

A COMPARATIVE ANALYSIS OF ROOTKIT DETECTION TECHNIQUES

by

THOMAS MARTIN ARNOLD, B.S.

THESIS

Presented to the Faculty of  
The University of Houston Clear Lake

In Partial Fulfillment  
of the requirements  
for the Degree

MASTER OF SCIENCE

THE UNIVERSITY OF HOUSTON-CLEAR LAKE

May, 2011

A COMPARATIVE ANALYSIS OF ROOTKIT DETECTION TECHNIQUES

by

THOMAS MARTIN ARNOLD

APPROVED BY

---

T. Andrew Yang, Ph.D., Chair

---

Sharon Hall, Ph.D., Committee Member

---

Sadegh Davari, Ph.D., Committee Member

---

Dennis M. Casserly, Ph.D., Associate Dean

---

Zbigniew J. Czajkiewicz, Ph.D., Dean

## DEDICATION

*I would like to dedicate this thesis to my family and friends for their constant love, support, and patience throughout the process of completing my graduate degree.*

## ACKNOWLEDGEMENTS

I would like to thank Larissa, Jonah, and Wesley for their love and patience as I worked many nights and weekends to complete my Computer Science graduate degree while working full-time.

I also want to thank Dr. T. Andrew Yang for being agreeing to be my thesis committee chair and for providing constant guidance and advice throughout the process of developing the thesis, performing the research, and presenting the results. I would also like to greatly thank Dr. Sadegh Davari and Dr. Sharon Perkins-Hall for serving on my thesis committee and for providing feedback during the past year. They are all fantastic professors and have directly provided me with a solid Computer Science education here at The University of Houston – Clear Lake.

## ABSTRACT

### A COMPARATIVE ANALYSIS OF ROOTKIT DETECTION TECHNIQUES

Thomas Arnold, M.S.  
The University of Houston, Clear Lake, 2011

Thesis Chair: Dr. T. Andrew Yang, Ph.D.

A rootkit is a type of malware that is designed to gain administrator-level control 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.

## TABLE OF CONTENTS

ABSTRACT .....	iii
1.0 Introduction.....	1
1.1 Research Objectives.....	3
1.2 Related Work .....	4
1.3 Potential Benefits.....	5
1.4 Thesis Outline .....	5
2.0 Survey of Rootkit Techniques .....	7
2.1 File Masquerading .....	7
2.2 Hooking.....	8
2.3 DKOM.....	9
2.4 Routine Patching.....	11
2.5 Filter Drivers .....	11
2.6 Hardware-Based .....	12
3.0 Survey of Rootkit Detection Techniques.....	13
3.1 Signature-Based .....	13
3.2 File Integrity Monitoring.....	14
3.3 Hooking Detection .....	14
3.4 Cross-View Analysis .....	15
3.5 Network-Based Detection.....	16
3.6 Heuristics-Based Detection.....	16
4.0 Research Methodology .....	18
4.1 Forensic Analysis .....	18
4.2 System Scanning Procedure .....	20
4.3 Network Scanning Procedure.....	20
5.0 Description of Rootkits used in Research .....	23
5.1 Rustock .....	23
5.2 TDL3 .....	24
5.3 Black Energy .....	25
5.4 Zeus/Zbot.....	27
6.0 Description of Rootkit Detectors used in Research.....	30
7.0 Experimental Results .....	64
7.1 System Performance and Forensic Analysis .....	64
7.1.1 Filesystem/Registry Modifications .....	65
7.1.2 Processor Utilization .....	66
7.1.3 Network Utilization .....	71
7.2 Anti-Rootkit System Scans.....	77
7.3 Network-Based Detection.....	89
8.0 Conclusions .....	98
8.1 Future Research .....	99
REFERENCES .....	100

## LIST OF FIGURES

Figure 1. Windows OS Architecture [41] .....	1
Figure 2. Windows OS Memory Protection Rings [12].....	2
Figure 3. Potential Hooking Locations in Windows [37] .....	9
Figure 4. DKOM EPROCESS example .....	10
Figure 5. Windows Device Driver Stack .....	12
Figure 6. Black Energy Use of Spare SSDT .....	26
Figure 7. Zeus/Zbot login form injection [34] .....	29
Figure 8. Antiy Atool.....	33
Figure 9. Avast! Antivirus.....	34
Figure 10. AVZ Antivirus .....	35
Figure 11. CMC Codewalker.....	36
Figure 12. ESET SysInspector.....	38
Figure 13. F-Secure Internet Security .....	39
Figure 14. GMER rootkit detector .....	41
Figure 15. Helios.....	42
Figure 16. HiddenFinder.....	43
Figure 17. IceSword .....	44
Figure 18. Kernel Detective .....	46
Figure 19. K X-ray.....	47
Figure 20. Kaspersky Internet Security .....	48
Figure 21. Malwarebytes Anti-Malware.....	50
Figure 22. McAfee Rootkit Detective .....	51
Figure 23. Microsoft Security Essentials .....	52
Figure 24. Panda Internet Security .....	53
Figure 25. Rootkit Revealer .....	54
Figure 26. Rootkit Unhooker.....	55
Figure 27. RootRepeal .....	56
Figure 28. Sophos Anti-Rootkit .....	57
Figure 29. Spybot Search & Destroy.....	58
Figure 30. Moosoft The Cleaner .....	60
Figure 31. Trend Micro Rootkit Buster.....	61
Figure 32. VBA32 .....	62
Figure 33. XueTr Diagnostic Tool.....	63
Figure 34. Processor Utilization for an Uninfected Machine .....	67
Figure 35. Processor Utilization for a TDL3-infected Machine.....	68
Figure 36. Processor Utilization for a Rustock-infected Machine.....	69
Figure 37. Processor Utilization for Black Energy-infected Machine .....	70
Figure 38. Processor Utilization for a Zeus-infected Machine .....	71
Figure 39. Outbound Network Traffic for a TDL3-infected Machine.....	73
Figure 40. Outbound Network Traffic for a Rustock-infected Machine .....	74
Figure 41. Network Utilization for a Black Energy-infected Machine .....	75
Figure 42. Network Utilization for a Zeus-infected Machine.....	76
Figure 43. Microsoft Security Essentials Detection of TDL3 .....	80
Figure 44. MBAM Detection of Rustock Driver .....	82
Figure 45. Black Energy Detection by Rootkit Unhooker .....	84
Figure 46. Clean System Netstat Output .....	90

Figure 47. Clean System Nmap Port Scan .....	90
Figure 48. Hacker Defender Netstat Output .....	91
Figure 49. Hacker Defender Nmap Port Scan .....	92
Figure 50. TDL3 Netstat Output .....	93
Figure 51. TDL3 Nmap Port Scan .....	93
Figure 52. Rustock Netstat Output .....	94
Figure 53. Rustock Nmap Port Scan.....	95
Figure 54. Black Energy Netstat Output .....	96
Figure 55. Black Energy Nmap Port Scan .....	96
Figure 56. Zeus Netstat Output.....	97
Figure 57. Zeus Nmap Port Scan.....	97

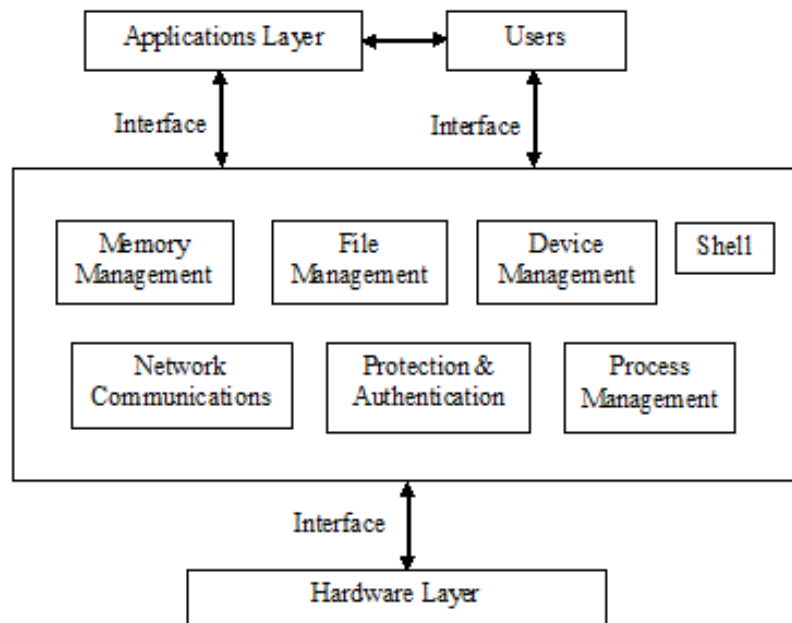


## LIST OF TABLES

Table 1. Anti-Rootkit Software Characteristics .....	32
Table 2. Nominal (Clean) Anti-Rootkit Scan Results.....	78
Table 3. TDL3 Anti-Rootkit Scan Results .....	79
Table 4. Rustock Anti-Rootkit Scan Results.....	81
Table 5. Black Energy Anti-Rootkit Scans .....	83
Table 6. Zeus/Zbot Anti-Rootkit Scans.....	85
Table 7. Overall Ranking of ARK Tools .....	87

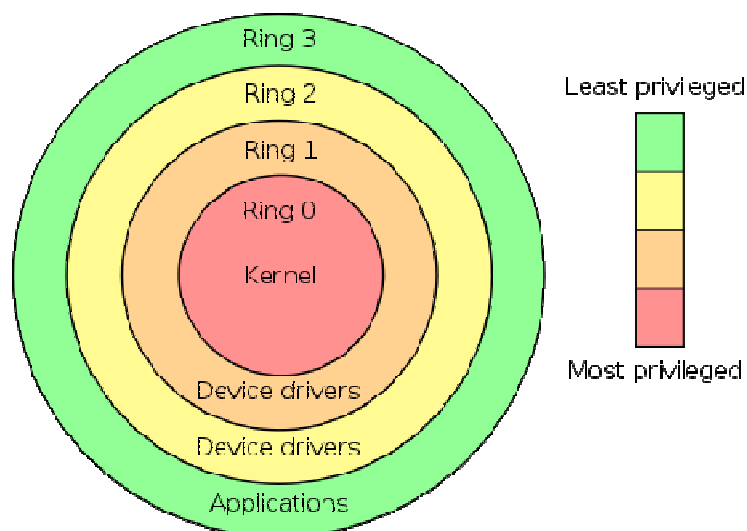
## 1.0 Introduction

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 shielded from the hardware details by a number of software layers in the Windows OS. The layering provides a high level of portability and extensibility, but at the same time creates a number of opportunities for attackers to compromise the system. If one of the communication paths between the layers is controlled by a malicious user, the attacker 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 detected by the user or the OS. Rootkits focus on these communication paths and interfaces to conceal their presence on the OS.



**Figure 1. Windows OS Architecture [41]**

The Intel IA32 and IA64 architectures provide several different levels of memory protection, often known as “rings”, shown in Figure 2. The rings are numbered 0 through 3, with Ring 0 representing the highest privilege level and Ring 3 representing the lowest. Rings 1 and 2 represent privilege levels that could be used by device drivers and user programs with I/O access permissions, respectively. The idea is that system code and data can be protected from being overwritten by a program running at a lower privilege level. Windows does not take advantage of all 4 levels of protection, instead focusing the OS operation only in Ring 0 (Kernel mode) and Ring 3 (User mode). This is an artifact of previous hardware architectures that Windows NT was designed to support, such as Compaq Alpha and Silicon Graphics MIPS which implemented only two privilege levels [39]. At a high level, Ring 3 users are limited to using the Application Programming Interface (API) to interface with the OS kernel, and Ring 0 users can interface directly with the memory and hardware.



**Figure 2. Windows OS Memory Protection Rings [12]**

Typically rootkit authors gain Ring 0 status by implementing the rootkit as a Kernel Mode Driver (KMD) [18]. The implications for having Ring 0 access are extremely serious. As described earlier, a kernel mode rootkit can interface directly with the OS internal structure, performing any number of malicious activities and hiding itself from the users and applications at the same time.

In spite of the serious threat posed by kernel mode rootkits, they were only estimated to occur in approximately 7% of all reported malware infections as of January 2010 [45]. However, the impact is still fairly large in terms of malicious activity. For example, in the second half of 2009, Microsoft estimated that the botnet enabled by the W32/Rustock rootkit was responsible for 39.7% of the over 400 billion spam emails that were detected by their servers [8].

## **1.1 Research Objectives**

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

performed. Additionally, forensic analysis of several different types of modern rootkits will be performed.

## **1.2 Related Work**

Anti-Virus Comparatives [2] is an independent organization that performs regular comparison testing of Anti-Virus software. Their testing methodology is very thorough, as they use the latest copies of almost all available Anti-Virus 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 thesis to provide valuable information to the community.

Yegulalp [48] provided a good functional description and comparison 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 [28] 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 outdated test set, so the relevancy of the results is questionable.

Finally, none of the AV comparison tests that have been performed on rootkits have attempted to compare which techniques appear to be most successful, which is the focus of this thesis.

### **1.3 Potential Benefits**

The development of rootkits and rootkit detectors is a constantly 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 thesis will help bring to light the fact that many formerly effective solutions have not kept up with the pace 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 effectively.

Additionally, a set of computer system and malware forensic analysis skills will be developed during the course of the research. This thesis will document how debugging tools and other analysis tools can be used in the analysis of recently-developed rootkits.

### **1.4 Thesis Outline**

The rest of the thesis is structured as follows: First, in sections 2 and 3, an overview of rootkit design techniques and detection methods is given. In section 4, the research methodology is described. In section 5, details of the specific rootkits used in the research will be provided. In section 6, a description of each of the rootkit detection software will be given. In section

7, the results of the experiments will be provided, as well as a discussion and analysis of the results. In section 8, conclusions will be provided, including proposed future research.

## **2.0 Survey of Rootkit Techniques**

This section will provide an overview 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 engaging in a constant process of one-upmanship. The initial rootkits focused on UNIX-based systems, and used fairly primitive 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 same name and services as the original. This technique is known as file masquerading [41]. For example, a system file that provides a service or function 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 [15], 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.

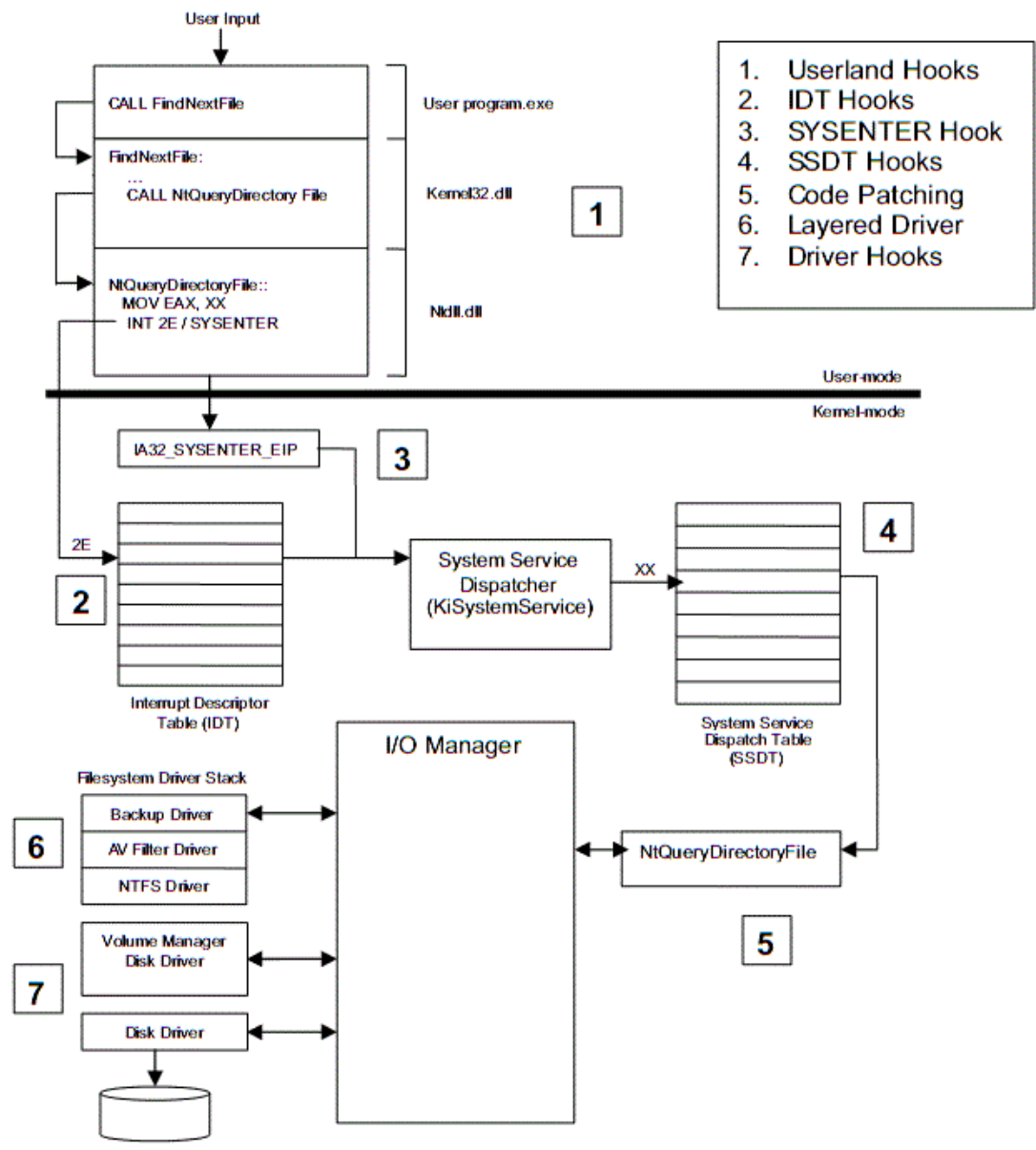


## 2.2 Hooking

The next step in the evolution of rootkits was to redirect system calls to malicious code, a technique known as “hooking” [41]. 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 redirected to a custom “dir” command in memory space that filters out the malicious files and folders.

Basically, hooking achieves the same effect as file masquerading, but is more difficult to detect, since the system files on disk are not altered. This type of technique cannot be detected by file integrity checkers as described in the previous section, so in order to counter this technique, memory scanners such as Rootkit Unhooker [13] were developed.

Figure 3 shows a typical path of a Windows-based function call starting at the user application and ending in the physical hardware. There are several different locations along the way that can be hooked to perform both malicious and legitimate activities. 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 [37]. These tables maintain memory addresses that point to various functions and interrupt request handlers, which can be modified to point to malicious programs that are resident in memory.

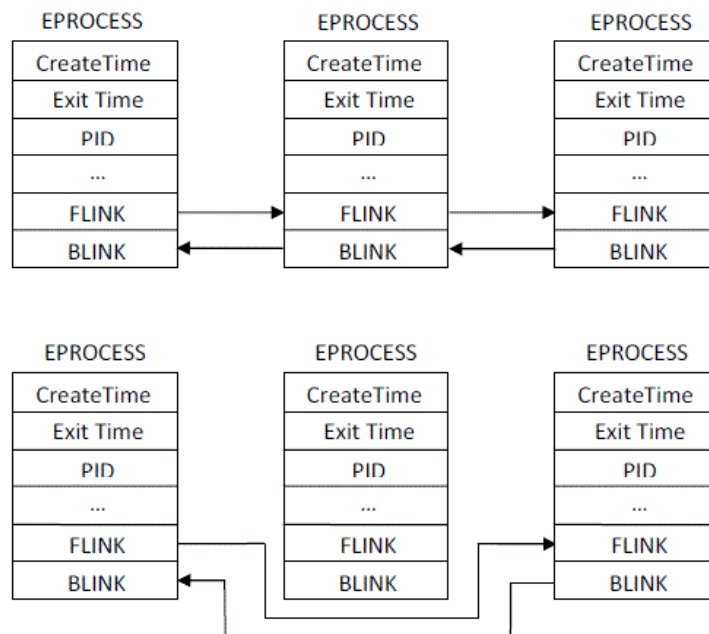


**Figure 3. Potential Hooking Locations in Windows [37]**

### 2.3 DKOM

The third generation of rootkits used technique known as Direct Kernel Object Manipulation (DKOM). DKOM can manipulate kernel data structures

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 processes and threads in Windows. As shown in Figure 4, by modifying the FLINK and BLINK 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 [37]. 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 proprietary nature of the Windows source code.



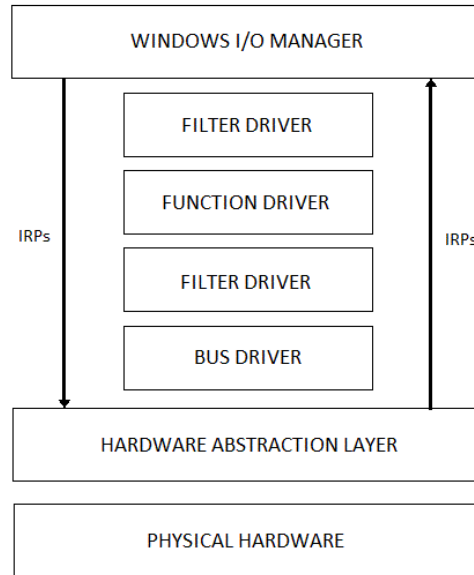
**Figure 4. DKOM EPROCESS list modification**

## **2.4 Routine Patching**

In this technique, the rootkit author modifies the source code of a system routine to cause the execution path to jump to malicious code which is resident either in memory or on disk. Some of the early UNIX-based rootkits completely replaced the system file with a modified version using the same name [41]. Modern Windows-based rootkits may embed a JMP instruction within the system binary to redirect the execution path [18]. This can be performed against the system binaries stored in the OS file system, or even against executing code loaded in memory. If the modification was performed on the file system, this can be easily detected by file integrity monitoring systems. Run-time modification can be detected by applications such as Kernel Path Protection, which is provided by the 64-bit versions of Windows.

## **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 [41]. 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 keystroke logging or filtering the results that are returned to anti-malware applications. Rootkit authors can perform hooking of drivers, patch driver routines, or even create an entirely new driver and insert it into a driver stack. Figure 5 shows a representative example of a Windows Driver stack.



**Figure 5. 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 BIOS or PCI expansion cards [41]. 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 [41].

### **3.0 Survey of Rootkit Detection Techniques**

There exist a number of different methods to detect rootkits, including signature-based detection, file integrity monitoring, cross-view analysis, hooking detection, heuristics (behavior)-based detection, and network-based detection. Most of the rootkit detectors employ several of these techniques, in order to provide the widest range of capabilities and increase their chances of success. In this section each of these techniques will be briefly described, and include some examples of current software that uses them. One caveat for all of the techniques described in this section is that a kernel mode rootkit can always alter the results that are reported to the anti-rootkit software, and the lack of a reported detection is not always indicative of a clean system. However, in practice, many rootkit authors do not always include anti-detection or anti-forensics code in the malware, due to large time and effort required to thoroughly address all the potential detection methods [22].

#### **3.1 Signature-Based**

The most common method for detecting rootkits (and malware in general) is the signature-based technique [22]. Once a sample of malware has been obtained, the byte pattern of the software is heavily analyzed to identify a unique fingerprint that will distinguish this specific malware from legitimate software, as well as other types of malicious software. The fingerprint "signature" will then be integrated into a database that can be used by detection software when performing system scanning. If a scanned piece of software has a pattern that matches an entry in the malware database, it is

extremely likely that it is malicious and should be flagged to the system and the user. While this technique has been successfully used for over twenty years, the main weakness is that it cannot detect new types of malware, until a sample can be analyzed to extract a signature. Popular programs that employ signature-based detection include the Microsoft Malicious Software Removal Tool [7], Kaspersky Internet Security [5], and Malware Bytes Anti-Malware [6], and many others.

### **3.2 File Integrity Monitoring**

File integrity monitoring is a detection technique that was first employed on UNIX systems by Tripwire in the early 1990s [15]. The method calculates cryptographic hashes for critical, unchanging operating system files and compares them to known values that are stored in a database. Typically this database is generated against a clean version of the operating system, so when a mismatch is detected, a file has been altered (likely by malicious software). This technique works well against the file masquerading rootkit design as described in Section 2.1.1, but rootkit authors quickly adapted to use hooking techniques instead. As a result, file integrity monitoring is not widely employed as a method of detection for modern anti-rootkit systems.

### **3.3 Hooking Detection**

Detection of rootkit hooking is a fairly straightforward process. The SSDT, IAT, and IDT each has a set of function pointers for each service or interrupt, which are all within a specific range in memory. When the rootkit modifies a hook to point to a malicious service or interrupt routine, the memory location

almost invariably is located outside this specific range of the “clean” system, and is easily detected by anti-rootkit software. Inline function and I/O Request Packet (IRP) hooking is detected in essentially the same manner. While hooking is easily detected, it should be noted that a kernel-mode rootkit can alter the results of the detection software and make it appear that everything is nominal. Hooking detection is thoroughly provided in tools such as Rootkit Unhooker [13] and GMER [24].

### **3.4 Cross-View Analysis**

The next detection technique to be discussed is known as cross-view analysis. It involves looking at the system from the high level “user”, or API view, and comparing it to the actual low level hardware view. The idea is that a rootkit will not be able to hide itself when the raw hardware is scanned. If a particular file or registry key is absent from the API view but is present in the hardware view, it is highly likely that a rootkit is attempting to hide itself from the system. This technique was first used in SysInternals’ RootkitRevealer software [38], and is now used in many other detectors such as IceSword [4].

