Imperial Journal of Interdisciplinary Research (IJIR)
Vol-2, Issue-5, 2016
ISSN: 2454-1362, http://www.onlinejournal.in

Detection Techniques of Rootkits

Shrikant Bhardwaj & Kumar Pranjal
Computer Science and Engineering, Poornima Group of Institutions, Jaipur, Rajasthan

Abstract: *A rootkit is a collection of tools that is designed to gain administrator-level access over a computer system while hiding itself from the user and the operating system, by compromising the communication channels within the operating system. A well-designed rootkit can hide files, data, processes, and network ports, and can typically survive a system restart. The effect of this stealthy design allows the rootkit to perform malicious activities such as keystroke logging or give a remote attacker control of the infected system. Even though current rootkits are extremely stealthy, there still exist a number of techniques that have been developed to detect their presence. These techniques include signature-based detection, heuristic or behavior based detection, host integrity monitoring, and network-based detection. This thesis will compare the operation of different types of detection methods against several of the most common rootkits that are currently affecting Windows-based systems.*

Keywords: *Rootkits, Operating System, Communication*

1 Introduction

The Windows Operating System (OS), like many modern operating systems, is designed as a layered architecture. Figure 1 shows how the users and applications are protected from the hardware minutiae by a number of software layers in the Windows OS. The layering provides a high level of extensibility and portability, but at the same instance creates a number of opportunities for attackers to cooperate the system. If one of the communication paths between the layers is controlled by a malicious user, the hacker can perform activities such as keystroke logging, or become a member of a botnet that sends spam emails or performs Denial of Service attacks, and not be detect by the user or the OS. Rootkits focus on these communication paths and interfaces to hide their presence on the OS.

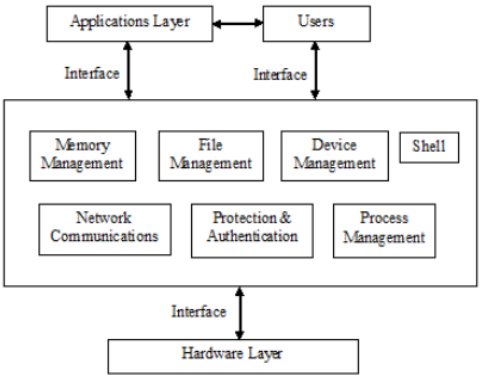

Figure 1. Windows OS Architecture [2]

1.1 Research Objectives

The goal of this research is to compare different types of detection techniques and their connected tools against numerous of the most common Windows based rootkits that are currently infecting computers. As part of the research, a detailed understanding of modern rootkit designs and detection techniques, as well as Windows networking internals will be gained. Specific outcomes from this planned research will include detailed analysis and comparisons of representative rootkit detection techniques, including their relevant strengths, weaknesses, performance/overhead, and ease of deployment. Both theoretical analysis and experimental evaluations will be 4 performed. Additionally, forensic analysis of several different types of modern rootkits will be performed.

1.2 Related Work

Anti-Virus Comparatives is a self-governing organization that performs regular comparison testing of Anti-Virus software. Their testing tactic is very thorough, as they use the latest copies of almost all available AntiVirus products against a representative sample of currently-active malware. However, rootkits and rootkit detectors are not the focus of this analysis, so there is an opportunity for this research to provide precious information to the community. Yegulalp [1] provided a good functional comparison and description of several of the recently-developed rootkit detection tools, but did not perform any methodical testing of these software packages against a variety of current rootkits. NT Internals [4] performed a fairly

thorough testing of almost all available rootkit detectors, but did not include any of the modern rootkits such as Rustock, Zeus, or TDL3/Alureon in the test set. This is significant, because many of the current rootkits have significantly evolved to use different techniques than previous versions, and are actively subverting many of the detectors that are available. The current detection technology should easily be able to find rootkits from this obsolete test set, so the relevancy of the results is disputed.

1.3 Potential Benefits

The development of rootkits and rootkit detectors is a regularly changing landscape, and it is important to have the most recent information available when making a decision on how best to protect or clean a computing system. The research in this research will help bring to light the fact that many formerly effective solutions have not kept up with the speed of modern rootkit development, and should no longer be used. Additionally, the characteristics of the rootkit detectors will be analyzed to determine if there is a particular technique or combination of techniques that is able to detect rootkits more successfully. Additionally, a set of computer system and malware forensic analysis skills will be developed during the course of the research. This research will document how debugging tools and other analysis tools can be used in the analysis of recently-developed rootkits.

2 Surveys of Rootkit Techniques

This section will provide an general idea of several rootkit design techniques that have evolved over the years. The design and detection of rootkits can best be described as an "arms race", with the rootkit authors and the security community appealing in a constant process of one-upmanship. The initial rootkits focused on UNIX-based systems, and used fairly primal designs that replaced system files with malicious versions, and were easily detected by file system scanners. Over time, the rootkit techniques have evolved into using undocumented operating system data structures and even extremely hardware-dependent systems that operate independently of the OS and are extremely difficult to detect.

2.1 File Masquerading

One of the earliest rootkit techniques was to replace system files with malicious versions that shared the services and same name as the original. This technique is known as file masquerading [2]. For example, a system file that provides a function or service to list files and folders (eg., Windows "dir" command) could be replaced with a version that filters out all of the malicious files, effectively hiding the malware from the system. However, this technique is easily detected by file system integrity tools such as Tripwire [4], which compares baseline "clean" versions of the system files against the current file system using a Cyclic Redundancy Check (CRC). If any discrepancies are found, the file system has likely been compromised and cannot be trusted.

