Chapter 12

TIMELY ROOTKIT DETECTION DURING LIVE RESPONSE

Daniel Molina, Matthew Zimmerman, Gregory Roberts, Marnita Eaddie and Gilbert Peterson

Abstract This paper describes a non-intrusive rootkit detection tool designed to support forensic investigations that involve the live analysis of computer systems. The tool, which does not require pre-installation, correlates outputs from multiple system data gathering utilities. Test results indicate that the tool successfully detects several well-known rootkits, including Hacker Defender, AFX, Vanquish, FU and FUto.

Keywords: Rootkit detection, live response

1. Introduction

Rootkits enable attackers to have undetected access to computer systems; they hide an attacker's presence by manipulating system data and/or operating system code. These malicious programs hamper digital forensic investigations – evidence that is collected carefully may still be compromised by active rootkits. In particular, rootkits may prevent forensic tools from gathering accurate information. Addressing this issue requires investigators to run more intrusive tools that can alter the system state.

This paper describes a rootkit detection tool designed for live analysis that minimizes system modification. The tool, which does not require pre-installation, correlates outputs from multiple system data gathering utilities. The detection tool has proved to be effective against five rootkits: Hacker Defender [8], AFX [14], Vanquish [17], FU [15] and FUto [16]. In the tests, these rootkits were hiding a backdoor (Back Orifice 2000 [19]) and a folder containing four files.

2. Rootkits

Rootkits are programs that enable attackers to have undetected access to computer systems. The term "root" comes from the UNIX world where "root" is the highest level of privilege afforded to a user. Originally written for UNIX, rootkits now target a variety of operating systems.

A rootkit typically incorporates Trojaned system processes and scripts that automate the actions involved in compromising systems [12]. Rootkits often attempt to be untraceable by hiding files, network connections, memory addresses and registry entries. Some rootkits are embedded in other programs or media as in the case of the rootkit found in Sony CDs in 2005 [4].

Windows rootkits – like those that target UNIX systems – seek the highest possible privilege level. Windows runs on the Intel x86 architecture, which employs a memory protection scheme using four rings (Rings 0–3). Ring 0, which has the highest level of privilege, represents the memory space where the operating system kernel and drivers reside. Ring 3 has the lowest privilege level and represents the memory space where user applications reside.

Stealthy rootkits tend to operate at a lower ring than Ring 3 where rootkit detection and prevention software typically operates. Hoglund [7] and Rutkowska [18] note that placing a rootkit detector in a lower ring increases the detection rate. On the other hand, a rootkit that executes in a lower ring than a detector can control or fabricate the information gathered by the detector, which enables the rootkit to remain hidden.

The rootkit detection technique described in this paper correlates outputs from multiple system data gathering utilities. These utilities are executed from user space (Ring 3) without prior installation.

2.1 Rootkit Categories

Rootkits are categorized as kernel, library, user-level, hardware-level and virtual machine based rootkits. Some rootkits fall in multiple categories, e.g., those with kernel and user-level components.

- Kernel Rootkits: These rootkits add additional kernel code and/or replace a portion of kernel code to enable them to obtain stealthy control of computer systems.
- Library Rootkits: These rootkits achieve stealth by modifying system libraries used by user and/or kernel applications [2].
- User-Level Rootkits: These rootkits, also called application-level rootkits, are programs that modify system files or binaries on disk [11].

- Hardware-Level Rootkits: These rootkits attempt to subvert computer systems at the lowest level. They are extremely difficult to implement, but Heasman [6] has demonstrated that such rootkits are possible. We do not attempt to detect hardware-level rootkits because of their complexity and lack of availability.

- Virtual Machine Based Rootkits (VMBRs): These rootkits attempt to take control of the virtual machine monitor (VMM), which lies between the hardware and operating system. Thus, VMBRs are able to control requests to the hardware that originate from the upper levels. A VMBR typically modifies the boot sequence and loads itself instead of the chosen VMM or operating system. After it is loaded, the rootkit loads the host operating system as a virtual machine. An example of a VMBR is SubVirt [9].