Detection of DKOM is more challenging than the signature-based techniques or hook detection as described earlier in Section 2.1.3, because most of the time the OS data structures that have been modified are not very well documented by the vendor to begin with (usually for proprietary reasons). The typical method to detect DKOM is to look for other locations in the OS kernel where the same data may be stored, and perform a comparison. If



any discrepancies are found, it is likely that a system has been modified by a rootkit. Rootkit detectors that provide DKOM detection include Rootkit Unhooker [13] and GMER [24], as well as several others.

### **3.5 Network-Based Detection**

A novel technique developed by Symantec researchers to detect the presence of a rootkit is to analyze the network traffic of the system [44]. In [44], Szor proposes to have the system periodically send a snapshot of the network traffic and open ports to a trusted gateway for analysis. The gateway will compare this data with its "external" view of the system's network activity. If there are ports that are not being reported as open, or traffic that is not being reported by the host system, but is observed by the gateway, then the host system is likely infected with a rootkit. At this time, this method of detection appears to not be used by Symantec in any of their products. It is an elegant solution that has a lot of potential to uncover "zero day" rootkits that would not otherwise be detected by the traditional signature-based techniques. It should be noted that if the rootkit author communicates via covert channel techniques, this technique would not be as effective.

### **3.6 Heuristics-Based Detection**

Heuristics-Based detection of malware attempts to classify malicious behavior according to certain pre-determined rules. For example, an application that attempts to modify kernel-data structures, decrypt instructions, or send a large amount of email in a short period of time likely has malicious intent. One significant advantage of this detection method is that "zero-day"

variants of malware can be detected, which is the weakness of the signature-based detection method. However, the big drawback to the heuristics-based method is that more false positives can be generated, thus the definition of the ruleset must be developed very carefully. Several malware detectors in this study utilize heuristics-based detection algorithms, including Malwarebytes Anti-Malware and F-Secure Internet Security.

## **4.0 Research Methodology**

All thesis research experiments were performed in the Distributed Computing Systems Laboratory (DCSL) at the University of Houston – Clear Lake campus. The workstations used at the DCSL provided the ability to perform simultaneous operation of several different families of malware in an isolated environment. Another benefit of utilizing the DCSL workstations is that the malware can operate in a real environment, as certain types of malware will not function in a virtualized system.

In order to provide a common baseline system, Windows XP Service Pack 3 was installed on all the workstations. The Partimage Is Not Ghost (PING) [10] application was used to ensure a consistent disk image was used across all machines, as opposed to manually installing Windows XP and the associated Service Packs. Also, to ensure that the disk image was not tainted, Derek's Boot and Nuke [27] was used to zero out the hard drive sectors prior to installation of the OS image.

The workstations in the DCSL have identical specifications, which include an Intel Pentium 4 3GHz processor, 1 GB of Random Access Memory, and a 112 GB portable hard drive.

### **4.1 Forensic Analysis**

In order to analyze the effect of a given rootkit on the system, a number of forensic experiments were performed. First, a comparison was performed of the filesystem and Windows registry before and after infection to evaluate modifications by the associated rootkit. Also, CPU utilization measurements,

as well as network utilization data was collected for a long duration (approximately 48 hours). Finally, once a system was infected with a rootkit, a kernel mode debugging session was performed using either the Microsoft Kernel Debugger (KD.exe) or Windbg.exe to analyze the changes to internal Windows OS structures.

The changes to the filesystem and Windows registry were evaluated by using several different software applications. First, the hard disk was wiped using DBAN and the Windows OS was installed using PING as described in Section 4.0. Once the OS was installed, a live CD (Bart's Preinstalled Environment, or BartPE) [31] was used to boot Windows and view the filesystem from an external perspective. Even with a rootkit installed, since the infected OS is not running, any modifications to the filesystem or registry will be apparent. After the BartPE successfully loaded Windows, the contents of the filesystem (directory and filenames only) were dumped to a text file. Additionally, the Windows registry was saved for later comparison. Next, the system was rebooted and the rootkit was installed. After the rootkit was verified to be installed, the filesystem and Windows registry were copied again using another BartPE session. In order to compare the filesystem changes, Windiff.exe [16] was used to highlight any modifications that occurred after the rootkit infection. Windows registry changes were evaluated using AlienRegistryViewer [1] to import the individual registry files and save them as a single .reg file, and RegSnap [11] was used to perform the actual comparison.

## **4.2 System Scanning Procedure**

In order to verify that the rootkit was installed, a kernel mode debugging session was performed, and certain behaviors such as characteristic TCP/IP traffic and web browser redirects had to also be observed. Once the portable DCSL drive was verified to be infected with the rootkit, each of the Anti-rootkit software packages was individually installed and scans were performed on the infected drive. If the rootkit was detected, and the ARK software provided an option to remove the rootkit, then it was attempted. Removal was confirmed by subsequent scans of corroborating tools, and verifying the lack of certain symptoms such as TCP connections and URL redirects. If removal was attempted, then the drive was subsequently wiped with DBAN and the OS installation/infection/scan process was continued with the remaining rootkit detectors.

## **4.3 Network Scanning Procedure**

In this experiment, two independent systems were used. Both machines used Windows XP Service Pack 3 installed on portable drives in the Distributed Computing Systems Laboratory, as described Section 4.0. One of the machines is infected with a rootkit from the thesis research (TDL3, Rustock Black Energy, or Zeus), and the other machine is used as a clean system to perform the external network port scans. To demonstrate the efficacy of the network-based detection technique, a rootkit that is known to hide network ports (Hacker Defender) was used as a control. Additionally, the technique was performed against an uninfected machine to provide a clean baseline.

The port scanning software used by the external host in the research was Network Mapper (Nmap), a freely available security scanner [33]. Nmap has the ability to perform many scanning functions against remote network hosts, including determining the status of TCP and UDP ports, the services that are attached to open ports, the remote host's operating system, as well as many other functions. For the purposes of this experiment, the port scanning function will be the primary use of Nmap. One limitation of Nmap is that it is only able to detect open or listening ports, due to the 3-way handshake of a TCP connection. Ports that are already connected cannot be detected by Nmap.

The network software used by the internal (infected) host was Netstat. Netstat [9] is a network statistics application that is automatically included with all versions of Windows, as well as many other operating systems such as UNIX, Linux, and Macintosh OS X. Netstat displays all incoming and outgoing network connections, and includes information such as the connection state and the local and foreign IP addresses associated with each connection.

In order to identify the potential presence of a rootkit, the output of an Nmap scan against the target infected machine is compared to the output from Netstat running on the infected machine. Netstat is a command-line application, so the output can be directed to a file using the ">>" operator in the Windows command line. In order to perform a thorough scan of all ports on the remote host, Nmap enumerated through all 65535 potentially

open TCP and UDP ports. In this set of port scans, Nmap determines all open or listening TCP/UDP ports on the target host, and if possible Nmap also determines the service associated with each connection. This output is also saved in a log file. The log file from Netstat is compared with Nmap to look for discrepancies. If there are ports that are included in the Nmap output and are not visible in the Netstat output, then it is highly likely that a rootkit is hiding its network connections.

## Procedure

1. On the local (infected) machine, open a command line window and determine the local IP address by executing the ***ipconfig*** command.
2. Next, again on the local (infected) machine, open a command line window and execute the ***netstat -a -n >> [rootkit]\_netstat.txt*** to enumerate all active TCP and UDP connections and direct the output to a log file, where [rootkit] is replaced by the name of the rootkit that the machine has been infected with (eg., TDL3).
3. On remote (clean) machine, open the Zenmap application, which is the Windows-based GUI for Nmap. In the "Target" textbox, enter the IP address of the infected machine that was obtained in Step 1. Under profile, select Intense Scan, and click the "Scan" button.

Manually compare the output of Nmap from the remote scan against the Netstat output from the local scans. Look for any discrepancies, in particular look for additional ports in Nmap that were not reported in Netstat.

## **5.0 Description of Rootkits used in Research**

### **5.1 Rustock**

The Rustock rootkit/botnet has been in existence since the 2006 timeframe, and has been in a constant state of evolution. The primary focus of the botnet is to distribute large quantities of spam email, although there are some reports that it has also been used to perform Distributed Denial of Service attacks [43]. As of July 2010, the botnet was responsible for approximately 50% of the total spam production, at a rate of approximately 30 billion spam messages a day [36]. There have been recent reports that the rate of spam production has slowed dramatically with the shutdown of Spamit.com, a large affiliate that specializes in pharmaceutical spam [26]. It is not clear whether or not this slowdown in production is simply a temporary phenomenon.

Rustock has undergone a number of major design evolutions over the years. Initially, the rootkit focused on performing System Service Dispatch Table (SSDT) hooking to hide the driver and associated registry keys. More recent versions has moved away from SSDT hooking (due to the ease of detection), and instead utilize a filter driver by hooking the IRP\_MJ\_CREATE routine in the ntfs.sys driver, which intercepts I/O Request Packets to and from the hard disk. Additionally, a recent update to the spambot component of Rustock includes communication with a random Wikipedia entry to download random phrases which can be used to evade spam email detectors [36].



On March 17, 2011, Microsoft announced that the Rustock botnet had been taken offline by a combination of legal and technical strategies. This takedown was a joint effort between Microsoft, FireEye Security, and the University of Washington [46]. The perpetrators of the botnet have not been apprehended, however several pieces of physical evidence, including hard drives from Command and Control servers located in the United States were recovered [32]. Even though the botnet has been effectively eliminated, is very possible that the malware authors will attempt to recreate a similar criminal enterprise in the future.

## **5.2 TDL3**

TDL3 is the 3<sup>rd</sup> generation of “Trojan Downloader” rootkits developed by the Dogma Millions cybercrime group [35]. The malware is used in a “Pay Per Install” scheme, which uses distributor identification to determine how many copies of the malware get installed on computers. The ultimate goal of the TDL3 rootkit is to download, install, and hide malicious programs that can perform illicit activities such as keystroke monitoring or Distributed Denial of Service (DDoS) attacks.

The rootkit installs itself via an exploit in the Windows AddPrintProcessor API call [3]. Basically, the malware adds itself as a new printer and as a result gets kernel-mode driver privileges. From there, the malware performs filter driver hooking as described in Section 2.5 and infects the hard-drive miniport driver for atapi.sys, as well as a randomly chosen driver that is loaded at boot-time. Also, the rootkit installs an encrypted filesystem that begins at

the end of the hard disk, and grows toward the beginning [3]. This way, the filesystem is outside of the range of the Windows filesystem and therefore is not detected via traditional scanning techniques. The TDL3 configuration files as well as the downloaded malware programs are stored in this encrypted filesystem [40].

TDL does not perform some of the more traditional rootkit techniques such as System Service Descriptor Table (SSDT) or Interrupt Descriptor Table (IDT) hooking or even Direct Kernel Object Manipulation (DKOM). The rootkit does perform I/O Request Packet (IRP) hooking to intercept and filter IRPs that are sent and received by the hard-drive miniport driver. The IRP hooking provides communication with the encrypted filesystem that was described in the previous paragraph. From a user perspective, many browser search requests to security websites are redirected to either malicious or heavily ad-supported websites that attempt to install other types of malware.

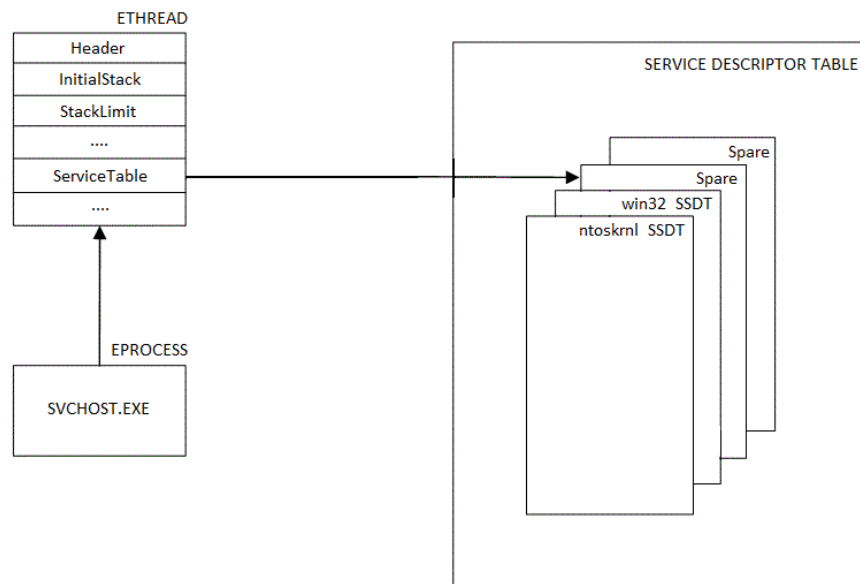
Overall, the TDL3 rootkit is one of the most sophisticated and actively-developed rootkits today. It performs active blocking of many prevalent anti-malware tools, and utilizes hiding techniques that many of the Anti-Rootkit detectors do not look for.

### **5.3 Black Energy**

The Black Energy Rootkit has been in existence since approximately 2007, when it was used to perform Distributed Denial of Service (DDoS) attacks against the country of Georgia [42]. The software was initially designed to

perform solely DDoS-type of attacks, but recently the rootkit has been updated to perform many other activities, including bank fraud [25].

The most recent version of Black Energy injects code into a svchost.exe process and also includes sophisticated methods of hiding itself from conventional rootkit detectors. This version, known as "2.1+", exploits the Windows operating system System Service Descriptor Table (SSDT) architecture [29], as shown in Figure 6.



**Figure 6. Black Energy Use of Spare SSDT**

The baseline Windows OS utilizes 2 SSDTs, although there are provisions built into the OS for 4 total SSDT, thus 2 are typically unused [39]. The primary SSDT consists of approximately 300 "system calls" to various services for opening files, terminating processes, etc. The secondary SSDT, known as the "shadow" SSDT, includes many system calls for the Windows Graphical User Interface (GUI) [39]. The recent Black Energy rootkit copies

these two SSDTs and points to them in their respective ETHREAD objects which are generated by the svchost.exe process [29]. These SSDT copies include hooks in various system calls to hide the rootkit components and control the relevant features of the OS that the author deemed necessary for operation. Conventional rootkit detectors such as IceSword may only report the hooking status of the primary two SSDTs, and not look for utilization of the other SSDT slots, since they are not typically used. Therefore, if the Anti-Rootkit (ARK) software only looks at the primary SSDTs, it may not detect the presence of Black Energy. Some ARK tools such as GMER and Rootkit Unhooker (and others) perform a comparison of the SSDTs which are pointed to by the active ETHREAD objects, and if there is a miscompare, the discrepancy is reported. This difference in ARK operation is illustrated in the results of the ARK scans, which are included in later sections.

#### **5.4 Zeus/Zbot**

Zeus/Zbot is a family of malicious software that focuses on stealing passwords for financial institutions, and includes several rootkit components to provide stealth capabilities. The Zeus malware, which originated in Russia, has been in existence since 2007 [23], and is continuously being updated. It is one of the largest botnets in existence, affecting approximately 75,000 computers in over 200 countries [20]. It is possible to purchase a Zeus "bot-maker" kit on underground Internet forums, which can be used to generate malware that is distributed to victims via drive-by downloads or spam email campaigns. The primary goal of the Zeus malware is to steal passwords and sensitive information for web-based financial

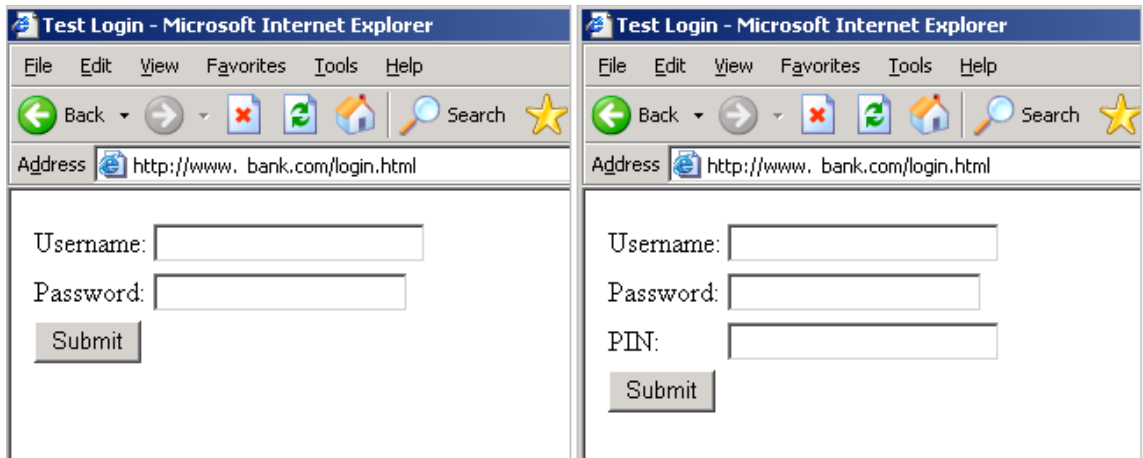
accounts, which are then used to transfer stolen money to criminals [30].

There have been a large number of recent arrests of criminals using the Zeus malware to steal personal information, but this has not slowed down the overall development or illicit activity associated with this malware.

A detailed analysis of the malware characteristics on a Windows-based system is provided in [17]. The malware, which is typically installed via drive-by download or by a user clicking on malicious links in a spam or phishing email, performs a number of modifications to the Windows Operating System (OS). The malware installs a copy of the main driver file, *sdra64.exe*, in the *Windows/system32* folder, and is subsequently hidden via hooking Windows services. This program is then injected into the *winlogon.exe* or *Svchost.exe* process, which allows kernel-level access to the OS [17]. Next, the *Windows/system32/lowsec* folder is created, and the *local.ds* and *user.ds* files are copied into this folder. These files store the malware's configuration file as well as the user's stolen sensitive information. Both of these files are encrypted and hidden via hooking Windows services. A listening TCP port is also opened and associated with the injected *Winlogon.exe* or *Svchost.exe* process, which is a likely backdoor communication link to the botmaster. Finally, a registry entry is created to ensure the malware is initialized upon a restart of the Windows OS.

Once installed, the malware waits until a user logs into a financial website that is specified in the configuration file. It then injects predetermined code into the browser to include additional textboxes for the user to enter

sensitive information. The configuration file can be customized based on the user's location and language. An example of this injection is shown in Figure 7. The malware logs the sensitive information and transmits it to the botmaster via encrypted network traffic.



**Figure 7. Zeus/Zbot login form injection [34]**

## **6.0 Description of Rootkit Detectors used in Research**

In this section, a brief description of each of the Anti-Rootkit (ARK) tools is provided, as well as a summary of the respective tool's performance in the rootkit scanning as described in Section 4.2. Table 1 displays an overview of the scope and detection techniques used by each of the ARK tools included in the research.

At a high level, the ARK tools can be divided into two groups: diagnostic tools and malware scanners. The diagnostic tools tend to focus on reporting the current state of various components in the operating system, such as system call tables, loaded services, active network ports, etc. The results usually have to be interpreted by a knowledgeable user, since it may not be clear if there is a problem. Examples of diagnostic tools include ESET SysInspector, Ice Sword, or XueTr. The malware scanners typically provide a very intuitive user interface to initiate a system scan, and the results are likewise very clear if an instance of malware is detected. Examples of malware scanners include Kaspersky Internet Security and Malware Bytes Anti-Malware. Some of the tools, such as GMER and Rootkit Unhooker, incorporate both diagnostics and scanning into their operation, but most of the programs can be included in one of these two groups.

In Table 1, the "Active" column indicates whether the application was still being actively developed over the last calendar year. For example, Ice Sword has not been updated since approximately 2008, so it is not considered to be in active development any longer. The "Self Protect"

column indicates whether the application provides mechanisms to protect its operation from tampering by malware. For example, F-Secure hooks many Windows services such as `NtTerminateProcess` and `NtTerminateThread` to prevent malware from terminating it. The "Diag" column indicates whether the application is primarily diagnostic in nature, or more of an "on demand" scanner. If the application provides real-time protection from malware (always-on versus on-demand), that is indicated in the "Real-time Protection" column. For example, Kaspersky hooks a large number of Windows services to monitor the creation of new processes, registry keys, network ports, etc. in an effort to identify malicious behavior. If the detector also provides a removal capability, that is indicated in the "Removal" column. There are also columns that show the various malware detection methods that the application provides, including hooking, cross-view, heuristic-based, as well as signature-based. These features were determined by analyzing the user interface and output of the application, as well as any available documentation provided by the developer.



**Table 1. Anti-Rootkit Software Characteristics**

Anti-Rootkit Tool	Version	Active	Self Protect	Services Offered		Detection Methods			Signature	
				Diag	Real-Time Protection	Removal	Hooking	Cross-View		Memory Scanning
Atool	1.0021	No	No	Yes	No	No	Yes	Yes	No	No
Avast! Antivirus	0.9.6	Yes	Yes	No	Yes	Yes	No	Yes	Yes	Yes
AVZ Antivirus	4.35	Yes	No	Yes	Yes	Yes	No	Yes	No	Yes
CMC Antirootkit	0.2.4.500	No	No	Yes	No	No	Yes	Yes	No	No
ComboFix		Yes	No	No	No	Yes	No	Yes	No	Yes
ESET SysInspector	1.2.021.0	Yes	Yes	Yes	No	No	No	Yes	No	No
F-Secure Internet Security	2011	Yes	Yes	No	Yes	No	No	Yes	Yes	Yes
GEMER	1.0.15.15281	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No
Helios Lite	1.005	No	No	Yes	No	No	Yes	Yes	No	No
Hidden Finder	1.5.6	No	No	Yes	No	Yes	No	Yes	No	No
Ice Sword	1.22Zen	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
K X-ray	1.0.0.102	No	No	Yes	No	Yes	Yes	Yes	No	No
Kernel Detective	1.3.1	Yes	No	Yes	No	Yes	Yes	Yes	No	No
Kaspersky Internet Security	2011	Yes	Yes	No	Yes	Yes	No	Yes	Yes	Yes
Malware Bytes Anti-Malware	1.50	Yes	No	No	No	Yes	No	Yes	Yes	Yes
Microsoft Security Essentials	2.0.657.0	Yes	No	Yes	No	Yes	No	Yes	Yes	Yes
McAfee Rootkit Detective	1.1	No	No	No	No	Yes	Yes	No	No	No
Panda Internet Security	2011	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Rootkit Revealer	1.7	No	No	No	No	No	No	Yes	No	No
Rootkit Unhooker	3.8.388.480 SR2	Yes	Yes	Yes	No	Yes	Yes	Yes	No	No
RootRepeal	1.3.5	Yes	No	Yes	No	Yes	Yes	Yes	No	No
Sophos Antirootkit	1.5.4	No	No	No	No	Yes	No	Yes	No	No
Spv Bot	1.6.2.46	Yes	No	No	Yes	Yes	No	No	Yes	Yes
Moosoft The Cleaner 2011	7.2.0.3510	Yes	No	No	Yes	Yes	No	Yes	7	Yes
Trend Micro Rootkit Buster	2.80.0.1077	No	No	No	No	Yes	Yes	No	No	No
VBA_32	3.12.4.0	Yes	No	Yes	No	No	Yes	Yes	Yes	No
XeuTr	0.34	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No

## 6.1 Atool

Atool, shown in Figure 8, is a rootkit detector that is developed by Antiy Labs in China. This application would likely be most useful as a diagnostic tool for knowledgeable users, and the interface is not simple or intuitive enough for use by the average user. Additionally, Atool provides insight into a number of different areas of the Windows operating system, including running tasks, processes, services, and drivers, as well as the state of the SSDT and file system drivers. The current version number of Atool is 1.0021, and it has not been updated since 2008.

