The Root of All Evil?
- Rootkits Revealed

Are rootkits the root of all evil, or just another branch on the threat tree? What you need to know about the rootkit threat.

David Harley BA CISSP FBCS CITP
Andrew Lee CISSP

Table of Contents

Introduction

Public awareness of rootkits has risen in recent years, but as with worms, viruses and other forms of malicious software (malware), the term rootkit is applied unselectively to a range of technologies and has attracted a number of not-very-compatible definitions. While several of these technologies and definitions are explored in this paper, our intention is to clarify common usages, not to supply a single "authoritative" definition. There are, however, some brief definitions in the glossary.

Rootkits are in danger of becoming the latest in a long line of poorly understood threats to be hyped as the "End of Computing as We Know It". Having attracted descriptions [1] such as "the most pernicious and sophisticated form of attack which currently can be made against a Windows system", they have acquired some of the superstitious dread that terms like "stealth" and "polymorphic" inspired earlier in the history of malicious software. Indeed, the concepts of rootkits and stealth (or what is now often referred to as stealthware) are closely related and overlapping, if not synonymous.

This paper aims to assess the realities of the rootkit threat, and to examine the state of the solutions available.

> "It's easy to see why the rootkit concept is so worrying. Software that uses stealth techniques is designed to be invisible ..."

It's easy to see why the rootkit concept is so worrying. Software that uses stealth techniques is designed to be invisible to anti-virus software, other security software, the operating system and file system. Although to some extent rootkits pose a unique set of challenges to the security industry, the technologies are evolving on both sides of the malware war, and while rootkits were, until fairly recently, a specialist security preoccupation — mainly in the UNIX/Linux communities — stealth is nothing new to the anti-virus industry.

Got Root?

On a UNIX based system, the most privileged user has the name "root": this user has ultimate power on the system, and is therefore the most desirable account for an attacker to compromise. "The root" or "root directory" also refers to the first directory in the UNIX file-system structure: that is, the single top-level directory in a conventional directory tree. (Why do directory trees grow downwards, we wonder?) The root directory, usually referred to with a single forward slash "/" is the directory (roughly equivalent to "C:\" on a DOS or Windows PC) through which all others can be accessed. Ordinary, unprivileged system users would normally be unable to alter files in this directory, or outside their own home directory. So "rooting" a system refers to the compromise of the root account, thereby gaining "root" access to the system and all its files and directories.

Rootkits and Stealth

Stealth technology was sometimes defined in the earlier days of malware detection along the following lines [2; 3]

- Negative Stealth (level -1): infection entails damage to the functionality of the infected object that makes discovery of an infection inevitable.
- Non-Stealth (level 0): no specific measures are taken to conceal the presence of the infection.
- Elementary Stealth (level 1): there is no characteristic display to draw attention. Basic anti-detection steps are taken, such as preserving date and time stamps.
- Intermediate Stealth (level 2): an image or partial image of the object in its pre-infected state is maintained to "show" to the system, so as to help conceal the "footprint" of the virus.
- Advanced Stealth (level 3): concealment methods are used aimed specifically to conceal infection from security software.

While this classification scheme is not much used outside the anti-virus community these days, it remains valid, not only in the context of viruses, but in terms of other stealthy malware including rootkits. If we take into account the fact that such mechanisms been known and dealt with perfectly adequately by anti-virus software for many years, we may be less alarmed at assertions that rootkit technology is now so advanced that only special tools can detect it, and that the only way of dealing with it is to reformat an infected drive and reinstall the operating system.

Potentially, anti-virus software can detect rootkits as easily as it does viruses, certainly before it gets the chance to install, but also, in many circumstances, after they are installed. That doesn't, of course, mean there is no rootkit problem, any more than it

"Potentially, anti-virus software can detect rootkits as easily as it does viruses"

means there is no virus problem. Nor does it mean that all anti-virus programs deal with the problem equally well, nor deal with all types of rootkits with the same success. The problem is, however, manageable: the sky is not falling.

Rootkit Definitions

According to Hoglund [4], a rootkit is "a set of programs and code that allows a permanent or consistent, undetectable presence on a computer". This is a reasonable top-level definition, but it does not really lessen the confusion between old-style "stealthing", newer "stealthkits", and rootkits.