  A VMBR is difficult to detect during live analysis because rootkit detection software is executed within the virtual machine. Software running on the target machine cannot access the state of a VMBR [9]. From the user's perspective, a VMBR is in a hidden VMM where malware can operate without interference. A VMBR thus has the ability to access all keystrokes, network packets, memory allocations, system events, etc. We do not attempt to detect VMBRs because of their complexity and lack of availability.

2.2 Rootkit Hiding Techniques

Kernel and user-level rootkits apply various hiding techniques, either individually or in combination. The principal hiding techniques are:

- Patching: This technique involves static or dynamic modification of binaries. Static patching is also used by software crackers to bypass software protection and registration methods.

- Hooking: This technique redirects or alters the normal flow of execution of a program by modifying one or more function calls in memory. Hacker Defender is an example of a rootkit that uses hooking.

- Direct Kernel Object Manipulation: This technique exploits the way the Windows OS schedules processes. Malicious processes are hidden by removing their entries from the doubly linked list used by the Windows Object Manager [7]. FU and FUto are examples of rootkits that use this hiding technique.

2.3 Rootkit Detection Techniques

Rootkit detectors fall in one or more of the following categories [21]:

- Signature Based Detectors: These detectors scan system files for rootkit fingerprints.
- Heuristic/Behavioral Based Detectors: These detectors check for deviations from normal system behavior.
- Cross-View Based Detectors: These detectors compare system parameters obtained in two or more different ways to detect inconsistencies or anomalies.
- Integrity Based Detectors: These detectors compare the current snapshot of a system to a known trusted snapshot.
- Hardware-Based Detectors: These detectors employ direct memory access to retrieve data, which is scrutinized for rootkit fingerprints.

At the time of writing this paper, software-based rootkit detectors have components that execute from user space, kernel space or both. A detector is much more effective when it runs at a level below the rootkit. For example, if a rootkit only executes in user space then the detector has a better chance of detecting the rootkit from kernel space.

Kernel-level rootkits can be identified by a detector that coexists with the rootkit in kernel space or by a hardware-based detector. However, such detectors cannot be used during a live response because they affect evidence integrity.

Some anti-virus programs include rootkit detection features. For example, F-Secure Internet Security 2005 offers "manipulation control," a behavior-blocking mechanism that prevents malicious processes from manipulating other processes [5].

Some of the well-known rootkit detectors are:

- BlackLight: This Windows rootkit detector identifies rootkit files, folders and processes, but not hidden registry keys [20]. It offers a removal option for detected rootkits; however, this feature must be used with care to avoid problems with the computer system.
- RootkitRevealer: This Windows tool detects rootkits by performing a high level scan from user space and a raw disk scan; the results of the two scans are compared for anomalies. The tool reports differences in the Windows registry and file system. However, it does not have any rootkit removal capabilities [20].

- IceSword: This Windows tool suite includes a process viewer, startup analyzer, port enumerator and other utilities. The tool only provides data, leaving rootkit identification to the user [20].

- Chkrootkit: This UNIX shell script checks specific system binaries to determine if a rootkit has been installed [10].

- Rootkit Hunter: This UNIX rootkit detector performs MD5 comparisons of critical system files and searches for known rootkit files, hidden files and suspicious information in loadable kernel modules. Also, it checks file permissions and scans plain text and binary files for strings that indicate the presence of rootkits [1].

3. Live Response Analysis

Digital forensic investigators typically employ live analysis tools (e.g., FRED or Helix) to collect evidence from running systems. However, even if the evidence is collected carefully and documented diligently, it may be compromised by active rootkits.

Forensic investigators must be cognizant of how user-level and kernel-level rootkits affect the integrity of computer systems. User-level rootkits alter the security subsystem and display inaccurate information; they intercept system calls and filter output APIs to hide processes, files, system drivers, network ports, registry keys and paths, and system services [3]. Kernel-level rootkits usurp system calls, hide processes, registry keys and files, and redirect calls to Trojan functions [13].

Due to the increasing threat of rootkits and their impact on computer systems, digital forensic investigators must strive to detect rootkits in a timely manner. Rootkit detection tools can aid investigators in determining if rootkits are present and which data may have been compromised. Some of these tools have to be installed before a system is exposed to a hostile environment. Installing a rootkit detector during live analysis can dramatically alter the state of the system.