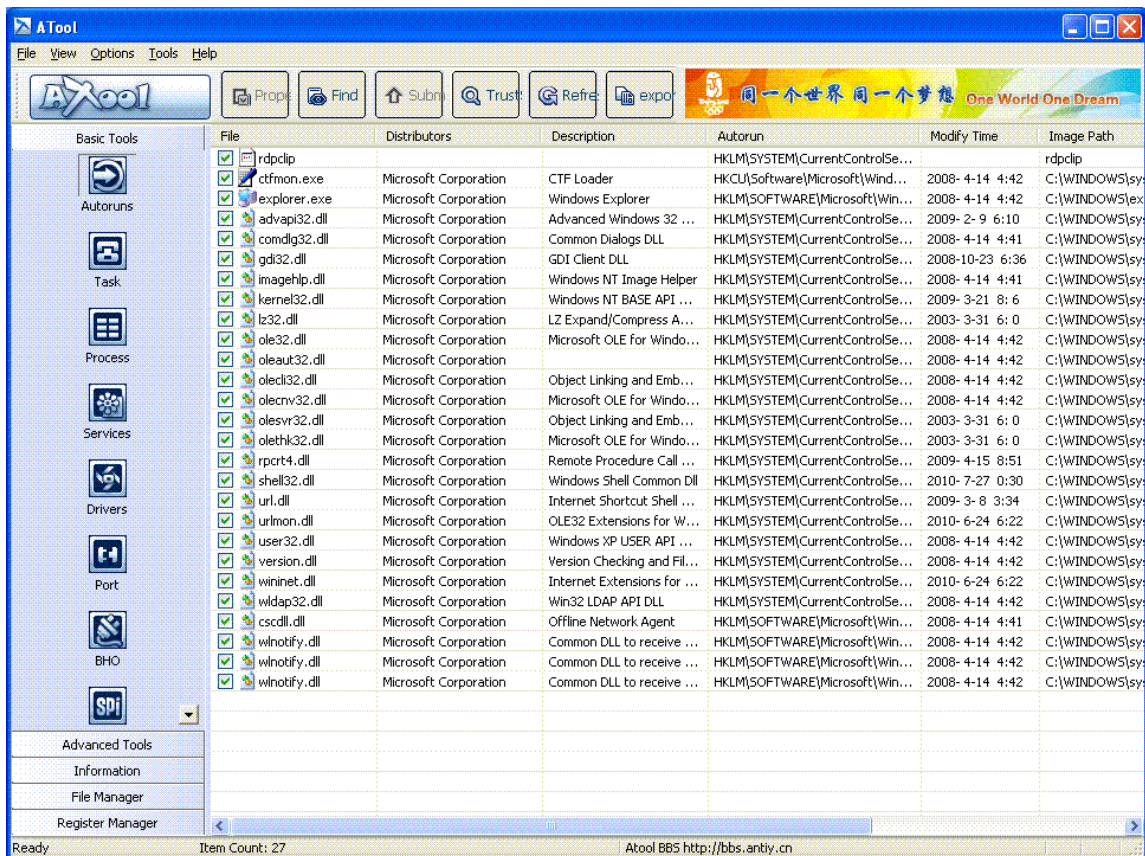
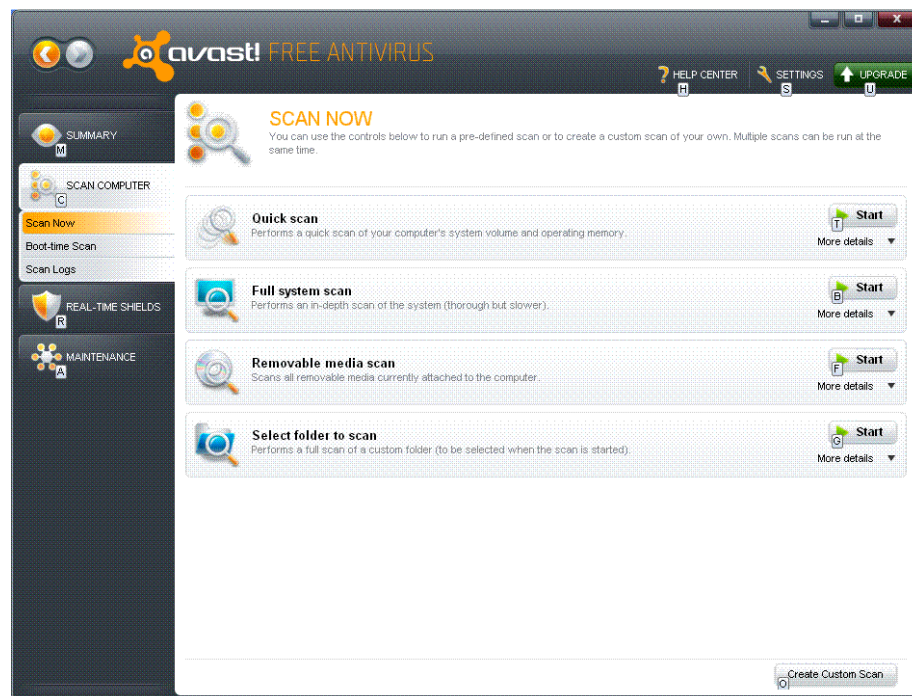


Figure 8. Antiy Atool

## 6.2 Avast! Antivirus

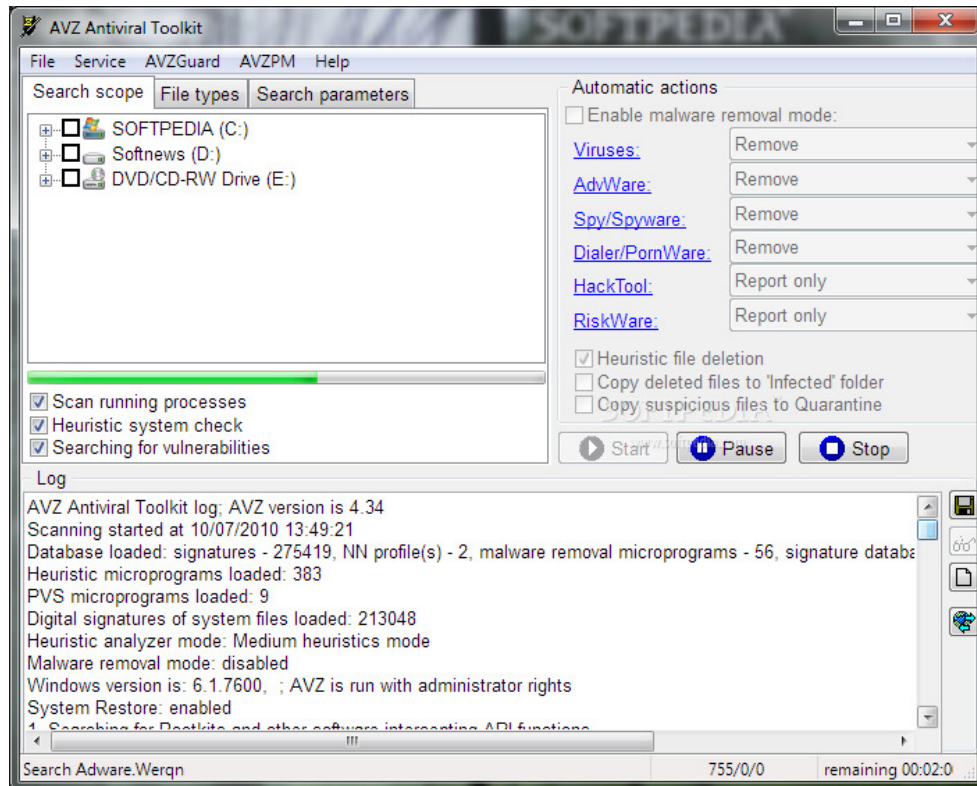
Avast! Antivirus, shown in Figure 9, is an antivirus tool that is developed by the Avast! Corporation. Approximately 2 years ago, the company began incorporating some of the GMER detection algorithms into the application. This software provides many of the “on demand” file scanning and real-time protection features that other anti-virus programs such as Symantec and McAfee offer, and additionally provides heuristics-based scanning to identify newly-developed malicious programs. The installation and use are both fairly straightforward, and there are many options to customize the level of system scanning and heuristics. This application can be used concurrently with other applications and there was not a noticeable performance penalty.



**Figure 9. Avast! Antivirus**

### 6.3 AVZ Antivirus

AVZ Antivirus, shown in Figure 10, is a free anti-malware application developed by Oleg Zaytsev. The application utilizes several different detection techniques, including signature-based, heuristics-based, as well as hooking detection. AVZ does provide real-time malware protection, however it does not appear to include any self-protection mechanisms. AVZ has not been updated since approximately 2008, so it is possible that many of the detection methods may not be relevant any longer.

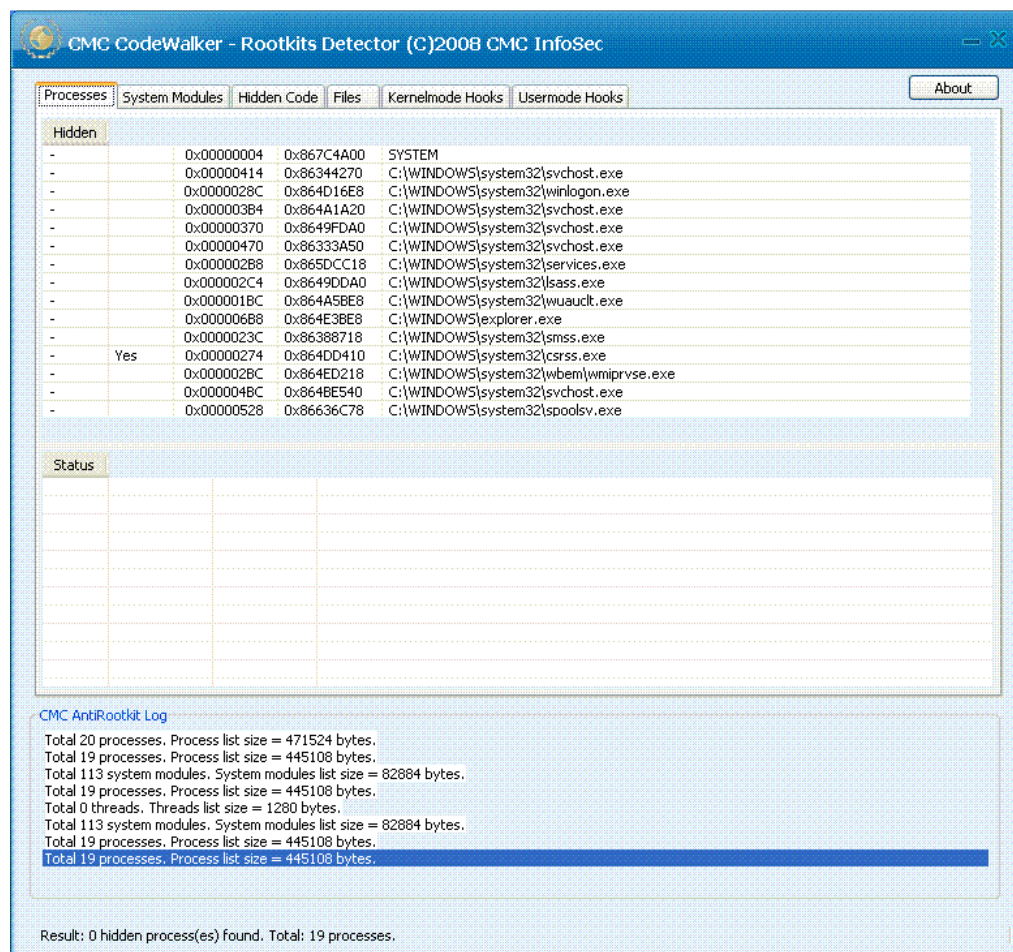


**Figure 10. AVZ Antivirus**

### 6.4 CMC Codewalker

Codewalker, shown in Figure 11, is a system diagnostic tool developed by CMC Infosec in Vietnam. The user interface is very similar to other tools

such as GMER and offers many of the same services, such as detection of hidden processes and files, as well as user and kernel mode hooks. The application is a standalone .exe and is very simple to install and use, however, there are few to no options available to configure the level of scanning and behavior. Additionally, there do not appear to be any removal features associated with the program, so if a rootkit is detected, another application must be used. It appears as though the tool has not been updated since 2008, which likely will adversely affect its ability to detect new forms of malware.



**Figure 11. CMC Codewalker**

## **6.5 ComboFix**

Combofix is a malware detection and removal application developed by an anonymous security professional known as "sUBs". The inner workings of this application are highly secretive, and there is very little information available other than some basic user guides and tutorials. It appears that Combofix is primarily a signature-based malware remover that is constantly updated, and does not implement more sophisticated heuristic-based algorithms. It does integrate rootkit-detection functionality using a GMER-based module, which can identify hidden OS objects. There do not appear to be any self-protection mechanisms built-in to Combofix.

## **6.6 ESET SysInspector**

SysInspector is a diagnostic tool developed by the ESET corporation. It provides insight into running processes, network connections, registry entries, drivers, etc. The user interface is very cumbersome, because there is no "Scan" button that is commonly available with most security applications. Each of the major categories (files, drivers, services, etc.) are assigned a color based on the most "risky" entry. For example, in Figure 12, the Critical Files section is green because there are no risky files detected, but the Drivers section is red because the application detected the presence of the Black Energy driver. There is a slider bar located near the top of the window that can be used to filter the entries based on risk level. Overall SysInspector appears to be a useful diagnostic tool, but there appears to be no hooking detection or removal features, so it is not recommended for use in detecting rootkits.

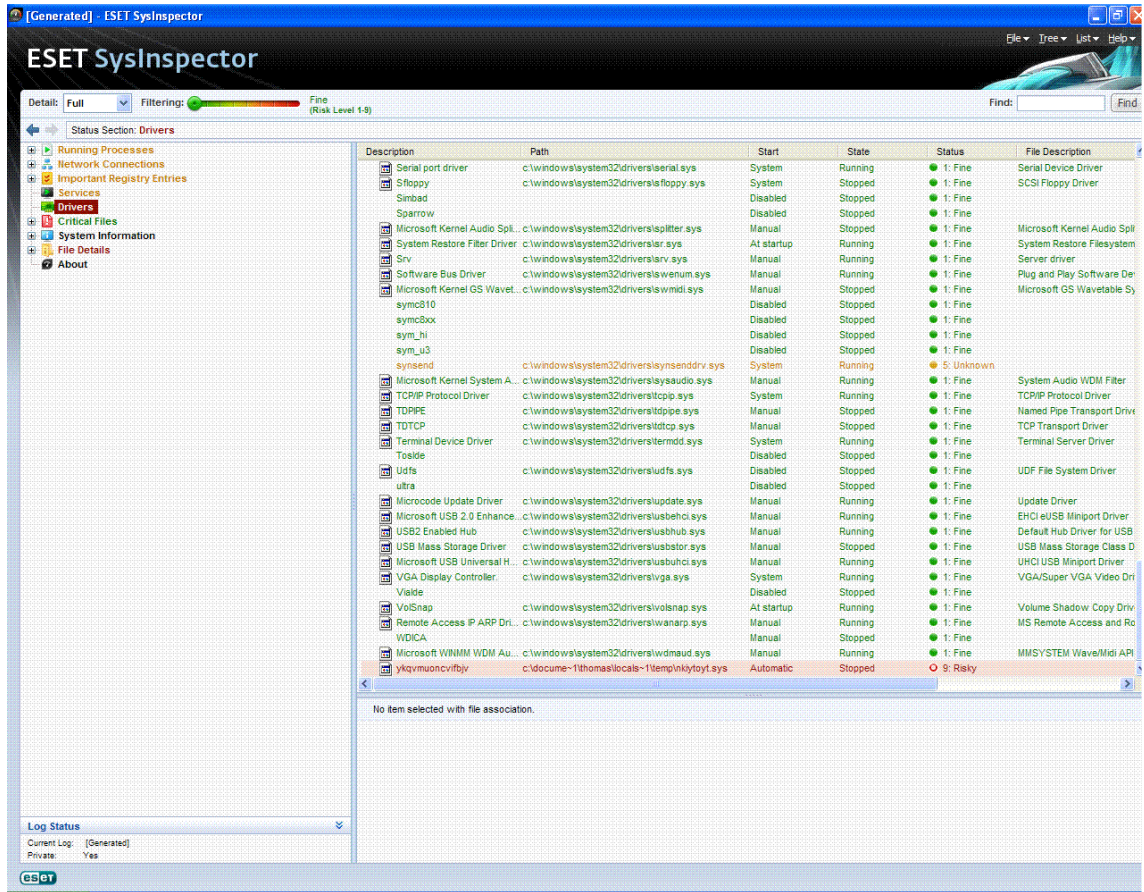


Figure 12. ESET SysInspector

## 6.7 F-Secure Internet Security

F-Secure Internet Security, shown in Figure 13, is a broad security application that provides detection of viruses, spyware and rootkits using on demand system scanning and real-time system protection, and also provides other features such as parental content filtering. The application is highly configurable, and includes options to control settings for the level of heuristic scanning. Internet Security 2011 has a free 30-day trial and there is a tiered pricing model based on the length of the subscription for updates.

The rootkit detection algorithms are based on a previous F-Secure product known as Blacklight, which was one of the more popular rootkit detectors in the 2006 timeframe. F-Secure Internet Security uses a combination of detection techniques, including signature-based and cross-view.

Additionally, F-Secure hooks several services in the Windows OS, in order to provide better real-time detection as well as self protection of the F-Secure application itself. For example, it hooks services such as NtCreateProcess, NtCreateThread, NtRenameKey, as well as several others. By hooking these services, F-Secure can monitor the creation or termination of processes, threads, even network ports, and intervene if it is determined that an application is performing a malicious activity.



**Figure 13. F-Secure Internet Security**

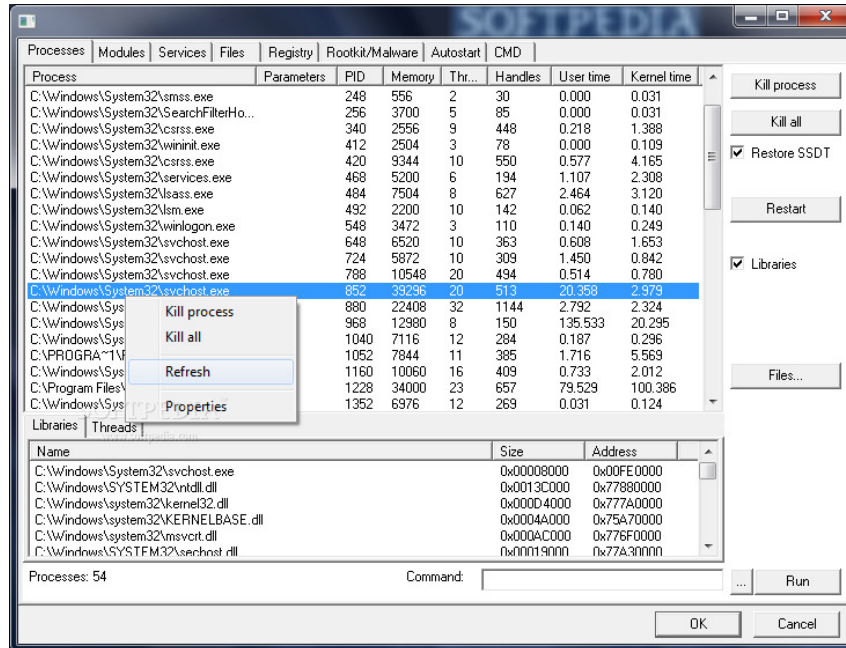
## **6.8 GMER**

GMER is a free rootkit detector developed by Przrmyslaw Gmerek, a Polish security researcher. This application has been consistently one of top



performing rootkit detectors since its initial release in the 2006-timeframe. It is still being actively developed and updated as of this writing. Some of the features that GMER provides include detection of hidden processes, threads, services, files, as well as detection of several different types of code hooks. It also provides removal and restoration options if a rootkit is detected. Additionally, GMER has built-in protection by hooking various Windows OS services to prevent malware from interfering with its operation. Additionally, GMER randomly changes the name of its running process as another method of self-protection.

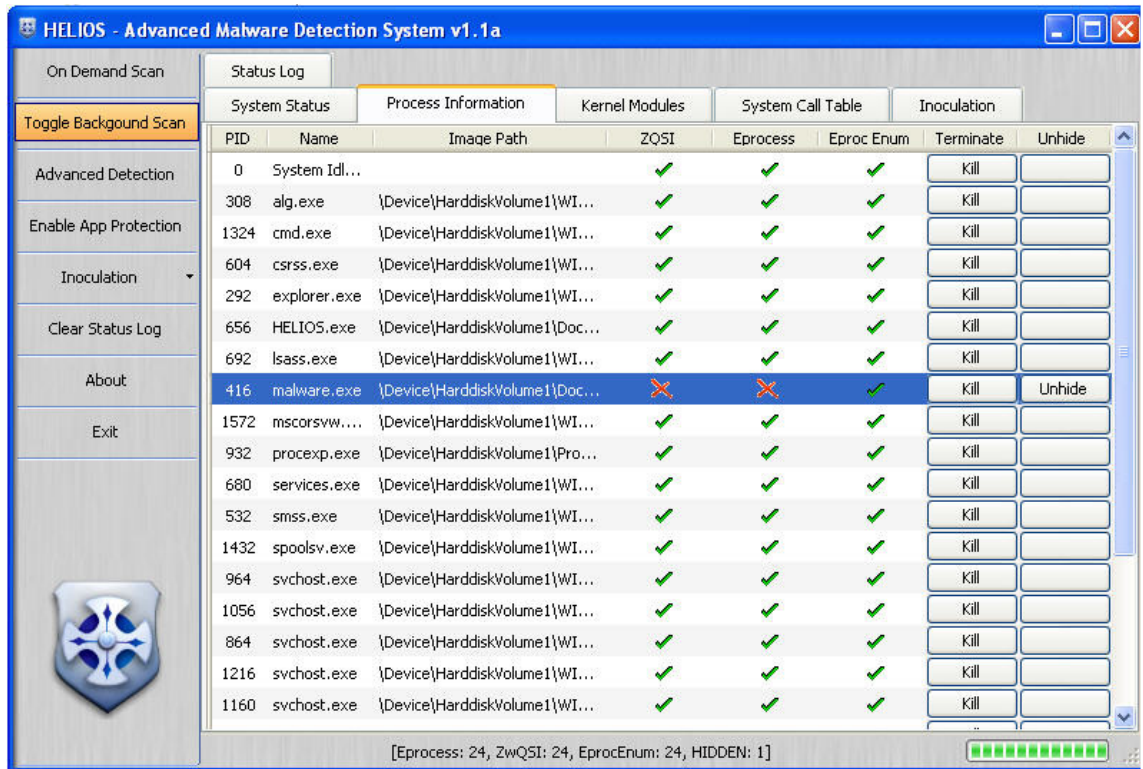
The installation and operation of GMER is very straightforward. Upon loading the application, there are several different tabs that the user can select to perform different types of scans and view the associated output, as shown in Figure 14. There is also an option to output all the scan results into a single report that can be saved and viewed as a text file. One significant downfall of using GMER is that it is extremely intrusive on the operation of the system, and it is important to not perform any other tasks while GMER is performing a scan. It is likely that the operating system will crash due to the low-level nature of the scans if other tasks are attempted.



**Figure 14. GMER rootkit detector**

## 6.9 Helios Lite

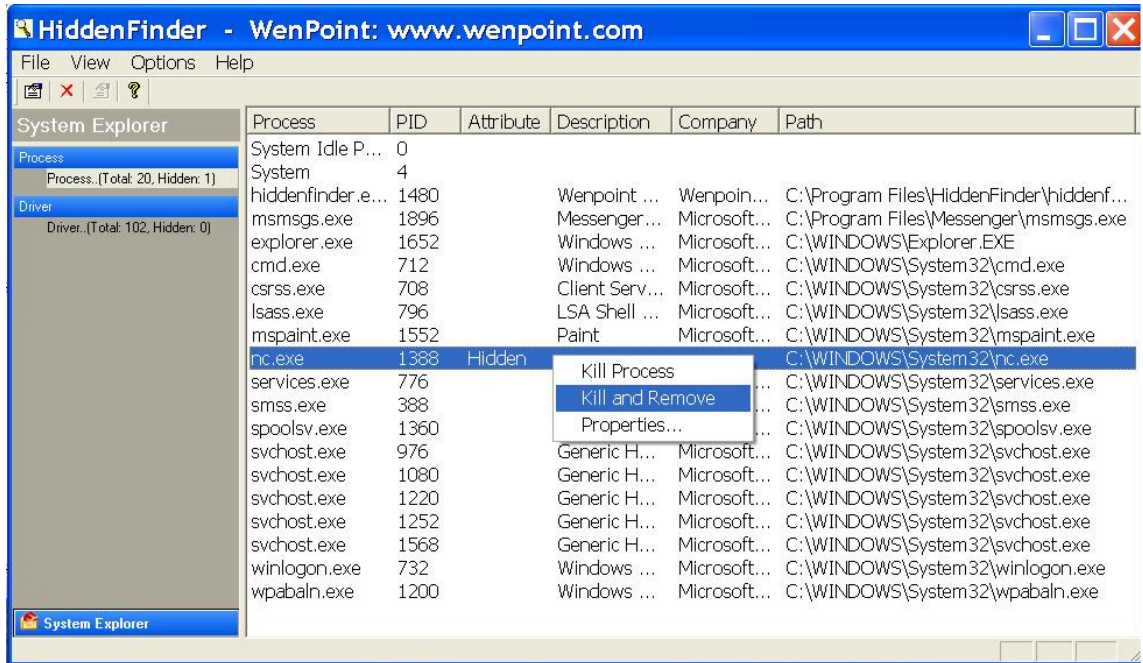
Helios Lite, shown in Figure 15, is a free rootkit detector developed by Miel Labs. It includes a subset of the features available in the Helios malware detection system, and was designed to be a portable application that can be executed from a USB drive. The primary detection algorithms utilize cross-view techniques, but also performs some limited hooking detection as well as heuristic-based detection. The user interface is fairly intuitive, with several options clearly provided on the left-side of the application.



**Figure 15. Helios**

## 6.10 Hidden Finder

Hidden Finder, shown in Figure 16, is a diagnostic tool developed by the WenPoint Corporation. It provides on demand scanning for hidden processes and drivers using the cross-view technique. It does not appear to be in active development any longer, and has not been updated since approximately 2008.