Let's consider, then, a basic definition [5] of a (malicious) toolkit: "A software package that contains scripts, programs, or autonomous agents that exploit vulnerabilities." A rootkit is a kind of toolkit usually associated with the attempt to gain privileged access or to maintain that access by concealing the fact that the system has been compromised. It might therefore be defined as a set of malicious programmatic tools that enable an intruder to conceal the fact that a system has been compromised, and to continue to make use of that compromise. In reality, the situation is rather more complicated, but before we can go into that, we need to understand something about user account management and user privilege. Having done so, we can consider specific toolkits in specific environments.

For the moment, however, a useful working definition of a rootkit might be a form of toolkit installed onto a compromised system in order to:

- maintain privileged access and control
- allow the individual and/or software to make use of that access in whatever way he chooses
- conceal or restrict access to objects or processes such as:
- Processes
- Threads
- Files
- Folders/Directories/Subdirectories
- Registry Entries
- Handles
- Open Ports

Note, by the way, that these definitions do not presuppose:

- "intrusion" – that is, unauthorized access
- malicious action or intent
- "rooting" (Gaining an inappropriate and unauthorized level of access and privilege – we explore this concept in more detail below.)

This evasion of the assumption of malicious hacking is actually quite defensible. Most multi-user operating systems apply some form of concealment and/or restricted access to sensitive data and critical systems files on the part of unprivileged users. This is done for entirely legitimate security reasons: e.g., to prevent end users from accessing data to which they are not entitled, or to prevent them deleting or modifying files and programs and thus damaging system or data integrity. Security in this sense does not presuppose malice on the part of the end user, and could easily fall within a slightly modified version of the stealth classification model quoted above. However, this definition covers use of more advanced stealth/rootkit-like techniques, also for legitimate reasons (see the account of the Sony "rootkit" later in this paper).

60-Second Guide to Account Management

Even the most humble contemporary PC has resources and processing power equivalent or superior to many of the mainframes and minicomputers of yesteryear. What may be less obvious is that today's personal computers have evolved from stand-alone, single-user systems with networking capabilities barely equivalent to those of a dumb terminal, to server class machines. Modern PCs and operating systems are capable of supporting not only multiple processes and user areas, but multiple concurrent user processes, though most desktops and laptops are generally used by one person, and therefore one account, at a time, unless they're in use to provide an online service to remote users. (Literally, to act as a server at the same time as being used as a workstation.)

A user's account not only allows them access to the system, but also defines what they can do on that system. Most users have limited administration and access rights or privileges. Users may be allocated or deprived of privileges according to their individual status and duties, but also according to their membership of particular groups. Groups may be defined on the basis of shared interest or duties, geographical, host or network/sub-network location, and so on. Administrators generally have rights to install or modify software, access other user's accounts and so forth that standard users don't have. Their rights may also be constrained according to a hierarchy: for instance, an administrator may have full privileges on some servers or domains, but not on others. The "root" account normally has all administrator (superuser) privileges on UNIX/Linux and non-UNIX systems.

The default administrator account on Windows systems is more-or-less equivalent to the UNIX root account. While Macintosh OS X systems are based on BSD UNIX, the account system as accessed via the graphical interface is rather different to standard UNIX, but the principle is the same.

User Privileges and Rooting

"Rooting" is a term applied to gaining root access and therefore complete control over a UNIX (or Linux) system. This may be done by direct privilege escalation: that is, exploitation of system vulnerabilities to achieve a higher level of privilege. However, it can also be done by virus action, as reported by Cohen [6] in the early days of malware research. The use of the term 'rooting' sounds odd in the PC/Windows context, where the account name root has no particular significance. Still, it can be used as a generic term for gaining privileged access to both UNIX/Linux and non-UNIX systems.

> "A rootkit enables the intruder to maintain and exploit an undetected foothold on the system"

Rootkit objectives:

The main objective of a rootkit is not necessarily to "root" the host system, i.e. to break into it, though it may well include programs designed to aid gaining such administrative access. Rather, its objective is to enable the intruder to maintain and exploit an undetected foothold on the system.