Once a computer system is turned off, a rootkit cannot actively hide itself or other information that could indicate its presence. A forensic investigator may, therefore, conduct an off-line analysis of the computer image and search for signatures that reveal the presence of rootkits. However, even if a rootkit is detected, the investigator may have difficulty determining whether or not the rootkit was active.

4. Rootkit Detection System

The Windows rootkit detection system described in this section uses open source utilities that perform a system scan from user space. These

utilities, which do not have to be pre-installed, are executed via a batch script that runs each utility and sends its output to a separate file. After the batch script completes, data from the output files is correlated by an automated analysis program to identify discrepancies. A Java-based GUI with a file parser is used to generate a report of the discrepancies. Because the utilities are executed from trusted media, it is important that the GUI and parser utility also run from trusted media. This is accomplished by using a Java Runtime Environment (JRE) located on the trusted media source.

The rootkit detector provides investigators with the ability to initiate a batch job that invokes all the command line utilities and scans the output files for potential threats. The outputs of the utilities are analyzed for (i) differences between output files that should display identical information, and (ii) combinations of discrepancies that could indicate possible threats. While it is simple to classify all of the differences between the outputs, it is difficult to categorize and assess all possible combinations of discrepancies. Therefore, the detector focuses on certain key differences and combinations of these differences.

Figure 1 illustrates the identification of discrepancies. The scenario involves identifying the presence of a hidden file called `secret.hide` by examining directory listings produced by two different tools (`dir` and `ls`). This method is effective at discovering files hidden by the Vanquish rootkit.

Having identified the discrepancies in the output files, predefined detection rules are applied to determine if the discrepancies indicate the presence of a rootkit. Figure 2 presents a scenario where three discrepancies have been identified: Discrepancy A comes from comparing the outputs of `dir` and `ls.exe`, Discrepancy B comes from comparing the outputs of `pslist.exe` and `handle.exe`, and Discrepancy C comes from comparing the outputs of `dir` and `handle.exe`. Information about the discrepancies is passed to the detection rules (as in Figure 2), which determine if a rootkit is present or not.

5. Results

The Windows-based rootkit detection tool was tested on a system running Windows XP with Service Pack 2. To make the testing process consistent, the victim system was run on VMWare, which enables malicious code (e.g., rootkits) to be executed without infecting the host operating system. In addition, it enables the user to start or stop a test image quickly and reliably, and to go back to previous snapshots.