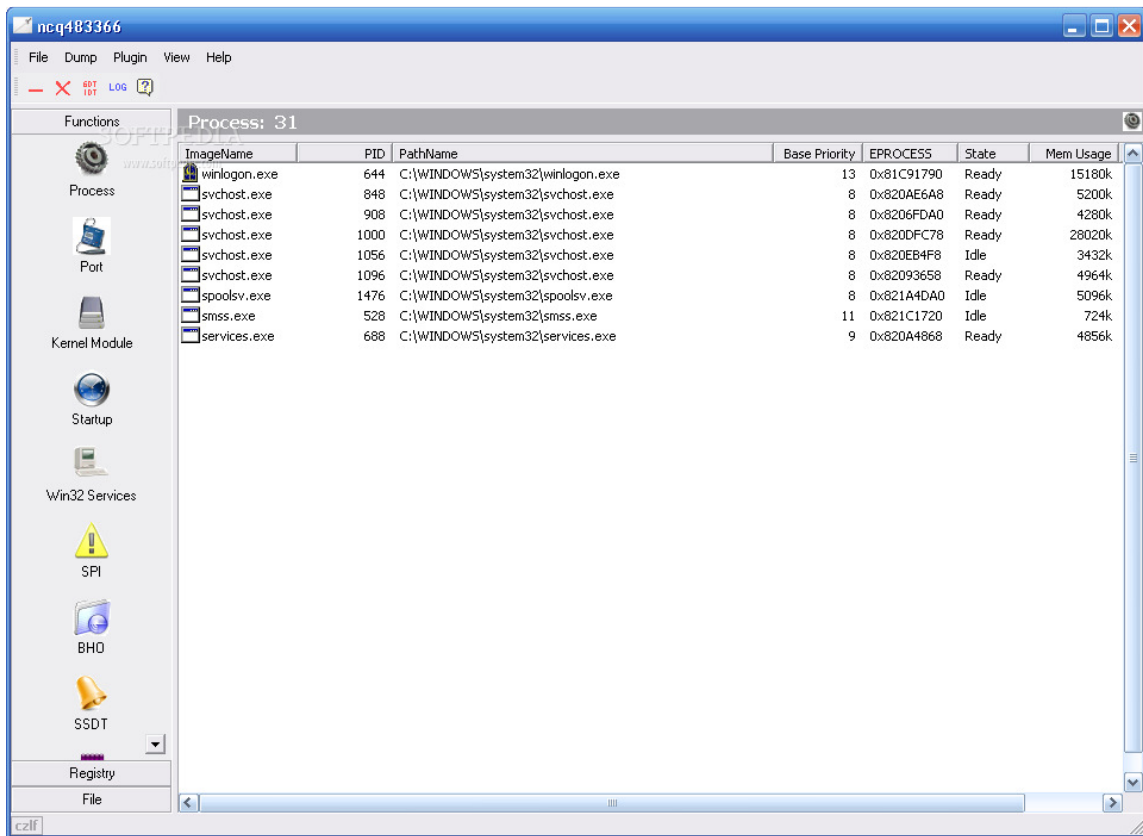
**Figure 16. HiddenFinder**

### 6.11 Ice Sword

Ice Sword, Figure 17, is a diagnostic tool that was developed by an anonymous Chinese security researcher. This was one of the top-performing rootkit detectors in the 2006-2007 timeframe, but has not been updated since 2008. One significant drawback is that Ice Sword only works under the Windows XP environment, and will not even start in Vista or Windows 7. Ice Sword provides utilities to detect hidden processes and files, as well as several different types of code hooks. The user interface is almost identical to Antiy Atool, given the geographic proximity of the developers, it is possible that some collaboration occurred.

Ice Sword was also one of the first applications to include self-protection mechanisms by hooking various Windows services, which prevent malware from interfering or terminating its operation. Some of the services Ice Sword

hooks include NtCreateProcessEx, NtTerminateThread, etc to monitor for any applications that attempt to create or terminate processes and threads, and intervene if needed. The hooking that Ice Sword performs is not nearly as thorough as F-Secure, for example no network services are hooked at all.



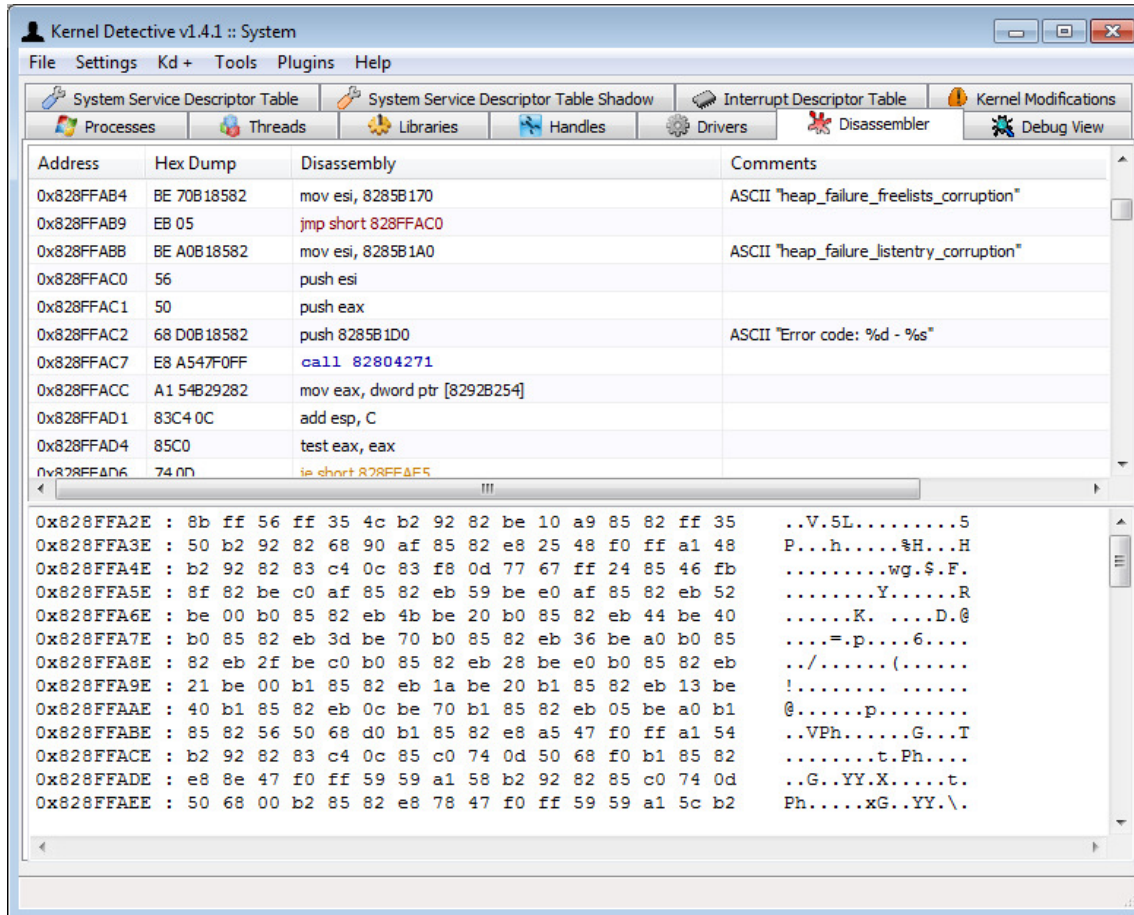
**Figure 17. IceSword**

## 6.12 Kernel Detective

Kernel Detective is a diagnostic and malware removal tool developed by the Arab Team 4 Reverse Engineering (AT4RE), a private team of security researchers. This application is still in active development and utilizes a number of different techniques to identify malware, including cross-view and hooking detection. Additionally, Kernel Detective hooks several Windows

services to provide self-protection from malware interfering or terminating its operation.

The application is very straightforward to install and use. There are a number of tabs to select different diagnostic views of the operating system, as shown in Figure 18. The tabs include running processes and threads, SSDT and IDT hooking, kernel modifications, and even includes a disassemble to inspect selection regions of kernel memory. However, one key interface component that appears to be missing is a comprehensive “one button” scan and reporting capability, which is available in tools such as GMER and Rootkit Unhooker. If a file is determined to be infected or if a hook is detected, the user can attempt to restore the affected component to the original state, or remove it entirely.



**Figure 18. Kernel Detective**

### 6.13 K X-ray

K X-ray is a diagnostic utility developed by an anonymous security researcher. It does not appear to be in active development any longer, has not been updated since approximately 2008, and only works in Windows XP. K X-ray acts as a standalone application and is easy to install, but the user interface leaves much to be desired. There are a number of different system views that can be selected on the left side of the application window, and the results are displayed on the right side of the window. The main issue with the user interface, shown in Figure 19, is that window resizing is not

possible, so if file paths or names exceed the width of the window, they are not viewable.

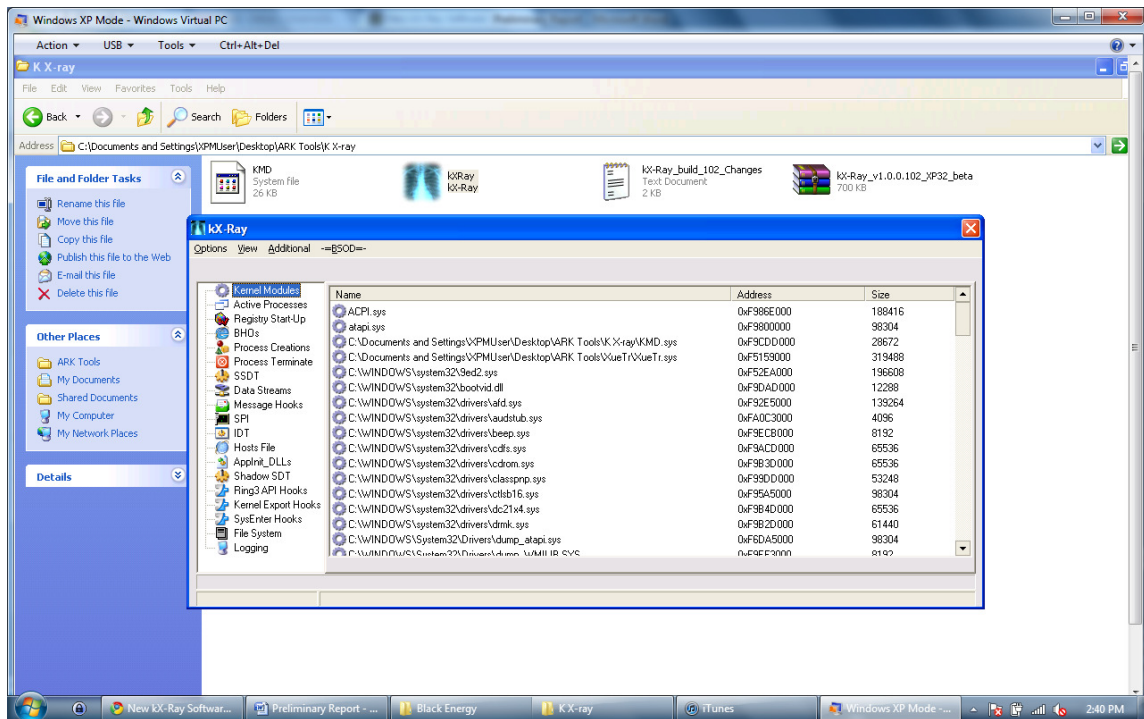


Figure 19. K X-ray

## 6.14 Kaspersky Internet Security

Kaspersky Internet Security (KIS), shown in Figure 20, is a broad anti-malware application developed by Kaspersky Labs in Russia. It is very similar to other Internet Security applications such as F-Secure and Panda, in that it offers on demand system scanning as well as real-time system protection. Its primary method of detection uses the cross-view technique, but it also includes heuristics-based scanning to detect malicious applications. KIS has a 30 day free trial period, but ultimately it is a commercial application.



KIS hooks a large number (over 50) Windows OS services in an attempt to provide better real-time protection as well as self-protection for the KIS application itself. By performing these hooks KIS can monitor the creation of process, threads, and network ports, as well as other applications attempting to install hooks themselves.



**Figure 20. Kaspersky Internet Security**

### **6.15 Malwarebytes Anti-Malware**

Malwarebytes Anti-Malware (MBAM), shown in Figure 21, is a free anti-malware application developed by the Malwarebytes corporation, a private security firm located in the United States. Like some of the other anti-malware tools, MBAM provides on demand scanning as well as real-time

protection. The primary methods of detection uses the signature-based and cross-view techniques, but MBAM also provides heuristics-based scanning using a proprietary algorithm. The installation is very straightforward, a Windows-based installer walks the user through the process and the directions are intuitive. The user interface is also fairly intuitive, with a tabbed-interface clearly providing commands to update the signature definitions and perform different levels of system scanning.

One area that differentiates MBAM from applications such as F-Secure and Kaspersky Internet Security is the lack of Windows service hooking. This means that MBAM does not have any self-protection mechanism built-in, and nor any real-time protection. A commercial version of MBAM can be purchased that includes these features.

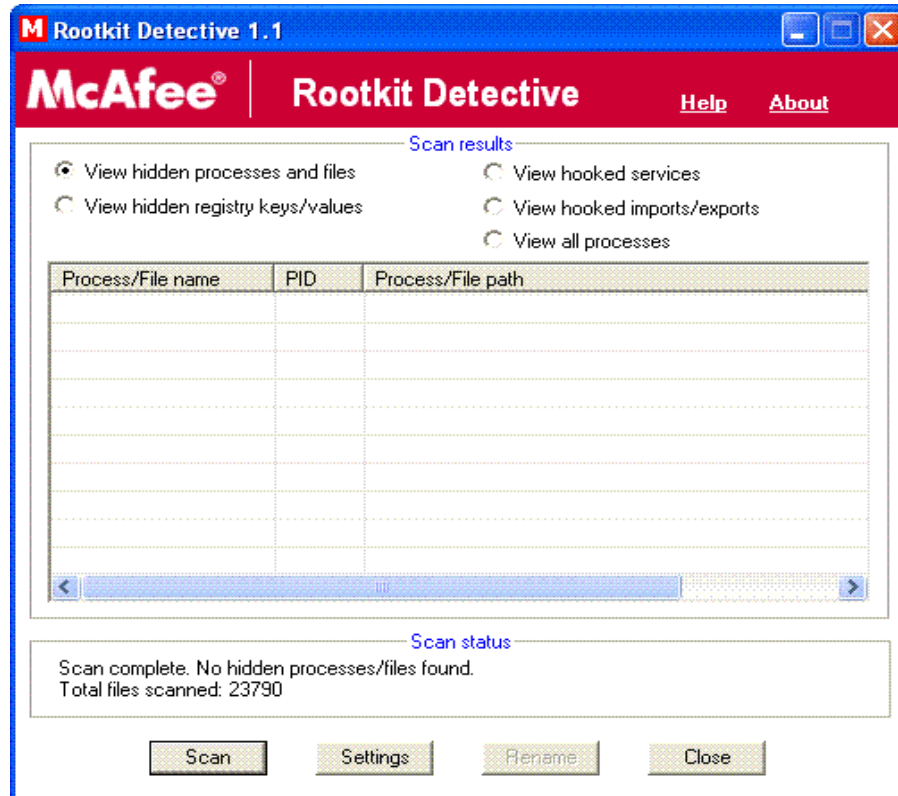


**Figure 21. Malwarebytes Anti-Malware**

## **6.16 McAfee Rootkit Detective**

Rootkit Detective is a standalone rootkit scanning application developed by the McAfee Corporation. It appears that the application has not been updated since approximately 2008, and only operates under the Windows XP environment. The primary method of detection utilizes the cross-view technique, but it appears that a limited hooking detection capability is also provided. The installation is very straightforward; the application is a standalone .exe file and can be used via USB drive. The user interface is also fairly intuitive, there are a limited number of options to select which type of

scan to perform and the results are clearly indicated in a textbox, as shown in Figure 22.

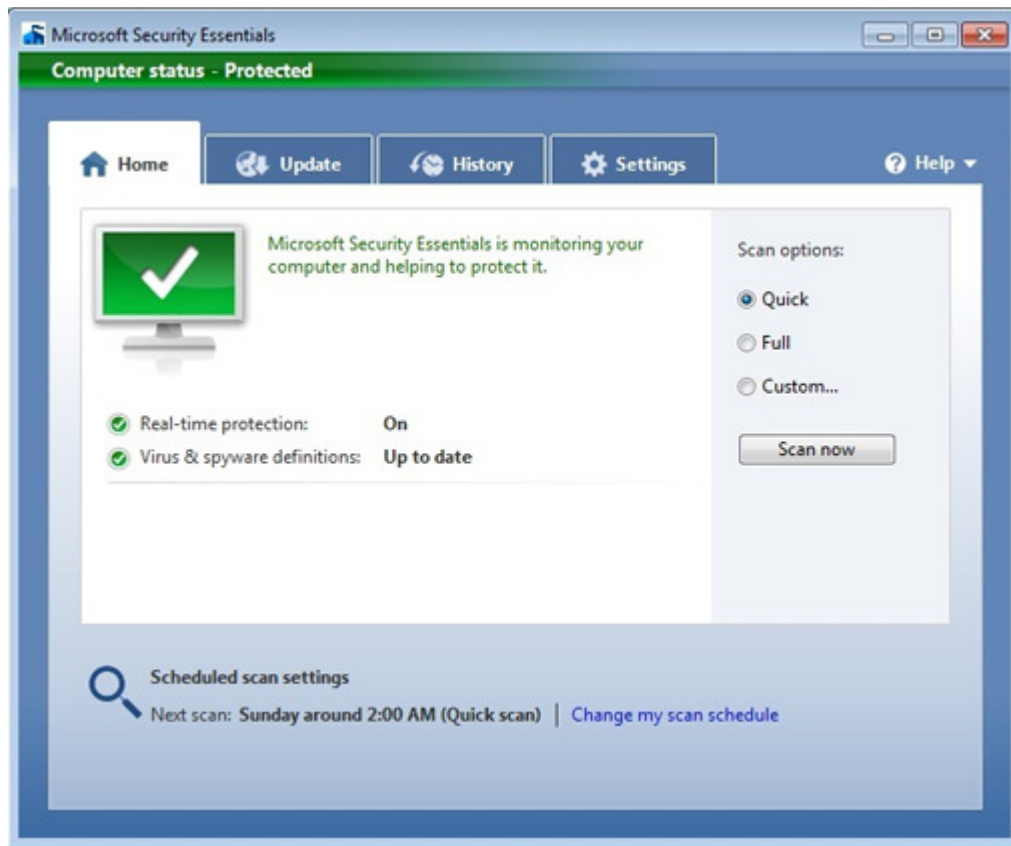


**Figure 22. McAfee Rootkit Detective**

### **6.17 Microsoft Security Essentials**

Security Essentials, shown in Figure 23, is a free application developed by Microsoft. It is similar in functionality to other malware detection applications such as MBAM and F-Secure, however it is freely available at no charge. The tool incorporates several different detection techniques, including signature-based, cross-view, as well as heuristics-based detection. Security Essentials hooks a number of Windows services in an attempt to provide better real-time protection, however it does not appear to have any

self-protection mechanisms built in. One significant advantage to using Security Essentials is that it integrates well with the Windows OS, due to collaboration between the associated departments at Microsoft. This ensures that OS changes that could adversely affect the operation of the detector should be minimized.



**Figure 23. Microsoft Security Essentials**

### **6.18 Panda Internet Security**

Panda Internet Security, shown in Figure 24 is a broad malware detection application developed by Panda Security. It is similar to applications such as Kaspersky and F-Secure Internet Security in that it provides on demand system scanning, as well as real-time protection from malware. Installation

and operation are both very intuitive, with options to perform customized configuration and various levels of system scanning being clearly displayed.

The application also hooks various Windows services such as NtTerminateProcess and several others to provide better real-time protection as well as preclude malware from interfering or terminating its operation.



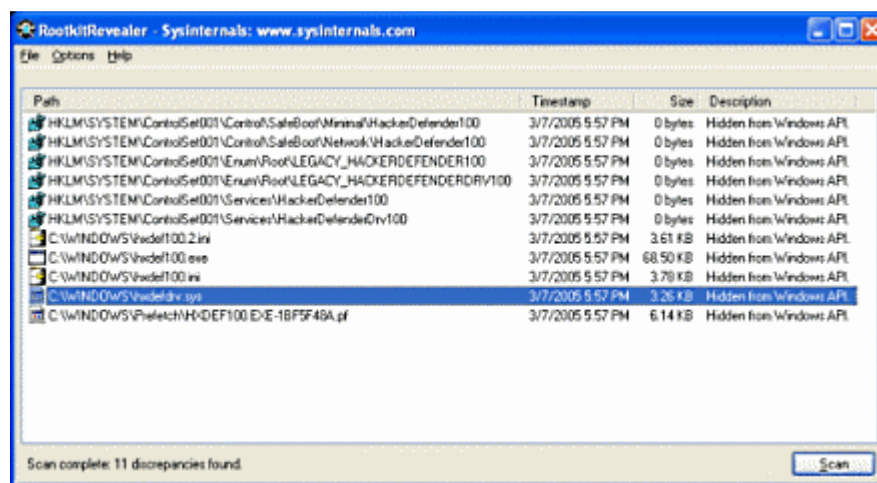
**Figure 24. Panda Internet Security**

### **6.19 Rootkit Revealer**

Rootkit Revealer, shown in Figure 25, was one of the first rootkit detection applications, and was developed by Mark Russinovich from Microsoft. It was the first application to use the cross-view technique to reveal the presence of hidden files and registry keys within the Windows operating system. Rootkit Revealer does not perform any sort of detection of code hooks or kernel

memory modifications, it only detects hidden files and registry keys. One other drawback is that Rootkit Revealer does not offer any removal services.

While Rootkit Revealer was a high-performing application in the 2006-2007 timeframe, and was recommended by a large number of security experts, it has not been updated in almost 4 years. As a result, it is unable to detect most of the modern rootkits available today.

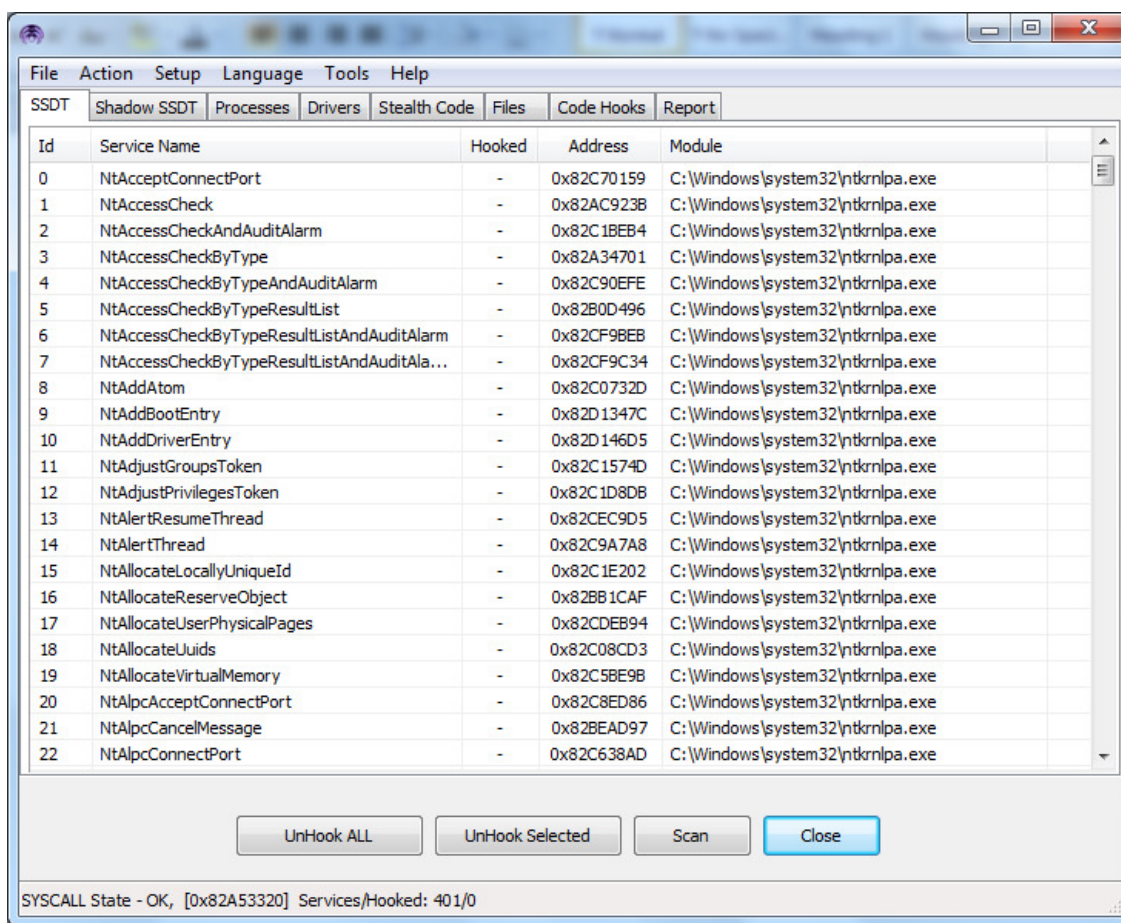


**Figure 25. Rootkit Revealer**

## 6.20 Rootkit Unhooker

Rootkit Unhooker is a free, on-demand rootkit scanner and system diagnostic utility developed by an anonymous Russian security researcher known as DiabloNova. This tool utilizes several detection techniques, including cross-view, hooking, and kernel modification detection. The installation is very simple, since the application runs as a standalone .exe file. The user interface is also very intuitive, as shown in Figure 26. A tabbed interface is used to clearly show the different types of diagnostic scans available. A "Report" tab provides a comprehensive system scan that

integrates all of the results in one output file. The output from the various types of scans clearly explains the system discrepancy, and makes it very easy for the user to determine if action needs to be taken. In addition to the scanning and system diagnostics, Rootkit Unhooker provides the ability to restore system hooks as well as limited file removal capability. Rootkit Unhooker also hooks several Windows services such as NtCreateProcessEx and NtTerminateThread to provide self-protection mechanisms, although real-time protection is not provided.

