dump file or a kernel crash dump file by using an NMI on a Windows-based system". Microsoft. Archived from the original on 2015-03-24. Retrieved 2010-11-13.*
86. ^ *Seshadri, Arvind; et al. (2005). "Pioneer". Proceedings of the twentieth ACM symposium on Operating systems principles. Carnegie Mellon University. pp. 1–16. doi:10.1145/1095810.1095812. ISBN 1595930795. S2CID 9960430.*
87. ^ *Dillard, Kurt (2005-08-03). "Rootkit battle: Rootkit Revealer vs. Hacker Defender". Archived from the original on 2014-02-13. Retrieved 2026-02-21.*
88. ^ *"The Microsoft Windows Malicious Software Removal Tool helps remove specific, prevalent malicious software from computers that are running Windows 7, Windows Vista, Windows Server 2003, Windows Server 2008, or Windows XP". Microsoft. 2010-09-14. Archived from the original on 2016-12-30. Retrieved 2016-05-16.*
89. ^ *Bettany, Andrew; Halsey, Mike (2017). Windows Virus and Malware Troubleshooting. Apress. p. 17. ISBN 9781484226070 – via Google Books.*
90. ^ *Hultquist, Steve (2007-04-30). "Rootkits: The next big enterprise threat?". InfoWorld. Archived from the original on 2015-09-26. Retrieved 2010-11-21.*
91. ^ *"Security Watch: Rootkits for fun and profit". CNET Reviews. 2007-01-19. Archived from the original on 2012-10-08. Retrieved 2009-04-07.*
92. ^ *Bort, Julie (2007-09-29). "Six ways to fight back against botnets". PCWorld. San Francisco: PCWorld Communications. Archived from the original on 2012-10-11. Retrieved 2009-04-07.*
93. ^ *Hoang, Mimi (2006-11-02). "Handling Today's Tough Security Threats: Rootkits". Symantec Connect. Symantec. Archived from the original on 2021-07-26. Retrieved 2010-11-21.*
94. ^ Jump up to: ***a*** ***b*** *Danseglio, Mike; Bailey, Tony (2005-10-06). "Rootkits: The Obscure Hacker Attack". Microsoft.*
95. ^ *Messmer, Ellen (2006-08-26). "Experts Divided Over Rootkit Detection and Removal". NetworkWorld.com. Framingham, Mass.: IDG. Archived from the original on 2024-11-02. Retrieved 2010-08-15.*
96. ^ *Skoudis, Ed; Zeltser, Lenny (2004). Malware: Fighting Malicious Code. Prentice Hall PTR. p. 335. ISBN 978-0-13-101405-3.*
97. ^ *Hannel, Jeromey (2003-01-23). "Linux RootKits For Beginners - From Prevention to Removal". SANS Institute. Archived from the original (PDF) on October 24, 2010. Retrieved 2010-11-22.*
98. ^ *Li, Yong-Gang; Chung, Yeh-Ching; Hwang, Kai; Li, Yue-Jin (2021). "Virtual Wall: Filtering Rootkit Attacks to Protect Linux Kernel Functions". IEEE Transactions on Computers. **70** (10): 1640–1653.*

Bibcode:2021ITCmp..70.1640L. doi:10.1109/TC.2020.3022023. S2CID 226480878.

- *Blunden, Bill (2009). The Rootkit Arsenal: Escape and Evasion in the Dark Corners of the System. Wordware. ISBN 978-1-59822-061-2.*
- *Hoglund, Greg; Butler, James (2005). Rootkits: Subverting the Windows Kernel. Addison-Wesley Professional. ISBN 978-0-321-29431-9.*
- *Grampp, F. T.; Morris, Robert H. Sr. (October 1984). "The UNIX System: UNIX Operating System Security". AT&T Bell Laboratories Technical Journal. **62** (8): 1649–1672. Bibcode:1984BSTJ...63.1649G. doi:10.1002/j.1538-7305.1984.tb00058.x. S2CID 26877484.*
- *Kong, Joseph (2007). Designing BSD Rootkits. No Starch Press. ISBN 978-1-59327-142-8.*
- *Veiler, Ric (2007). Professional Rootkits. Wrox. ISBN 978-0-470-10154-4.*

- Media related to Rootkits at Wikimedia Commons