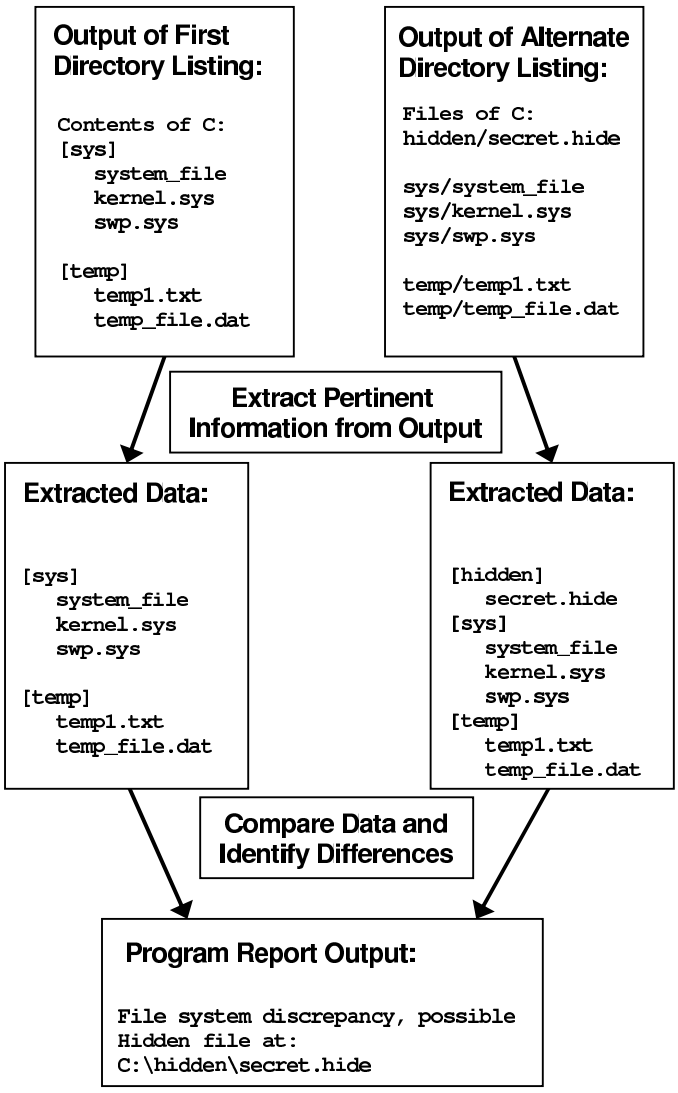

Figure 1. Identification of discrepancies.

Initial tests involved running the batch script and analyzing each output file for evidence of a rootkit. The output file of `handle.exe` clearly indicated the presence of the Hacker Defender rootkit via a non-existent process with a PID and an object named Hacker Defender. The AFX and Vanquish rootkits were also detected during initial testing.

The next step was to perform tests against rootkits that employed more sophisticated hiding techniques (i.e., kernel-level subversion). The

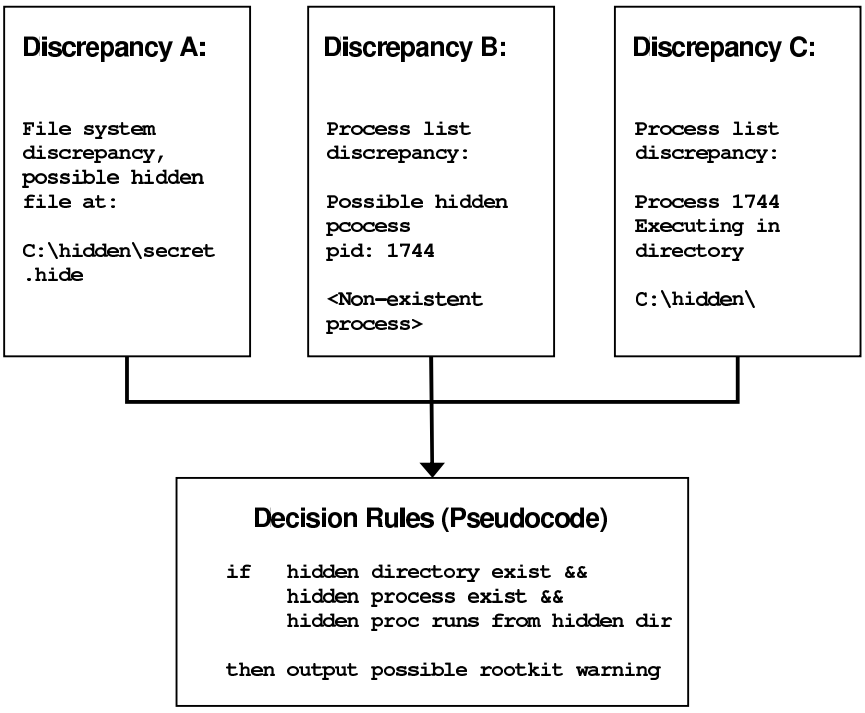

Figure 2. Discrepancy combinations used for detecting rootkits.

rootkits tested were FU and FUto, both hiding the Back Orifice PID. The tool identified the Back Orifice PID, but was unable to identify the FU and FUto PIDs. However, upon performing a directory listing of the folder `C:\Windows\Prefetch`, it was determined that `bo2K.exe` and `FU.exe` had prefetch files that indicated that the two programs had been executed.

Table 1. Experimental results.

Rootkit	Rootkit PID	Bo2k PID	Bo2k Port	Hidden Data
Hacker Defender	Found	Found	Not Found	Found
AFX	Found	Found	Not Found	Found
Vanquish	N/A	N/A	N/A	Found
FU	Not Found	Found	Not Found	N/A
FUto	Not Found	Found	Not Found	N/A

Table 1 summarizes the experimental results. The principal result is that Back Orifice was detected for all the user-level and kernel-level

rootkits tested. However, the rootkit detection tool was unable to locate Back Orifice's open ports.