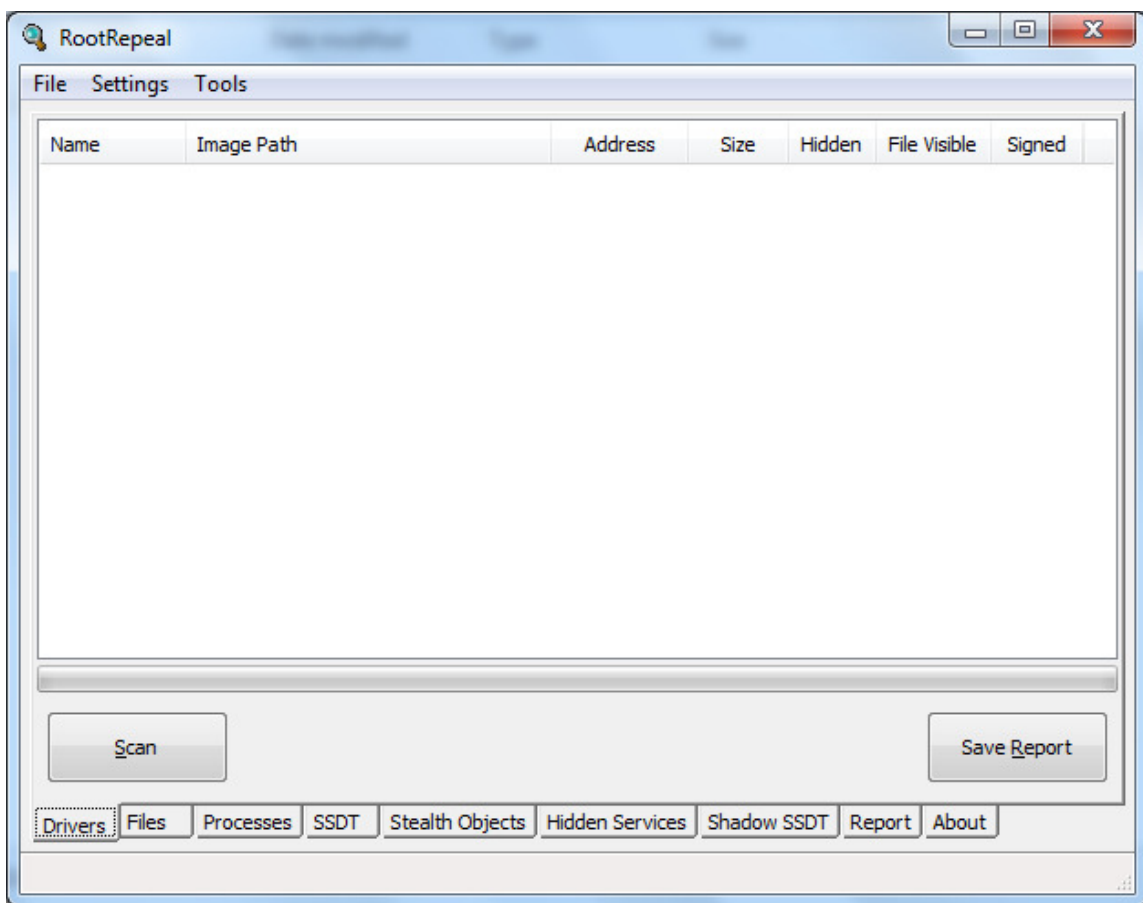


**Figure 26. Rootkit Unhooker**



## 6.21 RootRepeal

RootRepeal is an on demand rootkit scanner and system diagnostic utility developed by a private security researcher in Poland. It is very similar to tools such as Rootkit Unhooker and GMER, and provides essentially the same interface and capabilities. It operates as a standalone .exe file and provides a very intuitive tabbed interface to select between the various types of scans, shown in Figure 27. A comprehensive scanning report is also available.



**Figure 27. RootRepeal**

## 6.22 Sophos Anti-Rootkit

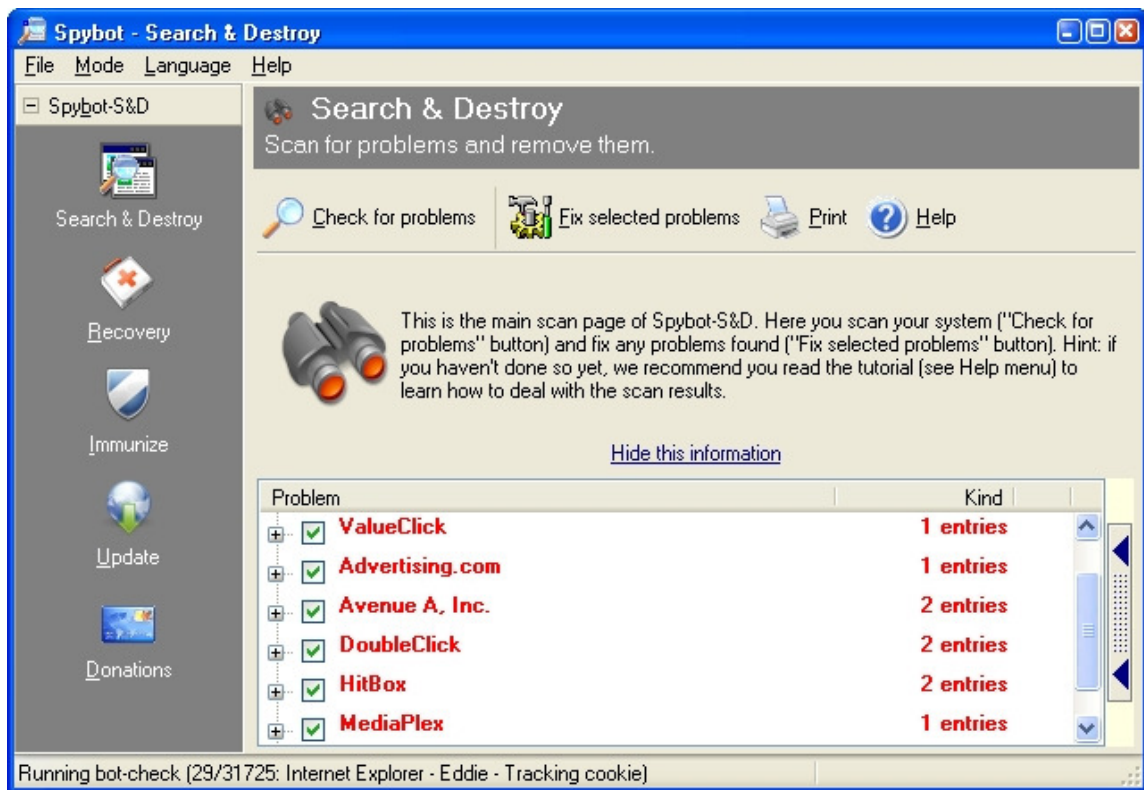
The Sophos Anti-Rootkit application is an on-demand rootkit scanner developed by the Sophos Security Corporation. The primary detection algorithm utilizes the cross-view technique to reveal the presence of hidden processes, files and registry keys. It does not perform any detection of system hooks or kernel memory modifications. The application operates as a standalone file, and the user interface is extremely simple. As can be seen in Figure 28, essentially the only option available to the user is to click a "Start Scan" button, and the results are listed in the window upon completion of the scan. There are virtually no options to configure the operation of the program.



**Figure 28. Sophos Anti-Rootkit**

## 6.23 Spybot Search and Destroy

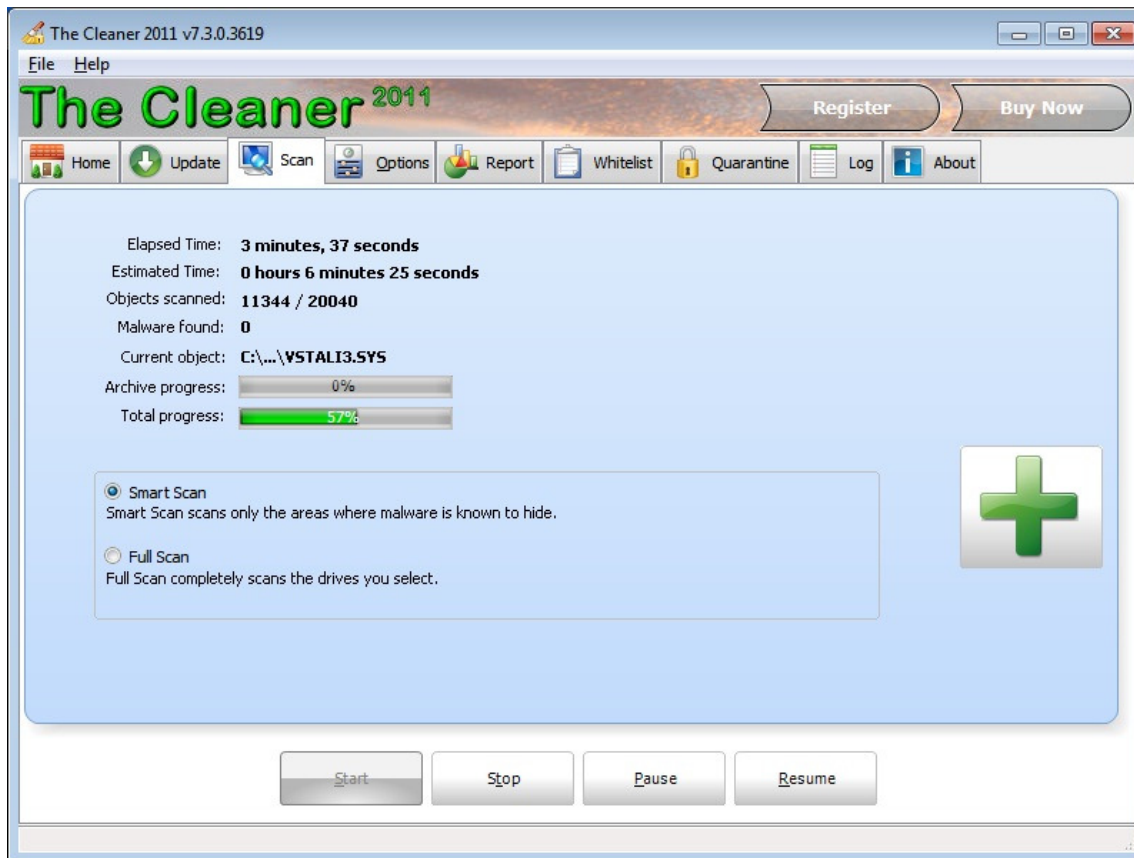
Spybot Search and Destroy, shown in Figure 29, is an on-demand malware scanning application developed by Safer Networking Ltd. The focus of Spybot is to detect and remove spyware and viruses; however, for the purposes of the thesis research, it was included to demonstrate the inability of this application to detect rootkits. Spybot uses a signature-based detection algorithm to scan both the hard disk and memory for malware, and does not perform any other type of detection technique such as the cross-view method.



**Figure 29. Spybot Search & Destroy**

## **6.24 The Cleaner**

The Cleaner 2011, shown in Figure 30, is a malware detection and removal application created by Moosoft Development. It incorporates several different types of detection techniques, including signature-based, cross-view, as well as heuristics-based detection. It does not appear to perform any hooking detection. The software acts as both an on demand disk and memory scanner, and also offers real-time system protection. The application does not appear to have any self-protection mechanisms to prevent malware from interfering with its operation, unlike applications such as GMER, Kaspersky, and MBAM. The installation and operation for The Cleaner is very intuitive, and there are several different configuration options to select different levels of system scanning.

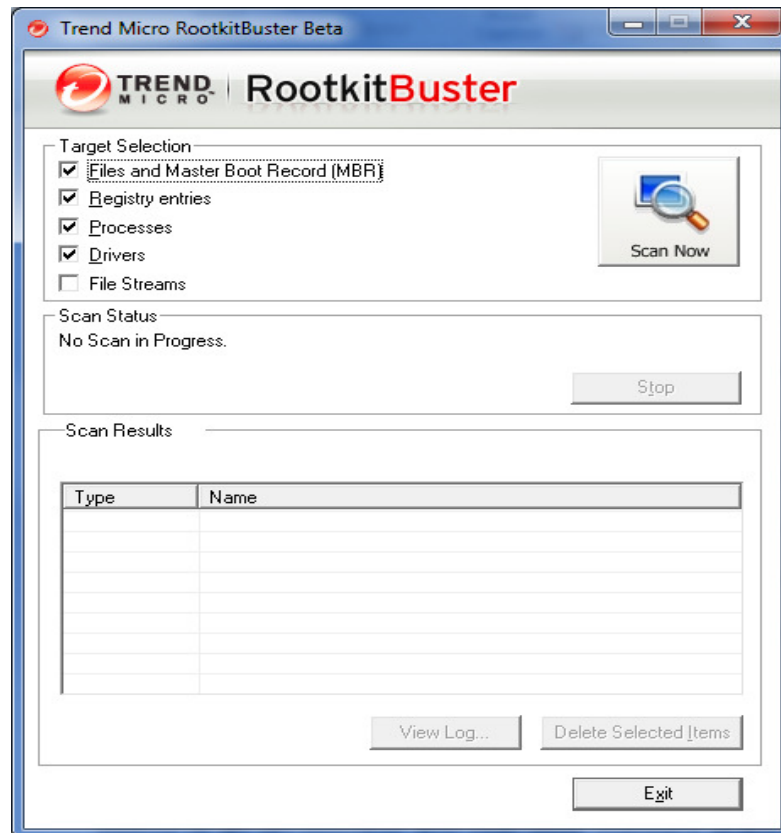


**Figure 30. Moosoft The Cleaner**

## 6.25 Trend Micro Rootkit Buster

Rootkit Buster, shown in Figure 31, is a free on-demand scanner developed by the Trend Micro Corporation. The tool primarily utilizes the cross-view technique to detect hidden files and processes, but also performs hooking detection as well. It appears that Rootkit Buster has not been updated since approximately 2007, so some malware authors have likely figured out how to work around Rootkit Buster's capabilities. The application is a standalone .exe file, and the user interface is very simple. The user simply has to select the various types of system components to inspect, and click the "Scan Now"

button. The results are clearly displayed in a textbox, and can be subsequently selected for removal.

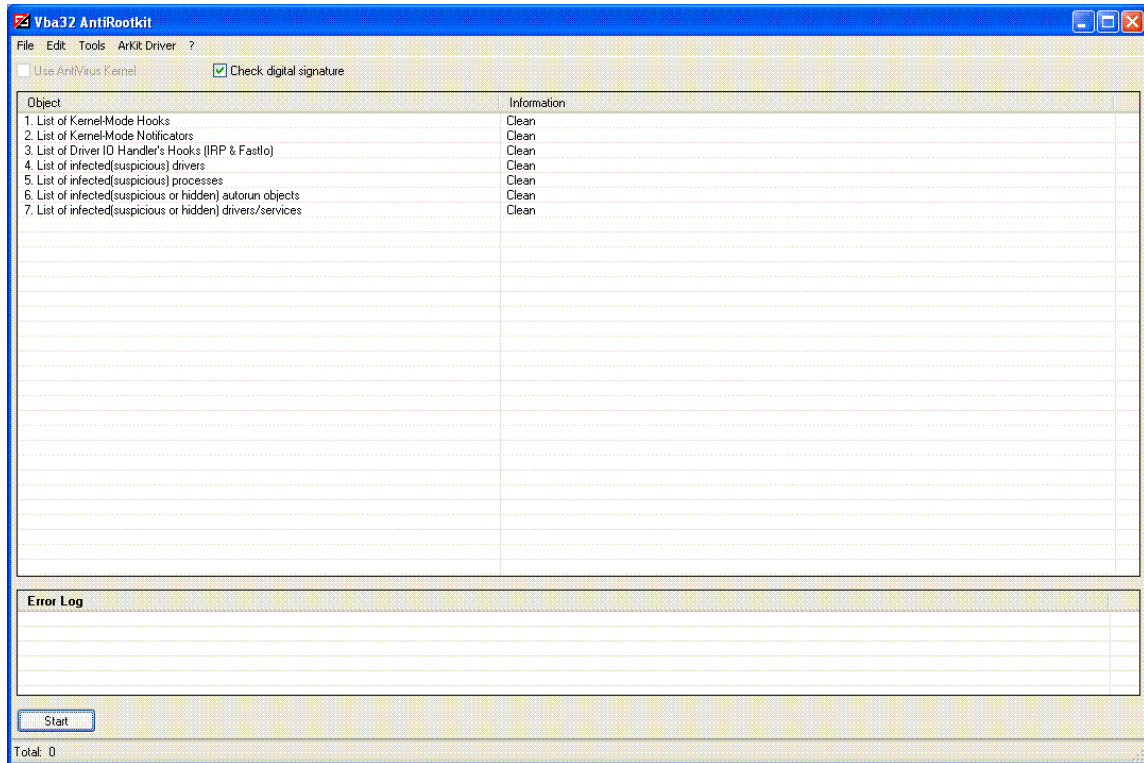


**Figure 31. Trend Micro Rootkit Buster**

## **6.26 VBA32**

VBA32, shown in Figure 32, is an on-demand and real-time malware scanner developed by Virus Blok Ada, a security company located in Belarus. The application incorporates several different types of detection techniques, including hooking detection, cross-view, as well as heuristic-based detection. The application does not appear to provide any self-protection mechanisms to prevent malware from interfering with its operation. The user interface for the VBA32 is not as clear as some of the other applications, but there are

several different configuration options for different levels of system scans, and the user just has to press the “Start” button to begin a scan.



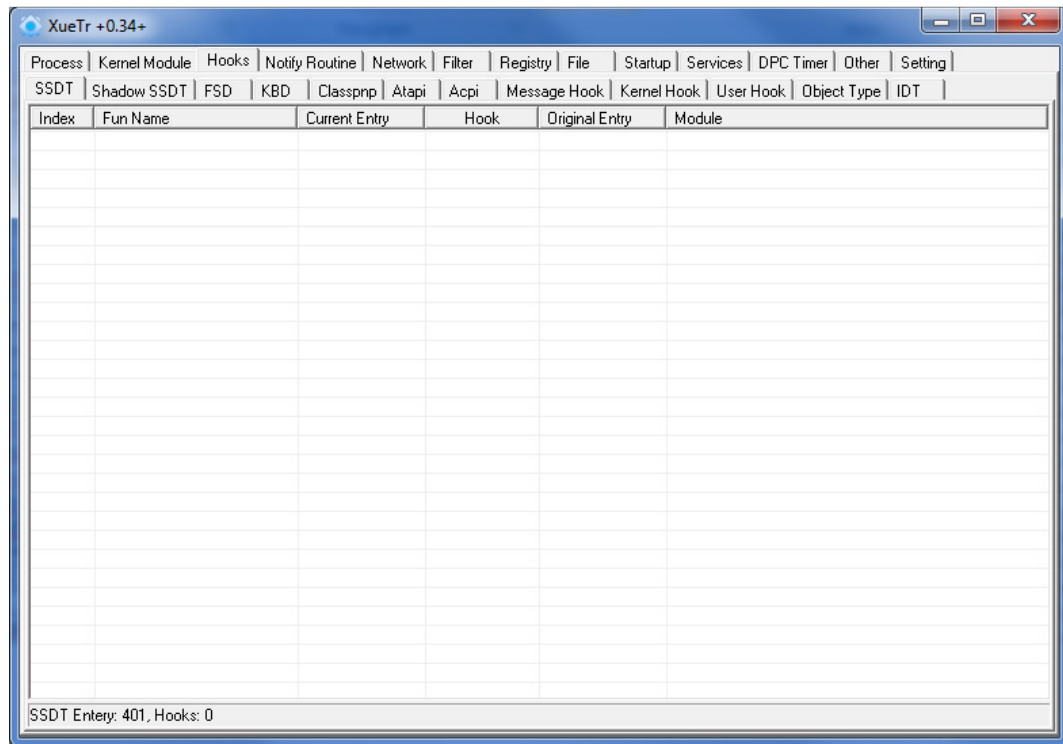
**Figure 32. VBA32**

## 6.27 XueTr

XueTr is a system diagnostic tool and rootkit scanner developed by an anonymous Chinese security researcher. XueTr primarily uses the cross-view and hooking detection techniques to identify malicious software. XueTr acts as a standalone .exe file, and the user interface is fairly intuitive. There are approximately a dozen different tabs (shown in Figure 33) that can be selected to view different areas of the operation system, and if a file needs to be removed or a hook restored, it is straightforward to perform those actions by right-clicking on the affected object. The program also provides self-

protection mechanism by hooking various Windows in the Shadow SSDT, to protect the GUI/window from being closed by a malicious application.

One thing that appears to be missing is a comprehensive scan and reporting capability, which is offered by applications such as GMER and Rootkit Unhooker.



**Figure 33. XueTr Diagnostic Tool**



## **7.0 Experimental Results**

The analysis of this thesis aims to investigate the ability of a large number of anti-rootkit tools to detect and remove a sample of modern rootkits. In this section, the results of the anti-rootkit tool scans will be presented for each of the rootkits. When possible, the characteristics of the anti-rootkit tools will be taken into consideration when analyzing the results of the scans. However, due to the highly proprietary nature of many of the anti-rootkit tools, the details of their detection and removal algorithms cannot be determined. When possible further testing was performed to isolate which technique (signature-based, heuristic-based, etc.) was successful at detecting the rootkit.

In addition to the scan results, the steady-state performance for each of the infected systems will be compared against a clean system. Both the steady-state processor utilization and network performance will be presented and analyzed. A limited amount of system forensic analysis will also be presented for each of the infected systems, including filesystem and registry changes, as well as any modifications to Windows OS internal structures.

Finally, the results of the network-based detection technique will be presented and analyzed.

### **7.1 System Performance and Forensic Analysis**

In this section, the steady-state CPU and network utilization for each of the infected systems will be presented and analyzed, and compared to a clean system. In order to provide an adequate baseline, approximately 60

consecutive hours of data was recorded. The time interval for the CPU measurements was 15 seconds, which is reasonable based on the large amount of observation time [47]. For the network traffic analysis, the dumpcap.exe Wireshark utility was used to capture each packet that was processed by the network interface.

### **7.1.1 Filesystem/Registry Modifications**

By using the methodology as described in Section 4.1, it was possible to observe several filesystem and Windows registry changes caused by each of the rootkits.

The TDL3 rootkit installed a randomly-named file (55wWS.sys) in the C:\Windows\Temp directory, which is a known location for the usermode component [14]. Additionally, the HKLM\system\ControlSet003\Services\Tcpip\Parameters\NameServer registry key was modified to include the IP addresses 93.188.163.73 and 93.188.166.108, which are both located in the Ukraine. It is likely that these domains are responsible for performing the URL redirects that TDL3 is known for.

The Rustock rootkit kernel-mode component file (sstamnsq.sys) in the C:\Windows\System32\Drivers directory was reported by Windiff, however this was expected because the rootkit was installed manually. The only registry keys that appear to have been modified were associated with the HKEY\_LOCAL\_MACHINE\system\ControlSet001\Enum family of keys, which is responsible for ensuring that the driver is loaded at startup.

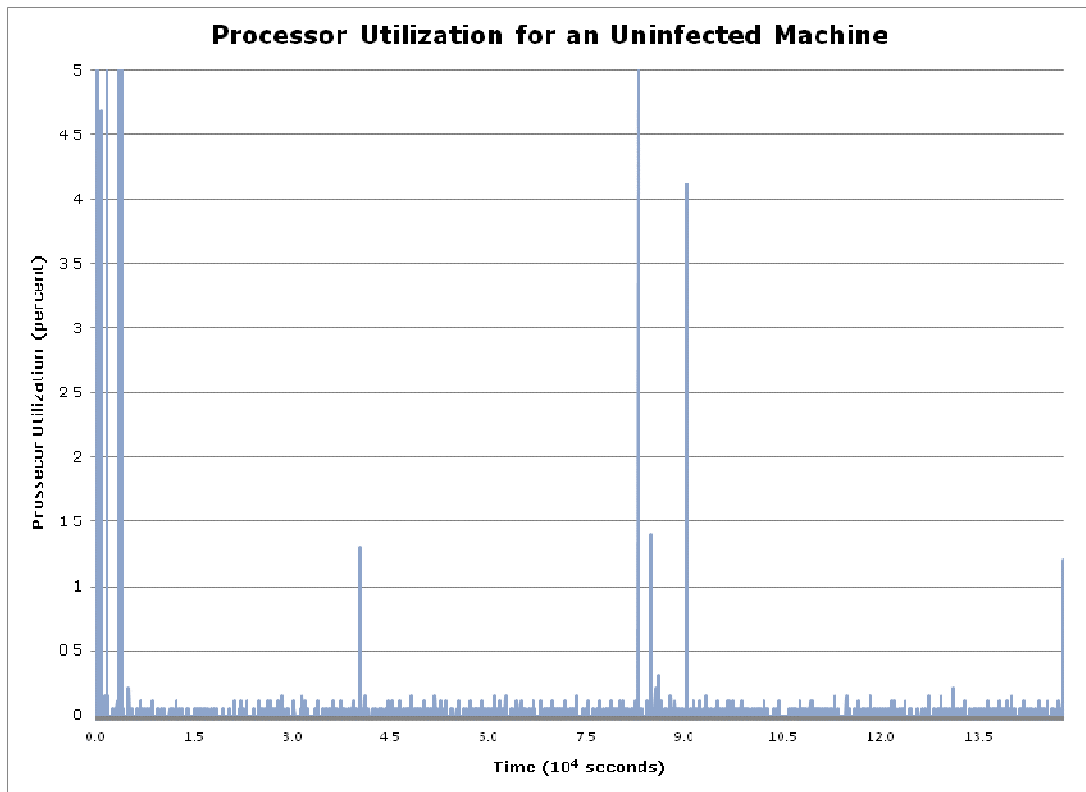
The Black Energy rootkit installed a randomly-named file (szbkqmckhcv.sys) in the Local Settings directory, which is the usermode component. The kernel-mode component (str.sys) was installed in the C:\Windows\System32 directory. There were also a large number of registry keys that were created in order to ensure the usermode driver was loaded as a service upon system startup. For example, the HKLM\system\ControlSet001\Services\domwjfvo registry key was assigned a value of "C:\Docume~1\Thomas\LOCALS~1\Temp\szbkqmckhcv.sys".