However, some sources distinguish between rootkits and stealthkits. By this distinction, a rootkit might be defined as a toolkit with functionality that includes tools for rooting a system. At least, so some have argued [7]. This paper, however, addresses a number of rootkit types and rootkit components, rather than clinging to a single purist view of what is and isn't a rootkit. This approach is based not only on maintaining clarity, but also the realities and multiplicity of the threats seen in the real world.

The secondary bjectives of a rootkit may be to:

- Conceal an intruder's tracks on a rooted (compromised) system
- Concealing the presence of malicious applications or processes
- Cover the activities of binaries masquerading as legitimate files
- Conceal presence of exploits – reversed patches/reversion to older versions, backdoors/trapdoors
- Harvest information to which the intruder might not otherwise have access or full access. This might include data on the compromised system, network traffic and so on.
- Use the compromised system as an intermediary for accomplishing other intrusions and/or malicious attacks.
- Store other malicious applications and act as a server resource for bot updates and so on.

Traditional UNIX Rootkit Components

Trojanized – or Trojaned – utility programs are substituted for legitimate system utilities in order to hide the presence or footprints of an intruder, to allow an intruder access as he wishes, or gather information to which the intruder is not entitled. In a UNIX environment, utilities like "top", "ps", "login" and "passwd" are common targets for substitution [8], but any program that can be exploited to spawn a root shell (see glossary entry "shell") is a natural target for a rootkit, since this aids an intruder to gain or maintain privileged access. These are not necessarily the means used to gain access to the compromised system, but may have a number of other possible uses, including compromising other accounts and systems, or regaining privileged access in spite of attempts to repair a known intrusion.

Programs that might betray to the administrator the presence of an intruder (e.g. "last", "ls", "netstat" and "ifconfig") [9] are likewise a target for substitution. Daemons (programs like "inetd", "rshd" and "syslogd" that run as background server or system processes (see glossary) rather than being called directly by the end user) can also be targets, both for concealment and for information harvesting purposes.

Log-wiping utilities and similar tools cover the intruder's tracks by, for instance, erasing entries relating to an intrusion from system log files.

Other forms of Trojan can also be found in classic rootkits, such as those used for information harvesting (such as keyloggers and packet sniffers) and for allowing future access at will (backdoors).

A typical UNIX or Linux rootkit is likely to contain substituted utilities such as port/shell daemons, utilities for escalating user privileges, resource usage monitor utilities (to conceal files), packet sniffers to monitor network traffic, and Trojanized utilities to hide processes, logs and connections.

"The term Windows rootkit is popularly used to describe programs used to hide processes, files or registry keys from the operating system."

Windows Rootkits

The term "Windows rootkit" is popularly used to describe programs used to hide processes, files or registry keys from the operating system. In fact, Windows rootkits can have very similar functionality to traditional UNIX rootkits, though the exact mechanisms vary according to platform. NT-derived Windows versions such as XP use a multiple-account security model very similar to older systems like VMS and UNIX, though the terminology and exact mechanisms can be very different. Older Windows versions are largely dependent on third party applications for any security, so that full-blown rootkits have very little to hook into in the way of a "legitimate" security file system.

Like UNIX systems, Windows NT-derived versions of the operating environment have degrees of privileged access. This is not only a matter of user access to data areas, but of access to kernel processes. NT-derived platforms support two execution modes (or privilege levels): user mode and kernel mode. (Modern x86 processors actually support four privilege rings, but only two are supported by Windows, as NT was intended to be portable to non-Intel processors.) Privilege rings are intended to protect the kernel (ring 0) so that system data cannot be modified or overwritten by an unprivileged process. Normal applications in an NT type system run at ring 3, the least privileged process.

User Mode versus Kernel Mode

While the distinction between privileged users and administrators does not map exactly to the distinction between kernel mode (ring 0) and user mode, there is a close relationship between the two. The kernel can be described as the literal core of the operating system [10]: system services run in kernel mode, so an unprivileged user cannot introduce inappropriate modifications such as removing or adding drivers, devices and programs without authorization. User applications are generally available to all users, and run in user mode, restricting an application's ability to cause damage through inappropriate or inadvertent modification to system processes.