2.2 Hooking

The next step in the development of rootkits was to forward system calls to malicious code, a technique known as "hooking" [2]. Hooking is when a given pointer to a given resource or service is redirected to a different object. For example, instead of completely replacing the file containing the "dir" command as described in the previous section, the system call can be forwarded to a custom "dir" command in memory space that filters out the malicious folders and files. Basically, hooking attains the same effect as file masquerading, but is more hard to detect, since the system files on disk are not changed. This type of technique cannot be detected by file integrity checkers as explained in the previous section, so in order to counter this technique, memory scanners such as Rootkit Unhooker [3] were developed. Figure 3 shows a typical path of a Windows-based function call starting at the ending and user application in the physical hardware. There are numerous different locations along the way that can be hooked to perform both legitimate activities and malicious. These locations include userland hooks in the Import Address Tables (IAT), the Interrupt Descriptor Table (IDT), the System Service Dispatch Table (SSDT), and device drivers via I/O Request Packets [3]. These tables keep memory addresses that point to various functions and interrupt request handlers, which can be altered to point to malicious programs that are resident in memory.

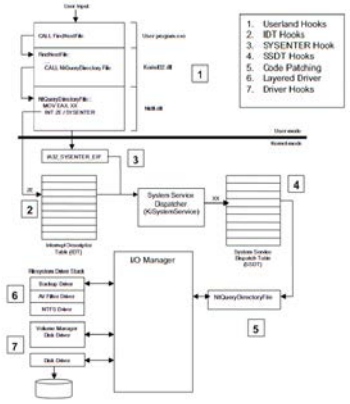

Figure 2. Potential Hooking Locations in Windows

2.3 DKOM

The third generation of rootkits used technique known as Direct Kernel Object Manipulation (DKOM). DKOM can operate kernel data structures 10 to hide processes, change privileges, etc. The first known rootkit to perform DKOM was the FU rootkit, which modified the EPROCESS doubly linked list in Windows to "hide" the rootkit processes. This technique took advantage of the fact that there are two separate lists for threads and processes in Windows. As shown in Figure 4, by modifying the BLINK and FLINK pointers in the EPROCESS list (and leaving the thread list alone), the rootkit was able to remove the offending process. The associated malicious threads are then allowed to continue being executed by the CPU scheduler [3]. DKOM requires a lot of reverse engineering and a detailed knowledge of OS internals, and can be very challenging to detect due to many undocumented features and the copyrighted nature of the Windows source code.

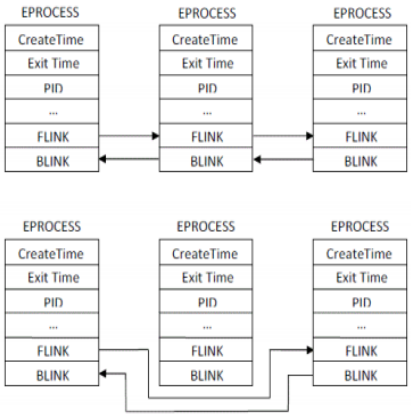

Figure 3. DKOM EPROCESS list modification

2.5 Filter Drivers

The Windows driver stack architecture was designed in a layered manner, so that third party hardware manufacturers can insert their drivers within already existing layers and utilize existing functionality provided by the Windows OS [2]. This feature also creates yet another opportunity for rootkit authors to inject their malicious code to interrupt the flow of I/O Request Packets and perform activities such as filtering or keystroke logging the results that are returned to anti-malware applications. Rootkit authors can perform patch driver routines , hooking of drivers, or even create an entirely new driver and insert it into a driver stack. Figure 4 shows a representative example of a Windows Driver stack.

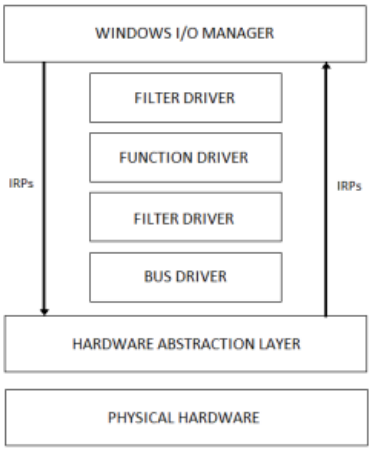

Figure 4. Windows Device Driver Stack

2.6 Hardware-Based

The fourth and final type of rootkit operates independently of the OS, but is extremely hardware dependent. These rootkits typically use hardware virtualization and chipset exploits to operate in the PCI or BIOS expansion cards [2]. At this time the hardware-specific rootkits are not very prevalent in the wild, and are more "proof of concept" techniques. Techniques to detect these types of rootkits are likewise very sparse.

References

[1]Yegulalp S. Review: Six Rootkit Detectors Protect Your System. In: Information Week.
[2] Sparks S, Embleton S, and Zou C, Windows Rootkits - A Game of
Hide and Seek. In: Handbook of Security and Networks, World Scientific Press, 2010.
[3]Ries C, "Inside Windows Rootkits", Vigilant Minds, Inc., May 22,2006
[4]Ionescu A, "NT Internals," Accessed on http://www.ntinternals.org/index.php.