The Zbot rootkit created the lowsec directory in C:\Windows\System32, and installed the three files as described in Section 5.4. Additionally, Zbot installed the rootkit driver component (sdra64.exe) in the System32 directory as well. The registry key HKLM\software\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit was updated to include the path to the sdra64.exe file, so that it could be executed upon startup. Note that this is slightly different than the Black Energy and Rustock startup registry modifications, since sdra64.exe is not being loaded as a Windows service, but it effectively achieves the same effect (surviving reboots).

### **7.1.2 Processor Utilization**

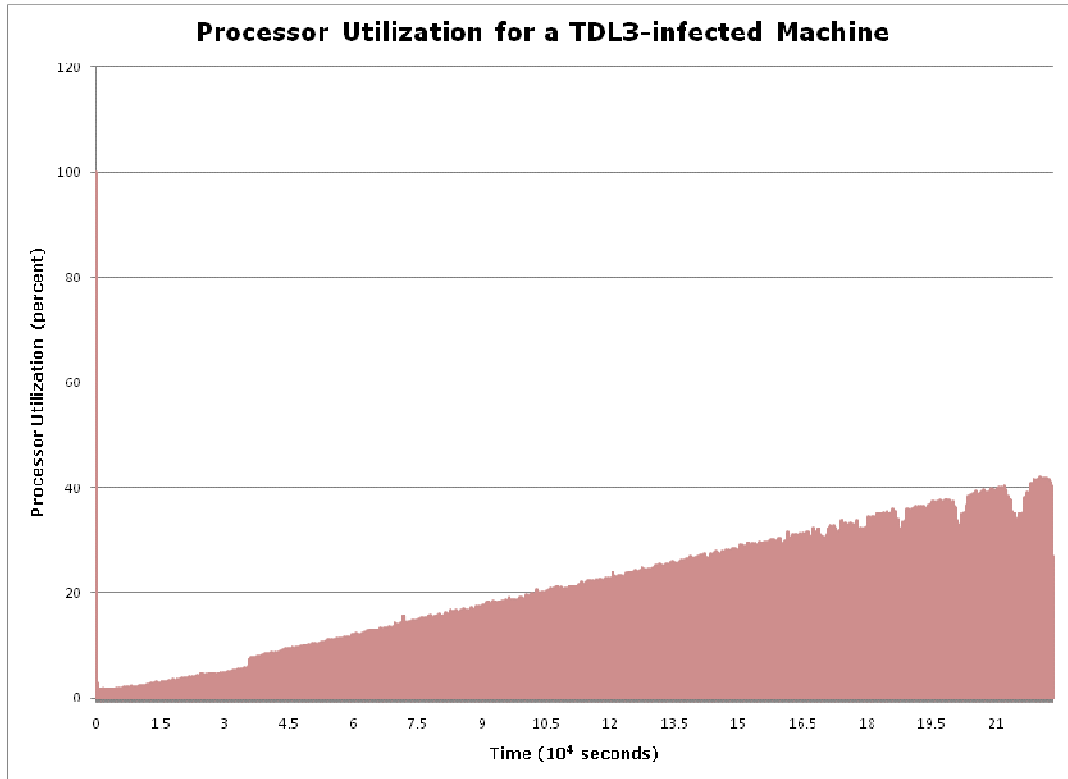
In Figure 34, the steady-state CPU utilization for an uninfected machine is presented. The processor utilization in percent is shown on the y-axis, and time in 10000 seconds ( $10^4$  seconds) is shown on the x-axis. It can be seen that over a period of approximately 41 hours, the steady-state CPU utilization is approximately .1 percent, which is expected since there is no

activity being performed, other than the data logging. This data will be used as a baseline which can be compared against each of the infected systems.



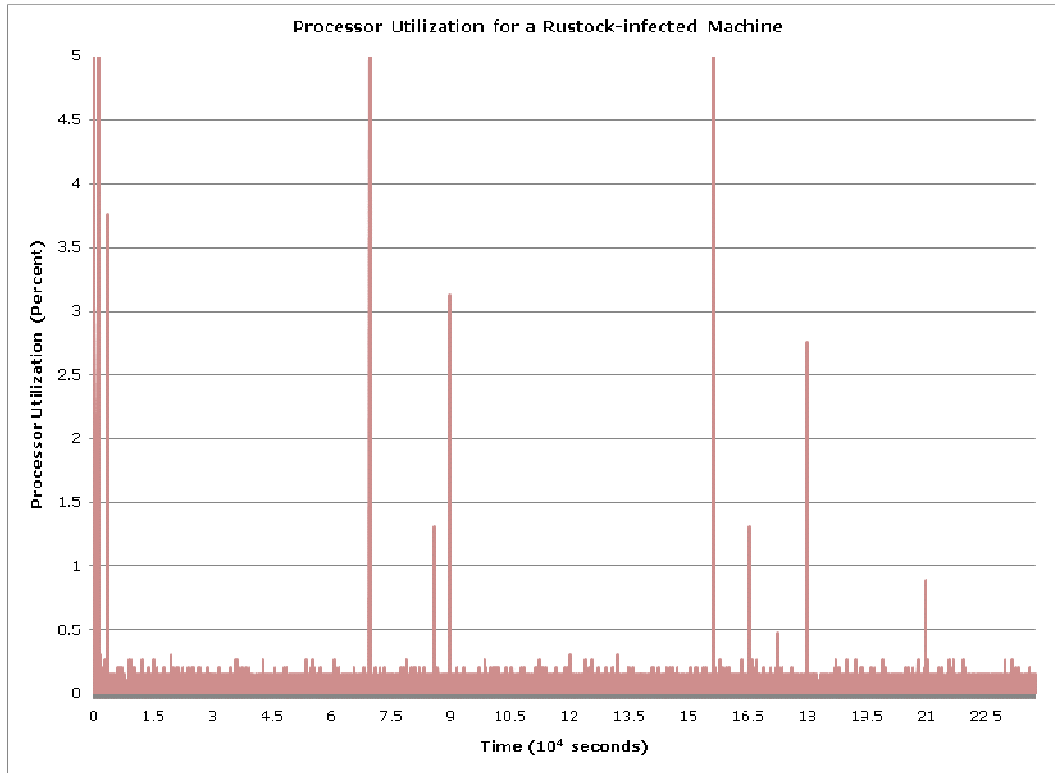
**Figure 34. Processor Utilization for an Uninfected Machine**

Figure 35 displays the CPU utilization for a TDL3-infected system. It can be seen that over a period of approximately 62 hours, the utilization is progressively trending upward in a linear manner. This is clearly a very unstable system and at some point will likely be unable to perform any useful work.



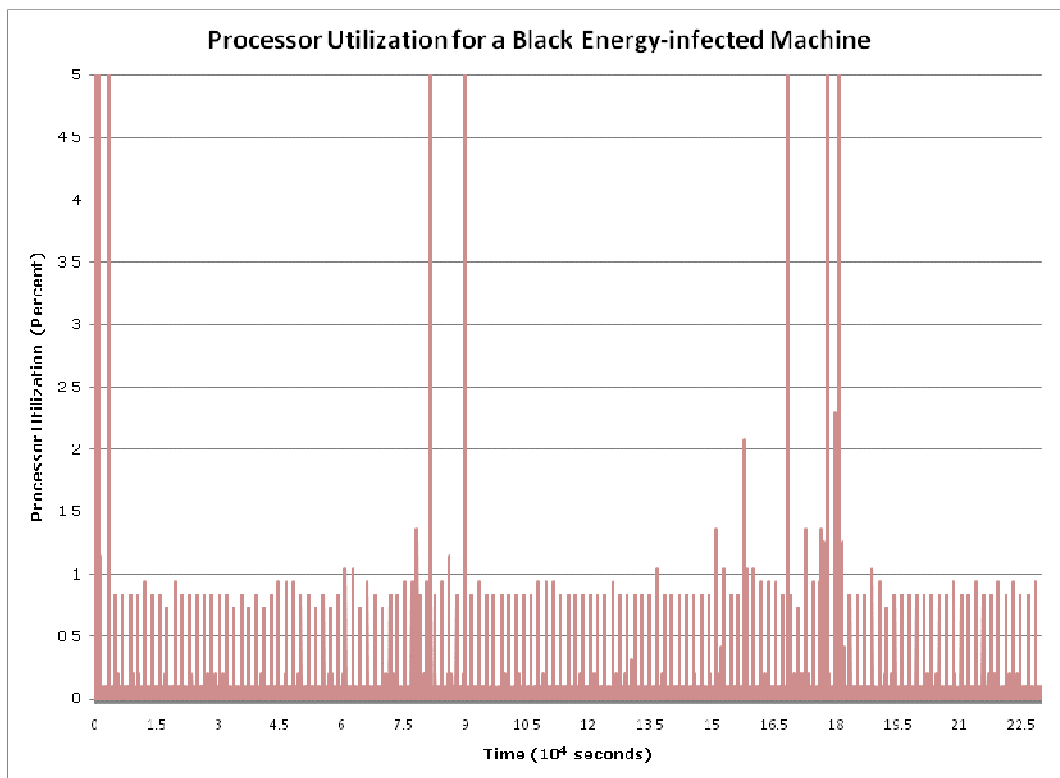
**Figure 35. Processor Utilization for a TDL3-infected Machine**

Figure 36 shows the CPU utilization for a Rustock-infected machine. As can be seen from the data, this system, which was observed for approximately 62 consecutive hours, appears to be much more stable than the TDL-3 machine. However, the average CPU utilization is approximately 0.2 percent, which is double that of the uninfected system. The stability would likely not be an issue, but overtime this system would use up more power resources, which could be very undesirable for large deployments in environments such as data centers.



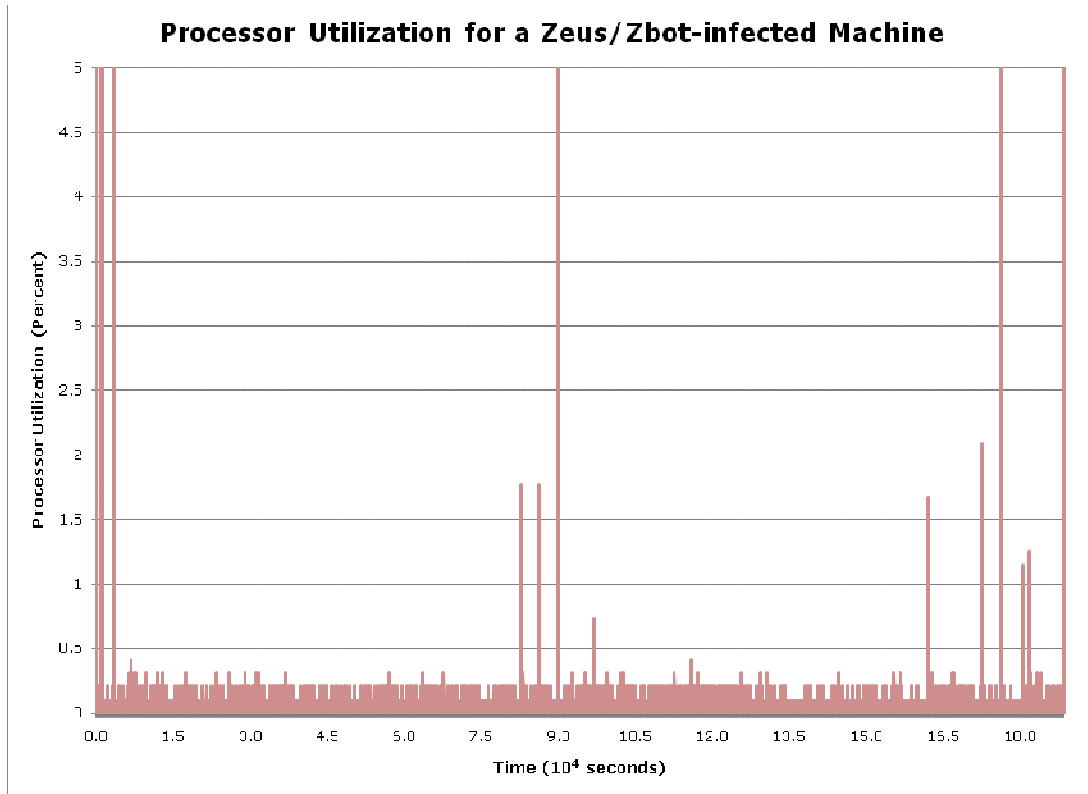
**Figure 36. Processor Utilization for a Rustock-infected Machine**

Figure 37 shows the steady-state CPU utilization for a system infected with the Black Energy rootkit, collected over a period of approximately 62 consecutive hours. As can be seen from the data, the system appears to be much more stable than the system infected with TDL3. However, similar to Rustock, the baseline CPU utilization does appear to be significantly higher than the clean system.



**Figure 37. Processor Utilization for Black Energy-infected Machine**

Figure 38 shows the CPU utilization for a system infected with the Zeus rootkit. As can be seen from the data, the system appears to be much more stable than the one infected with TDL3. However, the baseline CPU utilization is significantly higher than the uninfected system, similar to the Rustock and Black Energy systems.



**Figure 38. Processor Utilization for a Zeus-infected Machine**

### 7.1.3 Network Utilization

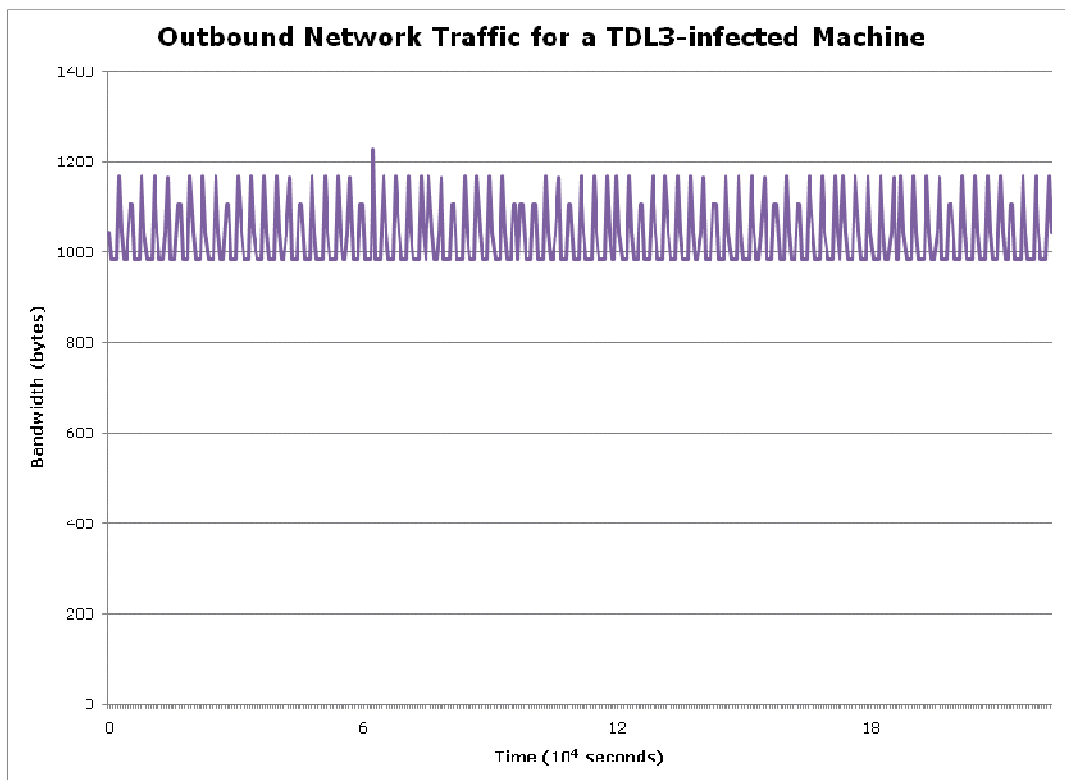
In this section, the steady-state network utilization for each of the rootkits will be presented and analyzed. The network activity was captured using the dumpcap.exe Wireshark utility, which records each packet that is processed by the Network Interface Card (NIC) for the system. Next, the tshark.exe Wireshark utility was used to process the captured packets and provide statistics for each 10 minute time segment during the captured period, using [21] as an example. The tshark.exe utility can calculate statistics for virtually any scenario which the user would like to analyze, by filtering the different types of packets that were captured. For this exercise, the



outbound HTTP traffic was analyzed, because this was the only outbound network traffic observed for each of the rootkits.

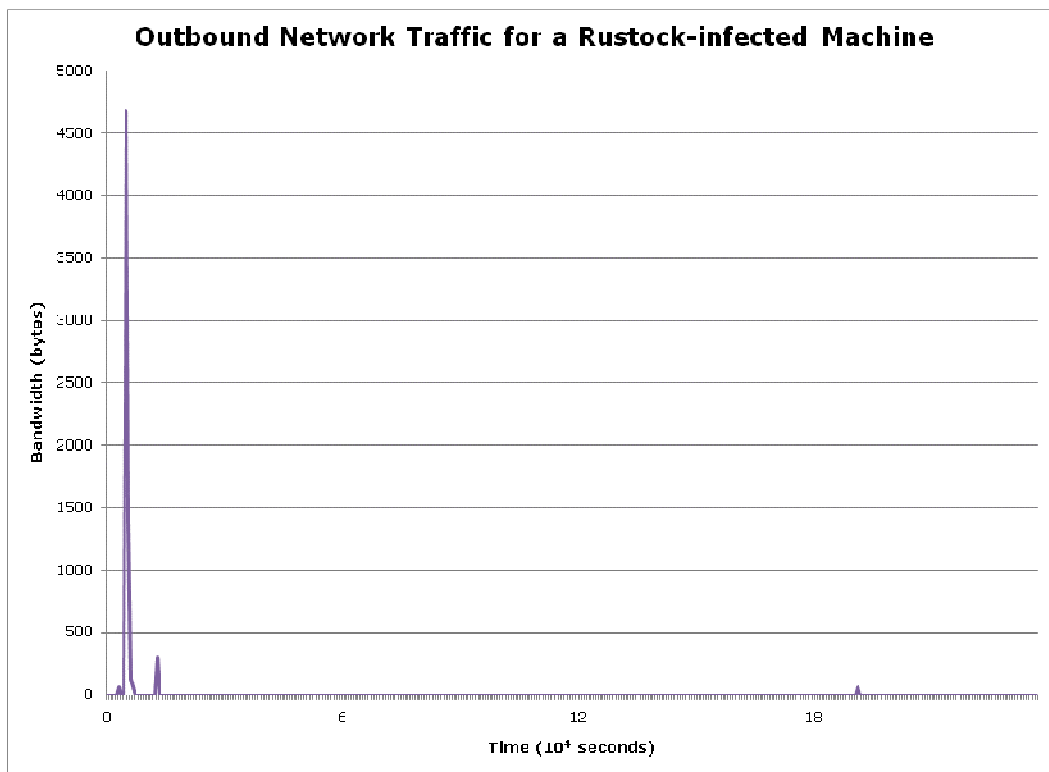
In order to provide a baseline to compare the rootkit network activity against, approximately 41 consecutive hours of network packets were captured. During this period, there was no user activity performed, in order to provide an adequate characterization of the steady-state network utilization of the operating system. During this 41-hour period, there were no outbound HTTP network packets observed. However, there were a large number of internal network protocol packets, such as Address Resolution Protocol, Cisco Discovery Protocol, NetBIOS Name Service, etc. These will not be counted as they are internal packets only.

In Figure 39, the steady-state network utilization for a machine infected with the TDL3 rootkit is presented. As can be seen, there is a significant amount of automated HTTP traffic that is generated by the TDL3 rootkit. All of the outbound HTTP traffic was directed to the IP address 174.142.51.9, which is a well-known TDL3 remote server [14]. As described in [19], an Intrusion Detection System should be able to recognize this traffic as unusual and flag it to a System Administrator.



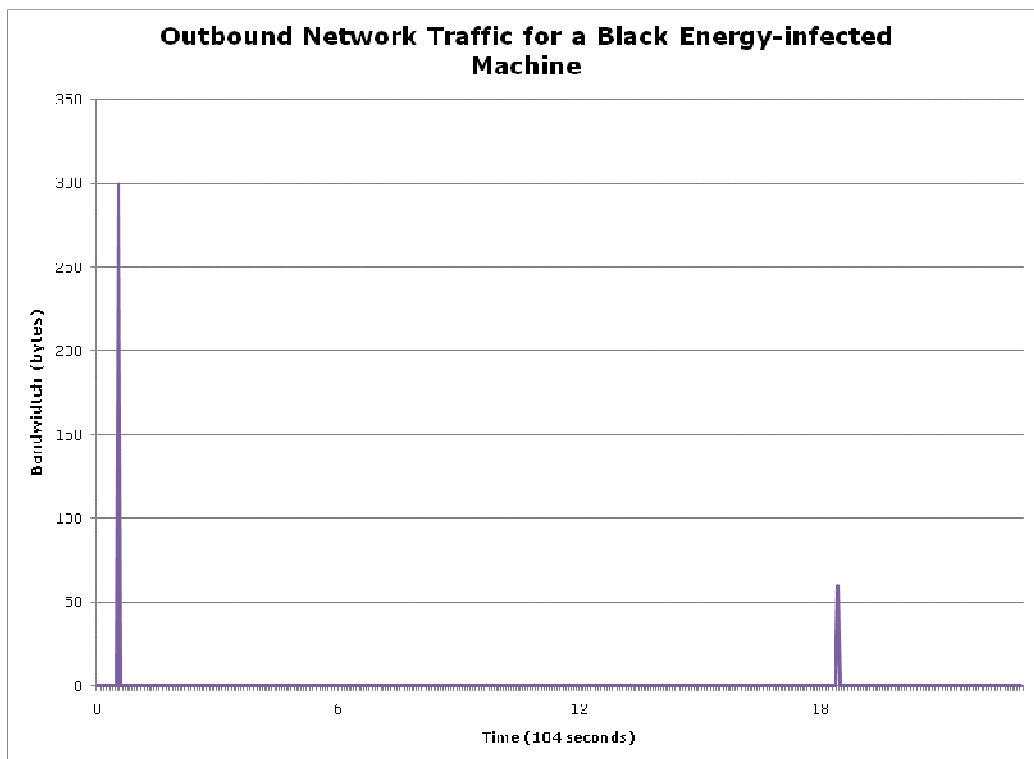
**Figure 39. Outbound Network Traffic for a TDL3-infected Machine**

In Figure 40, the steady-state network utilization for a machine infected with the Rustock rootkit is presented. As can be seen from the data, which was captured over a period of approximately 62 consecutive hours, the Rustock rootkit did not generate very much outbound HTTP traffic. However, at approximately 1.5 hours into the data capture, a brief surge in automated outbound HTTP traffic occurred. This HTTP was sent to several different IP addresses, and was likely spam emails.



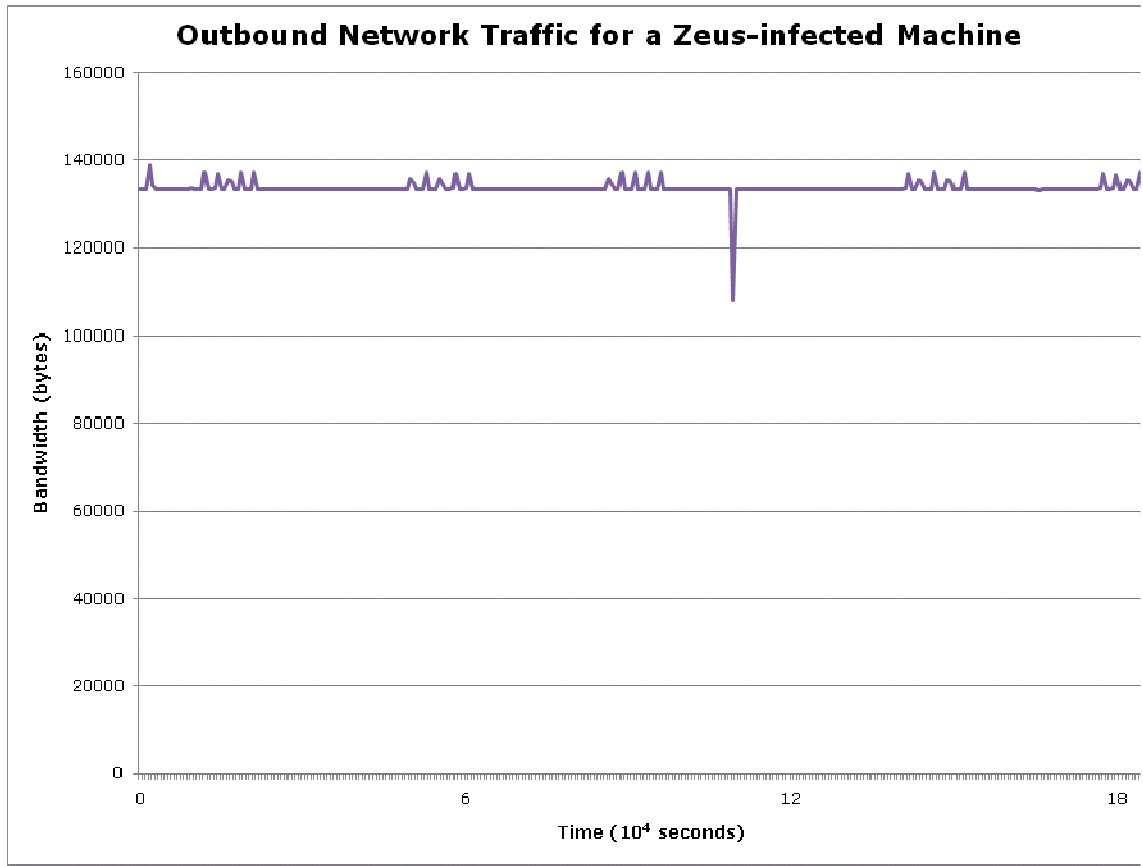
**Figure 40. Outbound Network Traffic for a Rustock-infected Machine**

In Figure 41, the steady-state network utilization for a machine infected with the Black Energy rootkit is presented. As can be seen from the data, there was virtually no automated outbound HTTP traffic generated by the Black Energy rootkit, with a couple of brief periods of communication with a remote server at IP address 207.46.141.43, which is located in Russia. This is likely communication with the botmaster for the Black Energy botnet.