> "User mode rootkit components run as or within a user application, by patching the Windows API in each application process."

User mode rootkit components run as or within a user application, by patching the Windows APIs (Application Programming Interface) in each application process. Each user application runs in its own memory space, so a user mode rootkit must modify the memory space of every running application, in order to filter each application's "view" of what is happening within the operating system. To do this, it must monitor the system so that it can patch memory space before a newly-opened application is fully launched. A common means of achieving these aims is by modifying system DLLs (Dynamic Link Libraries) at runtime.

> "A kernel mode rootkit has almost unrestricted potential for damage or manipulation of the

Kernel mode allows privileged access to system memory and the full CPU instruction set, and a kernel mode rootkit intercepts kernel mode APIs. Processes, files, registry keys and so on are thus concealed. When an information request is generated by a user-mode application, the information returned is filtered to hide the evidence. A kernel mode rootkit has almost unrestricted potential for damage or manipulation

of the system, compared to that of a user mode rootkit, but, due to its inherent complexity, is harder to install and maintain reliably. Hooking into kernel space is more reliable, and easier to process in that all core processes share the same address space, but it has to be done by a privileged user (not necessarily with his or her knowledge).

Note that hooking (see glossary) is not the only method of object hiding that a rootkit can use. Direct Kernel Object Manipulation (DKOM), instead of filtering the information returned by the kernel, directly modifies "executive objects" created for auditing purposes by the operating system, to hide processes and drivers. A rootkit that uses this method of concealment can hide a process by unlinking the object associated with a process [11] so that it's harder for security or other software to detect its presence.

Persistent versus Non-Persistent Rootkits

Many rootkits are persistent: that is, they are stored on disk and hook into the system boot sequence, so that they survive a system reboot. Non-persistent or in-memory rootkits don't do this: they install their code directly into volatile memory, and do not survive a reboot. This makes them harder to detect, as there is no "footprint" on the system to be detected by a clean boot scan. However, their functionality is limited by the length of time the infected system remains online (since if the system is rebooted the rootkit disappears): this is less of an issue on systems that are not normally powered down or rebooted (servers, for instance). There seems to be a trend towards non-persistency among vulnerability researchers and those who publish PoC (Proof of Concept) code and exploits, probably attracted by the concealment advantages for malware that doesn't make permanent changes to the file system. Some Trojans have used such non persistent rootkits (for instance the FU rootkit), which they load into memory once they themselves have loaded. The rootkit is then used to hide the Trojan files and processes.

> "Non-persistent rootkits install their code directly into memory, and do not survive a reboot."

Macintosh Rootkits

Some PoC rootkits exist for Apple's OS X. Their impact has been low, and their significance tends to be downplayed by the Mac-user community as they require root access to install – that is, they can't do privilege escalation. The same attitude is commonly found towards the few examples of Mac malware currently known – indeed, Mac users often deny that there are any OS X viruses because those that exist require user action in order to infect and replicate. (However, the same is true of many Windows viruses.) It is harder for a Mac user to run as a

privileged user (unless they really want to): however, it's naïve to think that no Mac user will ever be fooled into agreeing to run root temporarily so as to allow a rootkit to install.

While Mac users make much of the Mac's "superior" privilege management, it's not particularly common for Windows rootkits to be able to "get root" – more properly, self-escalate to administrator privileges – without assistance from the (psychologically manipulated) victim. However, Windows is arguably more lenient in terms of letting the user run with administrative privileges by default.

Good Intentions, and the Sony Rootkit Experience

The use of rootkit-like or stealth technology is not confined to those who break into systems for the usual "hacking" purposes. It can, of course, used by other forms of malware, including viruses, worms and Trojans as a means of concealing the presence of such malware.

"Greyware" (software that falls somewhere between legitimate software and out-and-out malware) such as adware, some spyware, "trackware" and so on, is cautiously referred to by some anti-virus vendors as something like "potentially unwanted programs". This is to avoid possible legal complications where it can be argued (by the manufacturer) that the software is legitimate. In some cases it may even be legitimate software that can be or has been subverted for malicious purposes. These also usually depend on a degree of concealment. While their presence is sometimes very obvious, due to the appearance of pop-ups, redirection to unexpected URLs and so on, considerable attention may be paid to preventing the detection and removal of actual program files.