6. Conclusions

Evidence collected during live analysis of systems can be compromised by active rootkits. Digital forensic investigators need automated tools that can detect rootkits during live response investigations of computer systems. The rootkit detection tool described in this paper has proved to be relatively effective in tests. Specifically, the tool was able to identify the PIDs of the Hacker Defender, AFX, and Vanquish rootkits, the PIDs of their backdoors and the folders they were attempting to hide. Tests against the FU and FUto rootkits were not as successful; the only evidence obtained was the name of the executable `FU.exe` in the prefetch folder. On the other hand, the PID and file name of Back Orifice were easily detected although FU and FUto were attempting to hide this information.

Topics for future research include performing experiments with other rootkits and backdoors, conducting an exhaustive examination of the Windows API to identify all the alternative ways for obtaining system information, and investigating rootkit detection in UNIX environments.

Acknowledgements

This research was supported by the Anti-Tamper Software Protection Initiative Technology Office, Sensors Directorate, U.S. Air Force Research Laboratory. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the U.S. Air Force, U.S. Department of Defense or the U.S. Government.

References

[1] M. Boelen, Rootkit Hunter (www.rootkit.nl/projects/rootkit_hunter.html).

[2] A. Chuvakin, An Overview of Unix Rootkits, iALERT White Paper, iDefense Labs, Chantilly, Virginia, 2003.

[3] K. Dillard, What are user-mode vs. kernel-mode rootkits? (search windowssecurity.techtarget.com/originalContent/0,289142,sid45_gci1086469,00.html), 2005.

[4] J. Evers, Microsoft will wipe Sony's rootkit, CNET News.com, November 13, 2005.

[5] F-Secure, The Threat – Rootkits, Helsinki, Finland (www.virus.fi/blacklight/rootkit.shtml).

[6] J. Heasman, Implementing and Detecting a PCI Rootkit, Next Generation Security Software, Sutton, United Kingdom, 2006.

[7] G. Hoglund and J. Butler, *Rootkits: Subverting the Windows Kernel*, Addison-Wesley, Boston, Massachusetts, 2005.

[8] Holy_Father, Hacker Defender (hxdef), 2005.

[9] S. King, P. Chen, Y. Wang, C. Verbowski, H. Wang and J. Lorch, SubVirt: Implementing malware with virtual machines, *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 314–327, 2006.

[10] J. Levine, B. Culver and H. Owen, A methodology for detecting new binary rootkit exploits, presented at the *IEEE SouthEastCon Technical Conference*, 2003.

[11] J. Levine, J. Grizzard and H. Owen, Detecting and categorizing kernel-level rootkits to aid future detection, *IEEE Security & Privacy*, vol. 4(1), pp. 24–32, 2006.

[12] K. Mandia, C. Prosise and M. Pepe, *Incident Response and Computer Forensics*, McGraw-Hill/Osborne, Berkeley, California, 2003.

[13] S. McClure, J. Scambray and G. Kurtz, *Hacking Exposed: Network Security Secrets and Solutions*, Osborne/McGraw-Hill, Berkeley, California, 2001.

[14] Rootkit.com, AFX Rootkit (www.rootkit.com).

[15] Rootkit.com, FU Rootkit (www.rootkit.com).

[16] Rootkit.com, FUto Rootkit (www.rootkit.com).

[17] Rootkit.com, Vanquish Rootkit (www.rootkit.com).

[18] J. Rutkowska, Introducing Stealth Malware Taxonomy, Technical Report, COSEINC Advanced Malware Labs (invisiblethings.org/papers/malware-taxonomy.pdf), 2006.

[19] Sourceforge.net, Back Orifice 2000 (www.bo2k.com).

[20] Tech Support Alert, Rootkit Detection and Removal (www.pcsupportadvisor.com/rootkits.htm), 2006.

[21] A. Todd, J. Benson, G. Peterson, T. Franz, M. Stevens and R. Raines, Analysis of tools for detecting rootkits and hidden processes, in *Advances in Digital Forensics III*, P. Craiger and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 89–105, 2007.