**Figure 41. Network Utilization for a Black Energy-infected Machine**

In Figure 42, the network utilization for a machine infected with the Zeus rootkit is presented. As can be seen from the data, the Zeus rootkit generated a large amount of automated outbound HTTP traffic, similar to the TDL3 rootkit. The remote IP address for all of the network communication was 122.155.1.200, which is a known Zeus/Zbot command and control server located in Thailand. This activity would likely be flagged by an Intrusion Detection System, similar to the TDL3 traffic.



**Figure 42. Network Utilization for a Zeus-infected Machine**

## **7.2 Anti-Rootkit System Scans**

In this section the results of the Anti-Rootkit scanning for a clean system as well as each rootkit will be presented, followed by a ranking to show the best and worst performers for the overall dataset.

Before performing any ARK scans on rootkit-infected machines, the tools were used on a clean system to provide a baseline. In

Table 2, the results of these nominal scans are presented. The scanning time is shown, as well as any false positive results. A false positive was defined as any object (file, process, etc.) that was flagged by the ARK tool as potentially malicious.

As can be seen from the table, only F-Secure Internet Security, Microsoft Security Essentials, Rootkit Revealer, and The Cleaner reported false positive results. Both of the F-Secure and Microsoft tools reported another ARK tool (K X-ray) as malicious, which is likely due to their heuristic algorithms detecting "rootkit-like" behavior such as hooking. Rootkit Revealer reported several Windows-OS registry keys as suspicious. The Cleaner reported glmf32.dll, a Windows library for creating Open Graphics Library (OpenGL) metafiles, as suspicious.

**Table 2. Nominal (Clean) Anti-Rootkit Scan Results**

<b>Anti-Rootkit Tool</b>	<b>Scan Time (MM:SS)</b>	<b>False Positives</b>
Atool	N/A	No
Avast! Antirootkit	8:42	No
AVZ Antivirus	0:36	No
CMC Antirootkit	N/A	No
ComboFix	3:49	No
ESET SysInspector	N/A	No
F-Secure Internet Security 2011	17:15	Yes
GMER	27:10	No
Helios	N/A	No
Hidden Finder	N/A	No
Ice Sword	N/A	No
K X-ray	N/A	No
Kaspersky Internet Security 2011	16:59	No
Kernel Detective	N/A	No
Malware Bytes Anti-Malware	9:43	No
McAfee Rootkit Detective	0:35	No
Microsoft Security Essentials	49:10	Yes
Panda Internet Security 2011	14:07	No
Rootkit Revealer	0:45	Yes
Rootkit Unhooker	6:04	No
RootRepeal	0:30	No
Sophos Antirootkit	4:01	No
Spy Bot	20:23	No
Moosoft The Cleaner 2011	6:45	Yes
Trend Micro Rootkit Buster	0:20	No
VBA 32	0:49	No
XeuTr	N/A	No

In Table 3



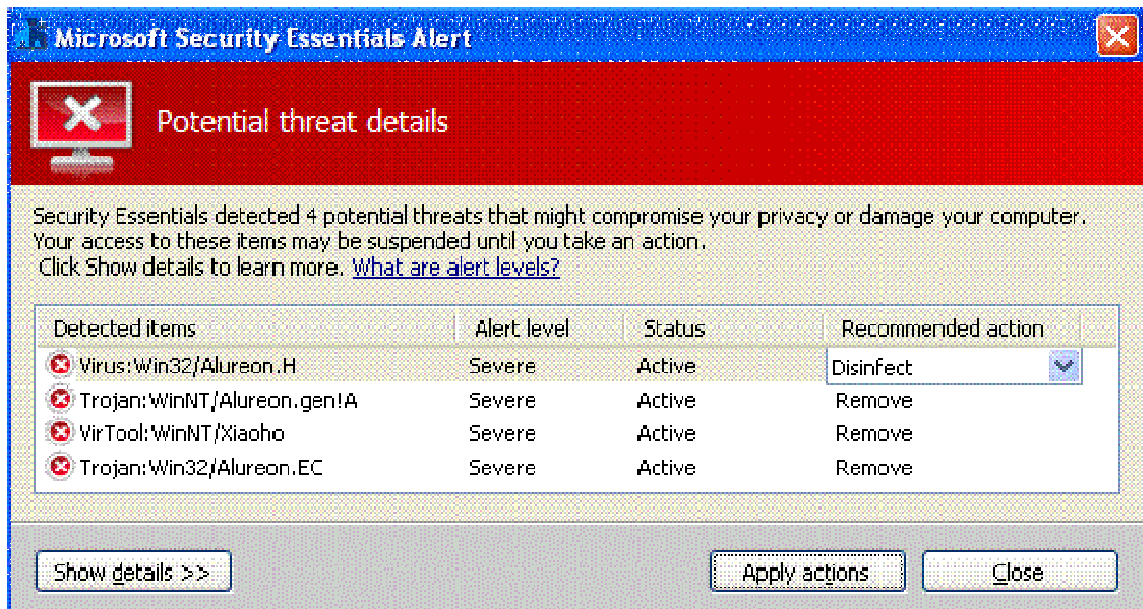
, the results of the TDL3 ARK scans are shown. The TDL3 rootkit dropper was downloaded from [malwaredomainlist.com](http://malwaredomainlist.com), a reputable source of malware that is used for research purposes. Additionally, the dropper was later uploaded to [VirusTotal.com](http://VirusTotal.com) for static analysis and was verified to be the TDL3 rootkit. The first observation that can be made from these scans is that only tools that are currently in active development were able to detect the presence of TDL3. This is not unexpected, due to the constant battle between the white hats/black hats in the development of their respective software. The authors of TDL3 have been able to figure out the various detection methods of outdated software such as Rootkit Revealer and Ice Sword, and have worked around them to remain hidden.

**Table 3. TDL3 Anti-Rootkit Scan Results**

Anti-Rootkit Tool	Detected	Scan Time (MM:SS)	False Positives	Removal
Atool	No	N/A	No	N/A
Avast! Antirootkit	No	24:49	Yes	N/A
AVZ Antivirus	No	4:41	Yes	N/A
CMC Antirootkit	Wouldn't start	N/A	N/A	N/A
ComboFix	Wouldn't start	N/A	N/A	N/A
ESET SysInspector	No	N/A	Yes	N/A
F-Secure Internet Security	Yes	19:15	No	Yes
GEMER	Yes	5:30	No	No
Helios	No	N/A	Yes	N/A
Hidden Finder	No	17:00	No	N/A
Ice Sword	No	N/A	No	N/A
K X-ray	No	N/A	No	N/A
Kaspersky Internet Security	Yes	25:00	No	Yes
Malware Bytes Anti-Malware	Yes	28:53	Yes	No
McAfee Rootkit Detective	No	0:40	No	N/A
Microsoft Security Essentials	Yes	26:00	Yes	Yes
Panda Internet Security 2011	Yes	16:15	Yes	No
Rootkit Revealer	No	1:30	Yes	N/A
Rootkit Unhooker	Yes	8:54	No	No
RootRepeal	No	N/A	Yes	N/A
Sophos Antirootkit	No	4:51	Yes	N/A
Spy Bot	Wouldn't start	N/A	N/A	N/A

Moosoft The Cleaner 2011	Yes	2:13	Yes	No
Trend Micro Rootkit Buster	No	0:05	No	N/A
VBA 32	No	0:45	No	N/A
XeuTr	No	N/A	Yes	N/A

Additionally, each of the tools that were able to detect TDL3 employ some method of self-protection. Most of these use kernel mode hooks to prevent their process or threads from being terminated by malware, as well as some other methods such as obfuscating their process name. As described earlier, TDL3 is able to actively blacklist certain anti-malware tools and undermine their successful operation. For example, Combofix and Spybot Search & Destroy would not even install, and Microsoft Security Essentials was not able to download updates to the malware definitions file. Figure 43 shows an example of the TDL3 detection by Microsoft Security Essentials.



**Figure 43. Microsoft Security Essentials Detection of TDL3**

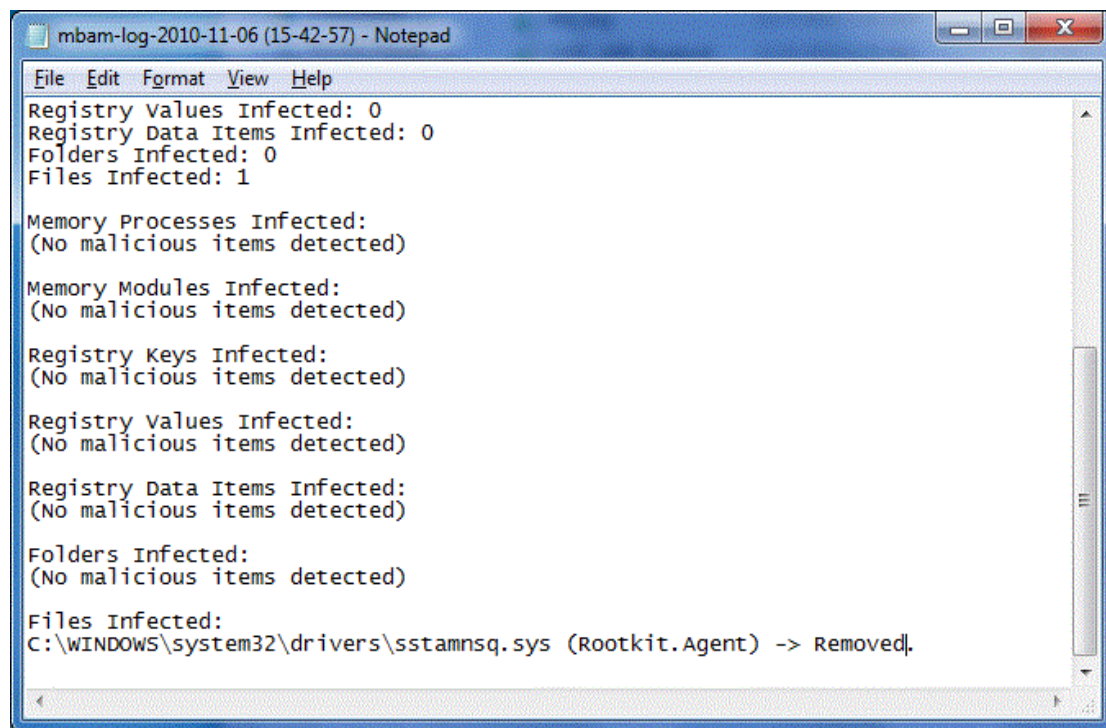
The particular detection techniques that the respective ARK tools used to successfully detect TDL3 were not clear from the output of the tools, although it appears that memory scanning and heuristic/emulation are likely important factors. Kaspersky Internet Security as well as several other of the tools use emulation to execute each of the drivers in a "sandbox" environment. If heuristic tests detect unusual behaviors such as remote network communication or unusual access to disk sectors, then the driver will be flagged as suspicious. Sysreveal was able to detect a large number (~250) of File System hooks, but was unable to remove any of them.

In Table 4, the results of the Rustock ARK scans are shown. The Rustock driver was downloaded from [www.kernelmode.info](http://www.kernelmode.info), which is a research-based website dedicated to analysis of malware and rootkits. The driver was also uploaded to Virustotal.com for static analysis and reported as Bubnix, which is the term that many antimalware software uses for the latest version of Rustock.

**Table 4. Rustock Anti-Rootkit Scan Results**

Anti-Rootkit Tool	Detected	Scan Time (MM:SS)	False Positives	Removal
Atool	No	N/A	No	N/A
Avast! Antirootkit	Yes	21:00	Yes	No
AVZ Antivirus	Yes	1:53	Yes	No
CMC Antirootkit	Yes	N/A	No	No
ComboFix	Yes	6:03	No	Yes
ESET SysInspector	No	N/A	No	N/A
F-Secure Internet Security	Yes	13:37	Yes	No
GMER	Yes	9:30	No	No
Helios Lite	No	N/A	No	N/A
Hidden Finder	No	N/A	No	N/A
Ice Sword	No	N/A	No	N/A
K X-ray	No	N/A	No	N/A
Kaspersky Internet Security	Yes	24:02	No	No
Kernel Detective	No	N/A	No	No
Malware Bytes Anti-Malware	Yes	13:10	No	Yes
McAfee Rootkit Detective	No	0:35	No	N/A
Microsoft Security Essentials	Yes	48:00	Yes	No
Panda Internet Security 2011	Yes	18:00	No	No
Rootkit Revealer	Yes	0:25	Yes	No
Rootkit Unhooker	Yes	5:00	Yes	No
RootRepeal	Yes	0:30	Yes	No
Sophos Antirootkit	Yes	4:29	No	No
Spy Bot	No	25:30	No	N/A
Sysreveal	No	N/A	No	N/A
Moosoft The Cleaner 2011	No	9:10	No	N/A
Trend Micro Rootkit Buster	No	0:10	No	N/A
VBA 32	Wouldn't run	N/A	N/A	N/A
XeuTr	Yes	N/A	Yes	No

One thing that is different from this set of scans versus the TDL3 scans is that outdated ARK tools were able to detect the infected driver and registry keys as suspicious. Rustock is not quite as sophisticated as TDL3, so it is reasonable to expect this type of result. The only tools that were able to remove the Rustock driver were Combofix and MBAM, as shown in Figure 44. The detection/removal techniques for these tools are highly proprietary, and it is not clear what differentiates them from other tools in this case.

A screenshot of a Notepad window titled "mbam-log-2010-11-06 (15-42-57) - Notepad". The window contains the following text:

```
Registry values Infected: 0
Registry Data Items Infected: 0
Folders Infected: 0
Files Infected: 1

Memory Processes Infected:
(No malicious items detected)

Memory Modules Infected:
(No malicious items detected)

Registry Keys Infected:
(No malicious items detected)

Registry Values Infected:
(No malicious items detected)

Registry Data Items Infected:
(No malicious items detected)

Folders Infected:
(No malicious items detected)

Files Infected:
C:\WINDOWS\system32\drivers\sstamnsq.sys (Rootkit.Agent) -> Removed.
```

**Figure 44. MBAM Detection of Rustock Driver**

Table 5 displays the results of the Black Energy ARK scans. The dropper was also downloaded from kernelmode.info and verified using Virustotal.com. As expected, many of the actively-developed ARK tools were able to detect various components of Black Energy, and a few of them (GMER, Rootkit Unhooker, Kernel Detective) even reported the use of the extra SSDTs.

**Table 5. Black Energy Anti-Rootkit Scans**

Anti-Rootkit Tool	Detected			Scan Time (MM:SS)	False Positives	Removal
	Registry	Driver	SSDT			
Atool	No	No	No	N/A	No	N/A
Avast! Antivirus	No	Yes	No	8:37	No	No
AVZ Antivirus	No			5:12	No	N/A
CMC Antirootkit	No	Yes	No	N/A	No	No
ComboFix	No	Yes	No	2:30	No	Yes
ESET SysInspector	No	Yes	No	N/A	No	No
F-Secure Internet Security	No			9:03	Yes	No
GMER	Yes	Yes	Yes	9:05	No	
Helios Lite	Yes	Yes	No	N/A	Yes	No
Hidden Finder	No				No	N/A
Ice Sword	No	No	No	N/A	No	N/A
K X-ray	No	No	No	N/A	No	N/A
Kaspersky Internet Security	No	Yes	No	22:30	No	Yes
Kernel Detective	No	No	Yes	N/A		No
Malware Bytes Anti-Malware	Yes	Yes	No	12:18	No	Yes
McAfee Rootkit Detective	Yes	No	No	1:05	No	No
Microsoft Security Essentials	Yes	No	No	35:00	Yes	Yes
Panda Internet Security 2011	No	Yes	No	15:22	Yes	Yes
Rootkit Revealer	Yes	No	No	0:45	Yes	No
Rootkit Unhooker	No	Yes	Yes	7:30	No	Yes
RootRepeal	No	Yes	Yes	0:50	Yes	Yes
Sophos Antirootkit	Yes			4:39	No	No
Spy Bot	No			12:50	Yes	N/A
Moosoft The Cleaner	Yes	No	No	15:47	No	No
Trend Micro Rootkit Buster	Yes	No	No	0:30	Yes	Yes
VBA 32	No			2:40	No	N/A
XeuTr	No	No	Yes	N/A	Yes	No
Sysreveal	No	No	Yes	N/A	No	No

MBAM was able to identify the kernel-mode component (str.sys) using the heuristics-based scan, and was able to detect the user-mode component with the filesystem cross-view scan. Also, several of the outdated tools such as

IceSword, McAfee Rootkit Detective, etc. were unable to detect the SSDT hooking, since they only looked at the primary 2 SSDTs. One of the more interesting observations was the detection and removal of Black Energy by Trend Micro Rootkit Buster, which is one of the older and more outdated tools. It is interesting that this tool was able to detect the registry keys via cross-view comparison and remove the offending key, ultimately killing the rootkit upon reboot. Typically older tools have not performed well against current rootkits, but this was an interesting exception. Overall, it was very interesting that the simple removal of the registry keys would prevent the rootkit from operating upon reboot. This could be an example of a “bug” due to the recent update of the rootkit software, and may very likely be fixed in the near future. An example of the detection of the faked SSDT by Rootkit Unhooker is shown in Figure 45.

```

RKU 10_25_10 - Notepad
File Edit Format View Help
0x8653B930 Faked ServiceTable-->wuauclt.exe [ ETHREAD 0x865575B0 ] TID: 1720
0x864D42C8 Faked ServiceTable-->wmiprvse.exe [ ETHREAD 0x861988C8 ] TID: 1728
0x864D42C8 Faked ServiceTable-->winlogon.exe [ ETHREAD 0x8654E468 ] TID: 1732
0x864D42C8 Faked ServiceTable-->wuauclt.exe [ ETHREAD 0x866D4B20 ] TID: 1736
0x864D42C8 Faked ServiceTable-->wuauclt.exe [ ETHREAD 0x865517C8 ] TID: 1740
0x864D42C8 Faked ServiceTable-->wmiprvse.exe [ ETHREAD 0x864F9930 ] TID: 1748
0x864D42C8 Faked ServiceTable-->wuauclt.exe [ ETHREAD 0x8655A868 ] TID: 1752
0x8653B930 Faked ServiceTable-->wuauclt.exe [ ETHREAD 0x864C5258 ] TID: 1756
0x8653B930 Faked ServiceTable-->RKUnhookerLE.EXE [ ETHREAD 0x865492A8 ] TID: 1760
0x8653B930 Faked ServiceTable-->explorer.exe [ ETHREAD 0x861C16A8 ] TID: 1816
0x864D42C8 Faked ServiceTable-->svchost.exe [ ETHREAD 0x861C7020 ] TID: 1832
0x864D42C8 Faked ServiceTable-->svchost.exe [ ETHREAD 0x86501DA8 ] TID: 1844
0x8653B930 Faked ServiceTable-->explorer.exe [ ETHREAD 0x8622BDA8 ] TID: 1860
0x8653B930 Faked ServiceTable-->csrss.exe [ ETHREAD 0x865A3520 ] TID: 1872
0x8653B930 Faked ServiceTable-->explorer.exe [ ETHREAD 0x865ACBF0 ] TID: 1952
0x8653B930 Faked ServiceTable-->svchost.exe [ ETHREAD 0x86631410 ] TID: 1960
0x864D42C8 Faked ServiceTable-->svchost.exe [ ETHREAD 0x861D0540 ] TID: 1976
0x864D42C8 Faked ServiceTable-->wmiprvse.exe [ ETHREAD 0x864A7C20 ] TID: 1984
0x864D42C8 Faked ServiceTable-->svchost.exe [ ETHREAD 0x861FF020 ] TID: 1988
0x8653B930 Faked ServiceTable-->svchost.exe [ ETHREAD 0x8623F6F8 ] TID: 2016
0x8653B930 Faked ServiceTable-->svchost.exe [ ETHREAD 0x865134B8 ] TID: 2020
=====
>Files
!-->[Hidden] C:\WINDOWS\system32\drivers\str.sys
=====

```

**Figure 45. Black Energy Detection by Rootkit Unhooker**

In Table 6, the results of the Zeus/Zbot ARK scans are shown. The Zbot dropper was also downloaded from [malwaredomainlist.com](http://malwaredomainlist.com) and verified using [VirusTotal.com](http://VirusTotal.com).

**Table 6. Zeus/Zbot Anti-Rootkit Scans**

ARK Tool	Detected			Scan Time (MM:SS)	False Positives	Removal
	Sdra64	lowsec	Reg			
Atool	No	No	No	N/A	N/A	N/A
Avast! Antivirus	No	No	No	9:42	Yes	N/A
AVZ Antivirus	No	No	No	0:41	Yes	N/A
CMC Codewalker	No	No	No	N/A	No	N/A
ComboFix	Yes	Yes	No	6:55	No	Yes
ESET SysInspector	No	No	No	N/A	No	N/A
F-Secure Internet Security 2011	Yes	No	No	19:55	Yes	Yes
GMER	No	No	No	N/A	No	N/A
Helios Lite	No	No	No	N/A	Yes	N/A
Hidden Finder	No	No	No		No	N/A
Ice Sword	Yes	Yes	No	N/A	No	No
Kernel Detective	No	No	No	N/A	No	N/A
K X-ray	No	No	No	N/A	No	N/A
Kaspersky IS 2011	No	No	No	24:00		
Malware Bytes Anti-Malware	Yes	Yes	Yes	9:26	No	Yes
McAfee Rootkit Detective	No	No	No	1:30	No	N/A
Microsoft Security Essentials	Yes	No	No	35:00	Yes	Yes
Panda IS 2011	No	No	Yes	14:50	Yes	Yes
Rootkit Revealer	No	No	No	0:30	Yes	N/A
Rootkit Unhooker	Yes	Yes	No	7:20	No	No
RootRepeal	No	No	No	1:00	No	N/A
Sophos Anti-Rootkit	Yes	No	No	4:03	No	No
Spy Bot Search and Destroy	Yes	Yes	Yes	15:40	No	Yes
SysReveal	No	No	No	N/A	No	N/A
The Cleaner 2011	Yes	No	Yes	9:00	Yes	No
Trend Micro Rootkit Buster	No	No	No	0:15	No	N/A
VBA32	No	No	No	2:30	No	N/A
XueTr	Yes	Yes	No	N/A	No	Yes



As can be seen from the table, ten of the detectors (approximately a third) were able to detect the presence of the hidden driver, the configuration files, or the registry keys. Out of those, only five detectors were able to completely remove the rootkit driver, files, and registry keys. Microsoft Security Essentials and F-Secure were able to detect and remove the hidden sdra64.exe driver, but did not remove the configuration files or registry keys. However, it should be noted that the removal of sdra64.exe effectively kills the rootkit, as it cannot copy itself into running processes. MBAM was able to only detect the sdra64.exe component using the filesystem cross-view scan, but was able to detect the lowsec directory and associated files once the heuristics-based scan was enabled.

Despite being one of the better performing tools, GMER was unable to complete its scan, and crashed after approximately 20 minutes of operation. This is a good example of the system instabilities that can occur when rootkits and anti-rootkit tools are utilizing low-level kernel data structures.

Removal of the Zeus rootkit was confirmed by rebooting and performing subsequent scans of corroborating tools, as well as observing the lack of certain behaviors, such as the hiding of the System32/lowsec directory and the lack of the backdoor TCP port associated with Winlogon.exe or Svchost.exe.

In Table 7, an overall ranking of the ARK tools is presented, based on their performance at detecting and removing rootkits, as well as reporting false positives. A simple scoring system was used: one point was given for

successful detection, one point was given for successful removal, and one point was taken away for each false positive that was reported. Typically the ARK tool would report the same false positive across all the tests, and this was counted only once.