Some commentators in the field [4] are eager to point out that rootkit (or rootkit-like) technology can be used for legitimate purposes, and indeed must be to overcome operating system deficiencies when it comes to data hiding. Some of the areas sometimes cited are:

- Intellectual Property Rights management
- Protecting programs from reverse engineering
- Assessing the threat from insiders
- Intrusion tracking
- Employee monitoring
- Digital Rights Management
- Protection of security software from malware or inappropriate user interference
- Backup software and data
- System recovery software
- Data encryption and concealment on multi-user systems

Not everyone is comfortable with the application of rootkit-related terminology to such issues, because it muddies the definitions somewhat. After all, Windows itself effectively employs such techniques by default, to hide important system files. However, some vendors have clearly been comfortable with both the terms and the concepts. In October 2005, Mark Russinovich reported in his Sysinternals blog [12] his discovery of what seemed to be a rootkit on his system. This turned out to be the notorious Sony "rootkit", and resulted in serious confusion in popular thinking between legitimate Digital Rights Management (DRM) and stealth/rootkit technologies. DRM can be defined as the technology used to control access to data and hardware so that the rights of the publisher/copyright owner are maintained.

Sony used XCP (Extended Copy Protection) technology from First 4 Internet Ltd to control access to some music CDs. XCP protected disks restricted the number of copies of CDs or DVDs to be made, and also controlled the "ripping" of music into a digital format suitable for storing and replaying on a computer or portable music player such as an MPEG player. It was not possible to play the CD on a PC without installing the software, which then hid files, processes, and registry keys/values by modifying the execution path of API functions. It did this using the common rootkit technique of patching the System Service Table (SST).

Sony's right to address the illicit distribution of their product is not generally in doubt, and it doesn't seem appropriate to regard their approach as a rootkit in the malicious sense. However, many are made uncomfortable by the fact that they modified the end-user's system without making it clear what they were doing or providing a ready means of uninstalling. This may have put them in danger of breaching legislation that outlawed unauthorized access or modification. Even worse, their solution was not altogether well-conceived or coded, and created a vulnerability that was seized on with gratitude by the black hat community. Indeed the real problem was that the "rootkit" was exploited to hide a Trojan, which itself had nothing to do with Sony, nor the copy protection software. This demonstrates a subtle difference between malicious intent and inadvertent vulnerability: clearly, Sony and First 4 Internet did not intend or expect that their DRM scheme would open up a gap that could be exploited by malware. Nor did Sony handle the subsequent publicity well, and their first attempt to correct the problem was inadequate. The de-installation was only made available to those who completed a form, and offered a stark choice of uninstalling software and not being able to use XCP-protected CDs, or just disabling the concealment function. Reportedly, the initial patch was both bloated and poorly tested, and had "phone home" functionality that wasn't explicitly mentioned [13] in the EULA (End User License Agreement).

The XCP technology hasn't much impressed the rootkit community - by which I mean those who take a deep technical interest in the (not necessarily illegitimate) use of stealth and rootkit-like technology - though it did focus the minds of some malware writers on ways of using the vulnerability it originally created. However, it does have a number of implications

for the industry generally.

It's not unlikely that there will be attempts to solve the rootkit problem by legal means [10]. This may take the form of specific legislation against malicious rootkit types, but may also result in attempts at the outlawing of rootkit-like technology even for legitimate purposes. If this were to happen, any application that uses any form of stealth to protect its own or other code or data could be in legal jeopardy. This would have serious implications for Digital Rights Management (DRM), which is generally reliant on a measure of concealment and restricted access.

"It's not unlikely that there will be attempts to solve the rootkit problem by legal means."

Even if such measures aren't taken, there are issues regarding how products are regarded by potential users. Take the example of the Symantec Protected Recycle Bin (a secure recycle bin that allows full recovery of deleted files that might not be recoverable through the Windows recycle bin) – a legitimate, documented and potentially useful feature. Nevertheless, after headlines about the "Symantec rootkit", it was unhidden [14] to lessen the risk of exploitation by malicious software. This has implications for other security software, some of which *has* to hook so as to play first (for instance, behavior monitoring and blocking software uses similar API hooking techniques to rootkits, though for purposes of detection rather than concealment. Other hidden functionality is used by security and other software – for example, to discourage reverse engineering, discourage user's self-exposure to malicious code (quarantined viruses etc.), "meddling" with system settings, and so on.

Rootkit Detection Methodology

Every so often, someone "discovers" that so-called signature-based malware detection is flawed, in that it "can't detect new viruses". Fortunately, anti-virus hasn't relied solely on known virus detection for many years. A whole raft of supplementary technologies (heuristic analysis, generic drivers, behaviour monitoring and so on) are already used to enhance the detection of new threats and variants. As with stealth viruses, the rootkit problem is one of "Who plays first?" It's difficult for a scanner to detect what's already installed and concealing the evidence of an infection. However, AV vendors have many years of experience at finding ways of circumventing novel spoofing mechanisms, once the malware has been analysed. In general, rootkits can usually be detected by file-system and memory scanning using so-called signature scanning.

"The growth in non-persistent rootkits reinforces the need for memory scanning and the detection of hidden processes"

A characteristic application of supplementary

technologies to detect possible rootkit activity heuristically is largely dependent on checking for disparities between a trusted (reasonably accurate) view of the system and the tainted view presented by the system when filtered through rootkit technology [15].

Classically, UNIX and UNIX-like operating systems have been well served by what might be called the *tripwire* or object reconciliation approach [9], though in the mainstream anti-virus industry, it's most commonly referred to as detection. It can still be useful in a Windows context, but tends to have a high maintenance overhead, as there are many instances where changes to the environment (executables, registry settings, configuration files) are routine. The growth in non-persistent rootkits reinforces the need for memory scanning and the detection of hidden processes, rather than relying on changes to the file system as an indicator of infection. A more proactive approach to the detection of variants is also desirable. The more "intelligent" a product is when it comes to discriminating between likely malicious hooks, registry settings (and so on) and their legitimate counterparts, the more effective such a heuristic approach is likely to be. [10]

Antivirus has, of recent years, become rather adept at the heuristic removal of viruses [16]. Nevertheless, difficulties of heuristic removal, where exact identification of the virus is not possible, remain, and the more so when it comes to non-viral malware. While claims that rootkit infection can only be redressed by complete reinstallation of systems are exaggerated, it can be more efficient to re-image a system than to "simply" remove the rootkit, even with known but deeply-embedded malware, particularly in cases where other executables have been patched or altered. To be able to do so without undue damage to data and productivity, however, requires careful adherence to good backup (of data as well as of system and user applications) and reinstallation practice.

Preventative Measures

There are countermeasures that make sense on any platform. Good backup practice is a vital defense against threats and unplanned disasters, and should be a cornerstone of any data integrity policy

Administrators should not run as root/administrator unless running a session that primarily needs privileged access to get the job done: use an unprivileged account for routine work, where possible. All the common modern platforms allow some form of switching accounts without rebooting where privileged access is needed.

It can be dangerous to rely on open source/volunteer security software. Many community projects have produced excellent work, but it's sometimes difficult to assess the competence (and, sometimes, the good intentions) of everyone involved in such a project.

"Good backup practice is a vital defense against threats and

Certainly, it's inappropriate for a commercial or public sector organization to entrust its security to a product for which no guarantees or publisher accountability are in evidence. Home and SOHO users may be less troubled by governance issues, but will not want to be left in the lurch by software that proves in some sense unfit for purpose. Also important is the need to conduct vulnerability testing, maintain regular patch management processes and to avoid social engineering tricks.

Conclusion: A Heuristic Paradox

Don't panic. The sky is not falling. Or is it?

Joanna Rutkowska's "Blue Pill" experimental technology is alleged to allow for the creation of "100% undetectable malware, which is not based on obscurity of the concept" [17], by exploiting AMD's SVP/Pacifica virtualization technology. Insufficient detail has been released to date concerning this approach to evaluate the claims made for it: it seems to be based on a non-persistent rootkit running inside a virtual machine.

The SubVirt rootkit [18] also uses virtualization, but is persistent (i.e. it survives a reboot). It's an interesting proof of concept, but is by no means undetectable. It will be interesting to see in due course if Blue Pill is really "superior" in this respect.

Nonetheless, it's safe to assume that the malware arms race, where the advantage regularly passes from the bad guys to the good guys and back, will continue for a good while yet. Neither panic nor complacency are appropriate, but vigilance is. Many of today's malware authors are now employing rootkit technology to hide their creations, but the prevalence of such objects is still fairly low, and the essential nature of malware tends to lend it more easily to detection because of its actions.

Reports of the death of anti-virus are much exaggerated; indeed, the term "Anti-Virus" is somewhat misleading when talking about today's security solutions. The days of anti-virus programs only detecting viruses and ignoring anything else are long gone. Many products are now capable of detecting a very wide range of modern malware threats. Just as the threats have evolved, the anti-virus industry has not stood still, and some AV vendors have had significant success in detecting rootkits. Nevertheless, it's not safe to close your eyes and expect everything to be taken care of for you. Purely "signature-based" anti-virus cannot protect against new threats that are substantially different from known threats, even assuming that they are updated regularly. End users must use products that have a proven track record using advanced heuristics and other generic, proactive detection methods and be vigilant in their evaluations of anti-threat technologies and product capabilities.

References

1. University of Minnesota ResNet FAQ: http://www.resnet.umn.edu/html/rn_security.html
2. "Viruses Revealed". David Harley, Robert Slade, and Urs Gattiker (Osborne).
3. "Dr. Solomon's Virus Encyclopaedia". Dr. Alan Solomon and Dmitry Gryaznov. (S&S International).
4. "Rootkits are not Malware". Greg Hoglund. http://www.rootkit.com/newsread.php?newsid=504; http://www.sysinternals.com/Forum/forum_posts.asp?TID=5798
5. "Using a 'common language' for computer security incident information". By John D. Howard & Pascal Meunier, in Computer Security Handbook (4th Edition) ed. Seymour Bosworth & M.E. Kabay (Wiley).
6. "A Short Course on Computer Viruses" 2nd Edition. Dr. Frederick B. Cohen, Wiley; "Models of Practical Defenses Against Computer Viruses". Dr. Frederick B. Cohen: http://all.net/books/integ/vmodels.html
7. http://blogs.securiteam.com/index.php/archives/382
8. Chey Cobb, Stephen Cobb, M.E. Kabay: "Penetrating Computer Systems and Networks". In "Computer Security Handbook 4th Edition", ed. Bosworth & Kabay (Wiley).
9. "Trojans" David Harley. In "Maximum Security" (SAMS).
10. "Rootkit Threats Explained". Andrew Lee. Eset, 2006. http://www.eset.com/joomla/index.php?option=com_content&task=view&id=1401&Itemid=5
11. "Windows Rootkits of 2005" Parts 1-3. James Butler and Sherri Sparks. http://www.securityfocus.com/infocus/

"12. Sony, Rootkits and Digital Rights Management Gone Too Far". Mark Russinovich. http://www.sysinternals.com/blog/2005/10/sony-rootkits-and-digital-rights.html; "More on Sony: Dangerous Decloaking Patch, EULAs and Phoning Home". Mark Russinovich. http://www.sysinternals.com/blog/2005/11/more-on-sony-dangerous-decloaking.html

13. http://cp.sonybmg.com/xcp/english/updates.html; http://cp.sonybmg.com/xcp/english/form14.html
14. http://securityresponse.symantec.com/avcenter/security/Content/2006.01.10.html
15. "Hide 'n Seek Revisited – Full Stealth is Back". Kimmo Kasslin, Mika Stahlberg, Samuli Larvala and Antti Tikkanen. In Proceedings of the 15th Virus Bulletin International Conference, 2005.
16. "The Art of Computer Virus Research and Defense". Peter Szor (Addison-Wesley)
17. "Subverting Vista Kernel for Fun and Profit" Joanna Rutkowska. http://theinvisiblethings.blogspot.com/
18. "SubVirt: Implementing malware with virtual machines". Samuel T. King, Peter M. Chen, Yi-Min Wang, Chad Verbowski, Helen J. Wang, Jacob R. Lorch. http://www.eecs.umich.edu/virtual/papers/king06.pdf

Glossary

API Hooking	In the rootkit context, hooking describes the following process. When a request for system information is generated, it calls an API function which feeds the request to the kernel and returns the requested information back to the requesting application via the same"execution path". Hooking describes the process of hiding data by diverting the execution path through a filter, which modifies the data returned.
AV	Anti-Virus
Backdoor-Ali (AKA ierk or slanret)	A backdoor Trojan often described as a rootkit
Daemon or demon	In UNIX, a server or system process that runs in the background, as opposed to being executed by the user. For example, the program ftpd is a daemon, a system service that runs all the time: the program ftp is a client program, called by an end user, which used the ftpd system service.
Deepdoor	Proof of Concept rootkit by Joanne Rutkowska
Generic Driver or Generic Detection	In anti-virus, a convenient but not universally used term to describe a virus signature or definition which has been generalized so as to detect a family of viruses or variants, rather than detecting a single unique variant. This is similar to but not the same as the distinction between exact and almost exact identification, since almost exact identification is usually associated with generic disinfection. In-depth discussion of these concepts is beyond the scope of this paper. A good source of further information is Peter Szor's "The Art of Computer Virus Research and Defense".
Governance (Information Governance)	Information management framework, particularly with regard to policy and compliance issues.
Grayware, Greyware	A somewhat fuzzy class of software that may include adware, spyware, joke programs, remote access programs etc. Tends to include a presumption of concealed functionality.
Hacker Defender	A very widely found rootkit.
Hacktool	Windows rootkit
Heuristics	A blanket term applied to a range of techniques for detecting currently unknown malware and variants.
Keylogger	Software that monitors keystrokes, often (but not necessarily) for sinister purposes (password stealing, for example)

Lrk	A well-known, well-established Linux rootkit. Described at http://staff.washington.edu/dittrich/misc/faqs/lrk4.faq
Opener (AKA Renepo)	Macintosh malware sometimes described as a rootkit.
OS X	The current Macintosh operating system, based on BSD UNIX but with an interface that continues the long-running Mac user-friendly "look" as well as access to a more conventional UNIX command line.
OSXrk	Macintosh OS X rootkit.
Shadow Walker	PoC rootkit by James Butler and Sherri Sparks, partly based on Butler's FU rootkit.
Shell	The command processor which interprets commands typed in at a terminal into the system's instruction set. However, the term is also used to describe a process or chain of processes launched by the command processor. This may also be referred to as spawning a shell or shelling out. A process that makes use of root privileges may be referred to as a root shell. It isn't always necessary for a user to be able to run as root themselves: some programs can run as root without extending root privileges to the user. However, this mechanism can, where vulnerabilities such as overflows exist, be used by a malicious intruder or program to "get root." (While the terminology used here is fairly UNIX-specific, analogous processes exist in other operating systems.)
Signature scanning	Strictly, searching for the presence of a virus by checking for a more-or-less static byte sequence. In fact, even basic AV scanning uses more complex and effective techniques nowadays, using algorithmic approaches, wildcards and so forth. The term signature is often deprecated in anti-virus research because of this ambiguity, though it's probably far too late to eradicate it from popular and media use and misuse. However, it is still routinely used in intrusion detection.
SOHO	"Small Office, Home Office"
Stealth	Adjective: often used in security as an alternative to "stealthy", by analogy to "stealth aircraft" and other military terminology relating to concealment strategies.
Stealthware	Software (usually malware) that uses stealth techniques to conceal itself

Trackware	Software that tracks system usage and information. Sometimes (but by no means invariably) used to distinguish software that tracks with the knowledge of the system user.
WeaponX	Kernel based rootkit for OS X

Corporate Headquarters

ESET, spol. s r.o.
Aupark Tower
16th Floor
Einsteinova 24
851 01 Bratislava
Slovak Republic
Tel. +421 (2) 59305311
www.eset.sk

Americas & Global Distribution

ESET, LLC.
610 West Ash Street
Suite 1900
San Diego, CA 92101
U.S.A.
Toll Free: +1 (866) 343-3738
Tel. +1 (619) 876-5400
Fax. +1 (619) 876-5845
www.eset.com

TROAE200812010