**Table 7. Overall Ranking of ARK Tools**

<b>Anti-Rootkit Tool</b>	<b>Detection</b>	<b>Removal</b>	<b>False Positives</b>	<b>Overall Score</b>
Malware Bytes Anti-Malware	4	3	-1	6
Combofix	3	3	0	6
Kaspersky Internet Security 2011	3	2	0	5
Panda Internet Security 2011	4	2	-1	5
Microsoft Security Essentials	4	2	-1	5
F-Secure Internet Security 2011	3	2	-1	4
Rootkit Unhooker	4	0	0	4
GMER	3	0	0	3
CMC Antirootkit	2	0	0	2
RootRepeal	2	1	-1	2
Sophos Antirootkit	3	0	-1	2
Moosoft The Cleaner 2011	3	0	-1	2
XeuTr	3	0	-1	2
Avast! Antirootkit	2	1	-2	1
Ice Sword	1	0	0	1
Kernel Detective	1	0	0	1
McAfee Rootkit Detective	1	0	0	1
Rootkit Revealer	2	0	-1	1
Spy Bot	1	1	-1	1
Trend Micro Rootkit Buster	1	1	-1	1
Sysreveal	1	0	0	1
Atool	0	0	0	0
ESET SysInspector	1	0	-1	0
Helios	1	0	-1	0
Hidden Finder	0	0	0	0
K X-ray	0	0	0	0
VBA 32	0	0	0	0
AVZ Antivirus	0	0	-1	-1

As can be seen from Table 7, the top performing ARK tools were those that are still in active development, and many of the worst performing tools were no longer being actively updated. Also, many of best performers were

“Internet Security” tools that performed a variety of malware detection tasks and used several different detection methods.

One feature that many of the top performing tools share is the use of heuristics to detect new version of malware. Some “isolation” testing was performed on several of the top-performing tools to determine which detection technique was driving the results. In these tests, heuristics did make a difference in the detection of varying components, such as the Black Energy kernel-mode component (str.sys) and the Zbot lowsec directory and associated files.

Also, these tools tended to have much longer scanning times than the lower-performing detectors. On the surface, the additional scanning time could be considered as poor efficiency/performance; however, it is more likely that these tools are performing much deeper looks at the filesystem and applying heuristics-based techniques on the files, which would take longer than a traditional “cross-view” type of scan. Finally, it should be noted that the best performing applications hooked Windows services to provide better real-time protection, as well as self protection for the ARK tool.

### **7.3 Network-Based Detection**

In this section, the results of the Netstat and Nmap operations are displayed as a series of window captures. After each set of window captures, a description of the results and will be provided. The initial window captures are for a clean system, followed by a set of window captures from a system infected by the Hacker Defender rootkit. After that, a set of window captures will be provided for each of the rootkits used in the main thesis research (Rustock, TDL3, Black Energy, and Zeus/Zbot).

Figure 46 and Figure 47 displays the output of Netstat and Nmap against an uninfected, clean system. This provides a baseline for the remaining rootkit scans. As can be seen from the output, there was a total of 10 non-loopback ports reported by Netstat. Nmap was able to detect all of these, and associate a service with each of them. Based on the lack of discrepancies, it can be inferred that no rootkits are hiding network activity.

```

clean_netstat - Notepad
File Edit Format View Help
Active Connections

Proto Local Address          Foreign Address        State
TCP    0.0.0.0:135             0.0.0.0:0              LISTENING
TCP    0.0.0.0:445             0.0.0.0:0              LISTENING
TCP    127.0.0.1:1027          0.0.0.0:0              LISTENING
TCP    192.168.10.114:139     0.0.0.0:0              LISTENING
UDP    0.0.0.0:445             *:*:                    *:*
UDP    0.0.0.0:500             *:*:                    *:*
UDP    0.0.0.0:4500           *:*:                    *:*
UDP    127.0.0.1:123          *:*:                    *:*
UDP    127.0.0.1:1900         *:*:                    *:*
UDP    192.168.10.114:123    *:*:                    *:*
UDP    192.168.10.114:137    *:*:                    *:*
UDP    192.168.10.114:138    *:*:                    *:*
UDP    192.168.10.114:1900   *:*:                    *:*

```

**Figure 46. Clean System Netstat Output**

```

clean_nmap - Notepad
File Edit Format View Help
Starting Nmap 5.50 ( http://nmap.org ) at 2011-02-01| 22:27 Central Day
Nmap scan report for 192.168.10.114
Host is up (0.00s latency).
Not shown: 131060 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
123/udp   open  ntp
137/udp   open  netbios-ns
138/udp   open|filtered netbios-dgm
445/udp   open|filtered microsoft-ds
500/udp   open|filtered isakmp
1900/udp  open|filtered upnp
4500/udp  open|filtered nat-t-ike
MAC Address: 00:01:80:3B:8D:52 (Aopen)

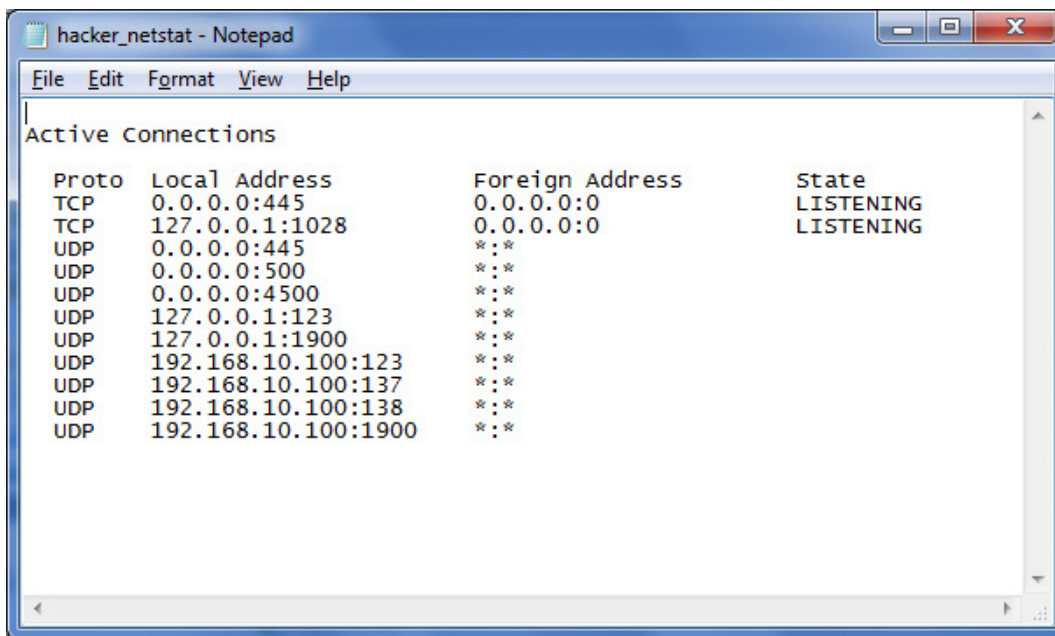
Nmap done: 1 IP address (1 host up) scanned in 23.13 seconds

```

**Figure 47. Clean System Nmap Port Scan**

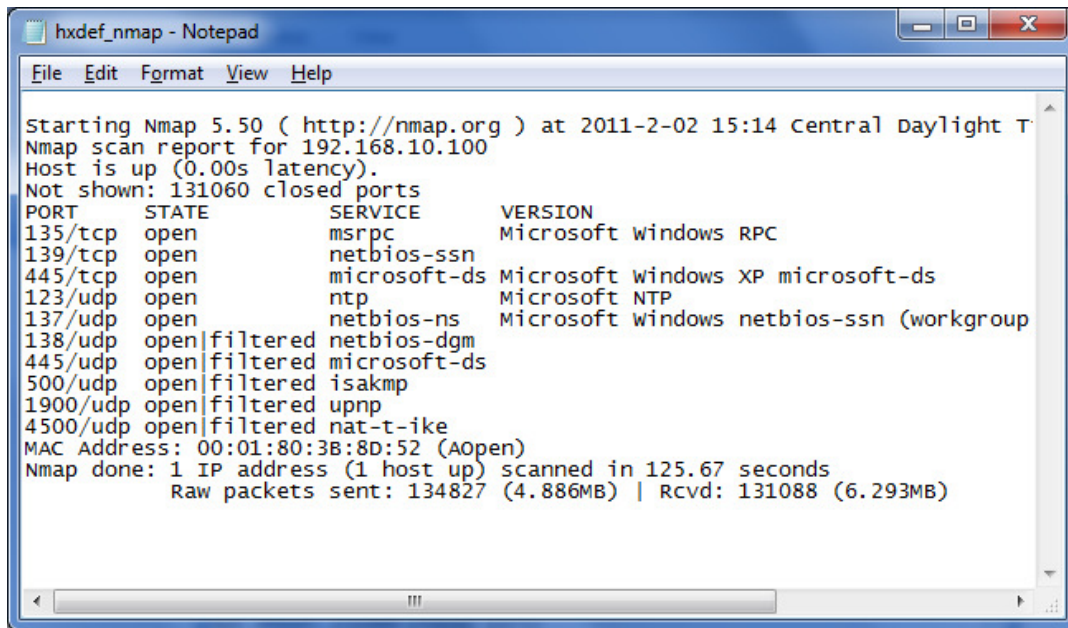
Figure 48 and Figure 49 display the output of Netstat and Nmap against a system infected with the Hacker Defender rootkit. This 2004-era rootkit is well-known to have the ability to hide network ports, which should demonstrate the efficacy of the network-based detection technique. For this

example, Hacker Defender was configured to hide TCP ports 135 and 139. As can be seen from the output in Figure 48, Netstat detected 8 non-loopback network ports, and TCP ports 135 and 139 were not reported. However, in Figure 49, it can be seen that Nmap was able to detect 10 TCP and UDP ports, including 135 and 139. Given the discrepancy in the output, assuming no other information was available, it would be very likely that a rootkit was hiding network activity from the local user.



```
hacker_netstat - Notepad
File Edit Format View Help
Active Connections
Proto Local Address Foreign Address State
TCP 0.0.0.0:445 0.0.0.0:0 LISTENING
TCP 127.0.0.1:1028 0.0.0.0:0 LISTENING
UDP 0.0.0.0:445 *:*
UDP 0.0.0.0:500 *:*
UDP 0.0.0.0:4500 *:*
UDP 127.0.0.1:123 *:*
UDP 127.0.0.1:1900 *:*
UDP 192.168.10.100:123 *:*
UDP 192.168.10.100:137 *:*
UDP 192.168.10.100:138 *:*
UDP 192.168.10.100:1900 *:*
```

**Figure 48. Hacker Defender Netstat Output**



```

Starting Nmap 5.50 ( http://nmap.org ) at 2011-2-02 15:14 Central Daylight T
Nmap scan report for 192.168.10.100
Host is up (0.00s latency).
Not shown: 131060 closed ports
PORT      STATE        SERVICE          VERSION
135/tcp   open         msrpc            Microsoft windows RPC
139/tcp   open         netbios-ssn     Microsoft windows netbios-ssn (workgroup)
445/tcp   open         microsoft-ds    Microsoft windows XP microsoft-ds
123/udp   open         ntp              Microsoft NTP
137/udp   open         netbios-ns      Microsoft windows netbios-ssn (workgroup)
138/udp   open|filtered netbios-dgm
445/udp   open|filtered microsoft-ds
500/udp   open|filtered isakmp
1900/udp  open|filtered upnp
4500/udp  open|filtered nat-t-ike
MAC Address: 00:01:80:3B:8D:52 (AOpen)
Nmap done: 1 IP address (1 host up) scanned in 125.67 seconds
Raw packets sent: 134827 (4.886MB) | Rcvd: 131088 (6.293MB)

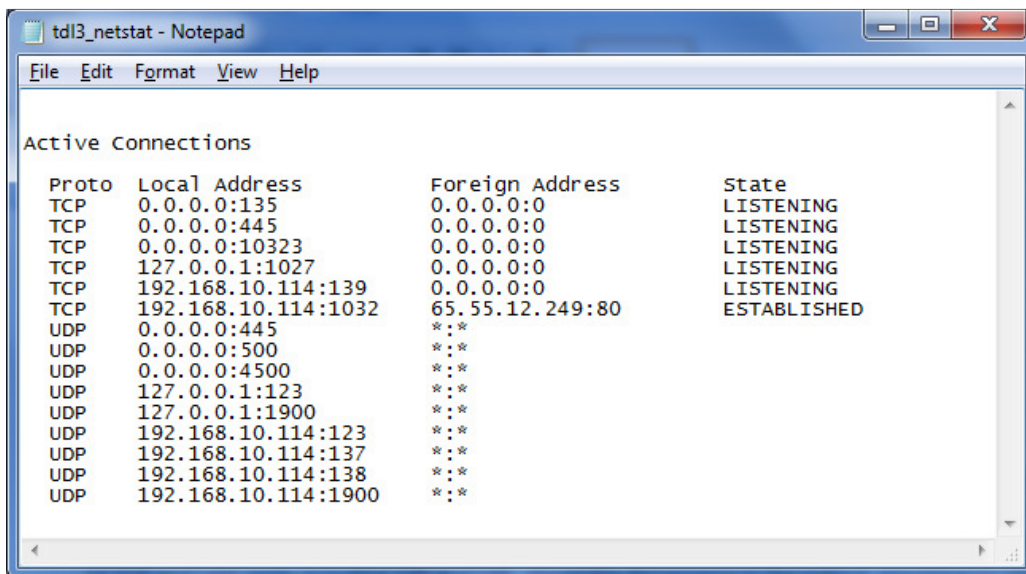
```

**Figure 49. Hacker Defender Nmap Port Scan**

Figure 50 and Figure 51 display the output of Netstat and Nmap against a system infected with the TDL3 rootkit. As can be seen from the output in Figure 50, Netstat detected 12 non-loopback network ports. A couple of differences from the baseline can be noted. First, a service on TCP port 10323 is listening for a connection, which is likely a backdoor. Additionally, a connection to a Microsoft Hotmail IP address has been established, again this is a possible backdoor method to communicate with a botmaster. There is no research to provide this; however, this is a consistent network signature with TDL3 infections, and this connection must be used in connection with the botnet in some way.

In Figure 51, the output of Nmap can be seen. It was able to detect 11 open or listening TCP/UDP ports, and as expected could not detect the established Hotmail connection as described in the previous paragraph. Based on a

comparison between Netstat and Nmap, it appears that TDL3 does not attempt to hide any network ports. This does not mean that there is no malicious network activity, but as described in the Introduction, many botnet/rootkit authors are no longer hiding the network ports.



```

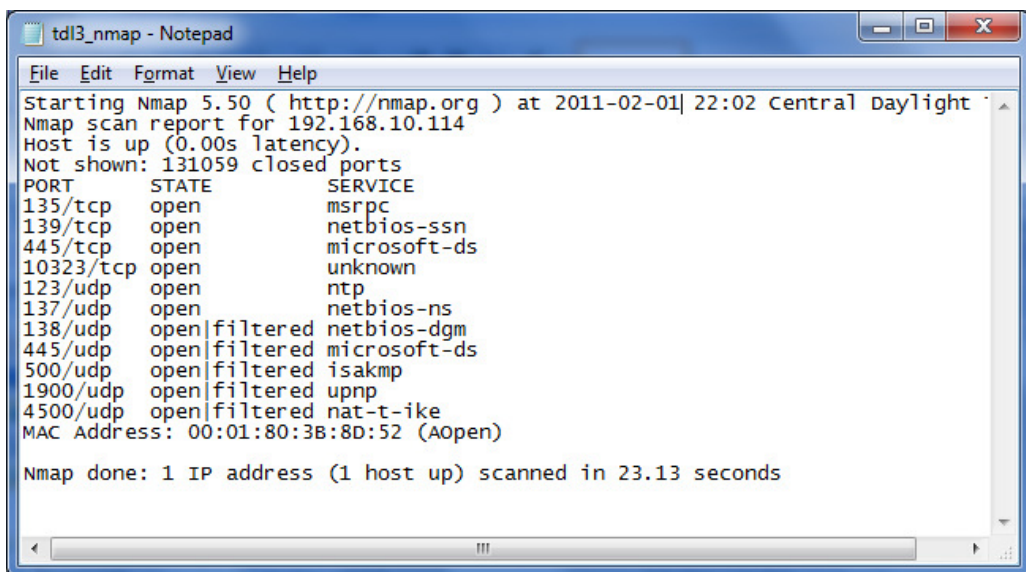
tdl3_netstat - Notepad
File Edit Format View Help

Active connections

Proto Local Address      Foreign Address    State
TCP    0.0.0.0:135           0.0.0.0:0         LISTENING
TCP    0.0.0.0:445           0.0.0.0:0         LISTENING
TCP    0.0.0.0:10323        0.0.0.0:0         LISTENING
TCP    127.0.0.1:1027       0.0.0.0:0         LISTENING
TCP    192.168.10.114:139   0.0.0.0:0         LISTENING
TCP    192.168.10.114:1032 65.55.12.249:80   ESTABLISHED
UDP    0.0.0.0:445          *:*
UDP    0.0.0.0:500          *:*
UDP    0.0.0.0:4500         *:*
UDP    127.0.0.1:123        *:*
UDP    127.0.0.1:1900       *:*
UDP    192.168.10.114:123  *:*
UDP    192.168.10.114:137  *:*
UDP    192.168.10.114:138  *:*
UDP    192.168.10.114:1900 *:*

```

**Figure 50. TDL3 Netstat Output**



```

tdl3_nmap - Notepad
File Edit Format View Help

Starting Nmap 5.50 ( http://nmap.org ) at 2011-02-01 22:02 Central Daylight
Nmap scan report for 192.168.10.114
Host is up (0.00s latency).
Not shown: 131059 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
10323/tcp open  unknown
123/udp   open  ntp
137/udp   open  netbios-ns
138/udp   open|filtered netbios-dgm
445/udp   open|filtered microsoft-ds
500/udp   open|filtered isakmp
1900/udp  open|filtered upnp
4500/udp  open|filtered nat-t-ike
MAC Address: 00:01:80:3B:8D:52 (AOpen)

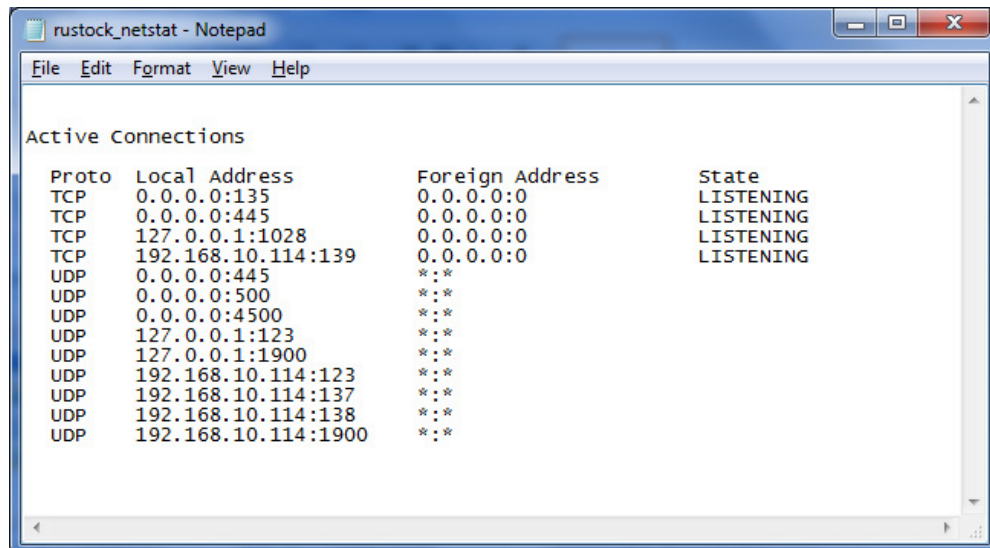
Nmap done: 1 IP address (1 host up) scanned in 23.13 seconds

```

**Figure 51. TDL3 Nmap Port Scan**



Figure 52 and Figure 53 display the output of Netstat and Nmap against a system infected with the Rustock rootkit. As can be seen from the output in Figure 52, Netstat detected 10 non-loopback network ports, which were all detected by Nmap in Figure 53. There appear to be no differences between this set and the baseline set. However, this Rustock-infected machine has been observed to perform suspicious connections upon bootup, but this is the steady-state network performance of the machine, and there appear to be no hidden connections or active backdoors. However, Rustock has been observed to perform spamming operations on a cyclical basis [7], so before any conclusions can be drawn, the network activity of the rootkit/botnet should be observed on a more extended basis.

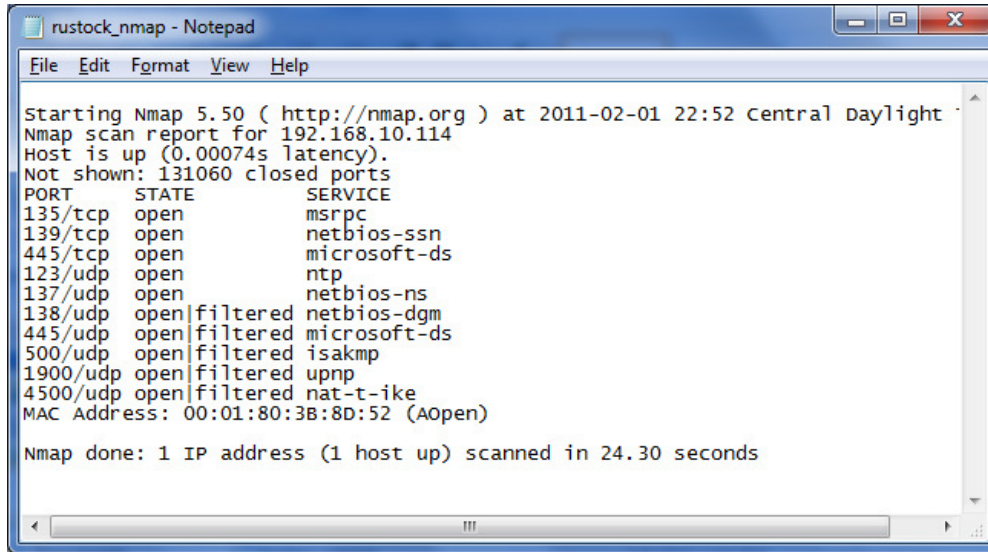


```
rustock_netstat - Notepad
File Edit Format View Help

Active Connections

Proto Local Address      Foreign Address    State
TCP   0.0.0.0:135         0.0.0.0:0         LISTENING
TCP   0.0.0.0:445         0.0.0.0:0         LISTENING
TCP   127.0.0.1:1028      0.0.0.0:0         LISTENING
TCP   192.168.10.114:139  0.0.0.0:0         LISTENING
UDP   0.0.0.0:445         *:                *:*
UDP   0.0.0.0:500         *:                *:*
UDP   0.0.0.0:4500        *:                *:*
UDP   127.0.0.1:123       *:                *:*
UDP   127.0.0.1:1900      *:                *:*
UDP   192.168.10.114:123  *:                *:*
UDP   192.168.10.114:137  *:                *:*
UDP   192.168.10.114:138  *:                *:*
UDP   192.168.10.114:1900 *:                *:*
```

**Figure 52. Rustock Netstat Output**



```

rustock_nmap - Notepad
File Edit Format View Help
Starting Nmap 5.50 ( http://nmap.org ) at 2011-02-01 22:52 central Daylight
Nmap scan report for 192.168.10.114
Host is up (0.00074s latency).
Not shown: 131060 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
123/udp   open  ntp
137/udp   open  netbios-ns
138/udp   open|filtered netbios-dgm
445/udp   open|filtered microsoft-ds
500/udp   open|filtered isakmp
1900/udp  open|filtered upnp
4500/udp  open|filtered nat-t-ike
MAC Address: 00:01:80:3B:8D:52 (Aopen)

Nmap done: 1 IP address (1 host up) scanned in 24.30 seconds

```

**Figure 53. Rustock Nmap Port Scan**

Figure 54 and Figure 55 display the output of Netstat and Nmap against a system infected with the Black Energy rootkit. As can be seen from the output in Figure 54, Netstat detected 11 non-loopback network ports. One difference from the baseline was an established UDP connection on port 58341, which was a likely backdoor for the Black Energy botmaster. Otherwise, there appear to be no differences between this set and the baseline. In Figure 55, Nmap was able to detect all the open or listening TCP and UDP ports, and as expected, was not able to detect the open UDP port. Based on this set, it appears that while Black Energy does have a backdoor UDP port, the rootkit does not attempt to hide any of its network activity.

```

blackenergy_netstat - Notepad
File Edit Format View Help

Active Connections

Proto Local Address          Foreign Address         State
TCP   0.0.0.0:135              0.0.0.0:0              LISTENING
TCP   0.0.0.0:445              0.0.0.0:0              LISTENING
TCP   127.0.0.1:1029           0.0.0.0:0              LISTENING
TCP   192.168.10.114:139      0.0.0.0:0              LISTENING
UDP   0.0.0.0:445              *:*
UDP   0.0.0.0:500              *:*
UDP   0.0.0.0:4500             *:*
UDP   0.0.0.0:58341           *:*
UDP   127.0.0.1:123           *:*
UDP   127.0.0.1:1900          *:*
UDP   192.168.10.114:123     *:*
UDP   192.168.10.114:137     *:*
UDP   192.168.10.114:138     *:*
UDP   192.168.10.114:1900    *:*
```

**Figure 54. Black Energy Netstat Output**

```

blackenergy_nmap - Notepad
File Edit Format View Help

Starting Nmap 5.50 ( http://nmap.org ) at 2011-02-01| 22:33 Central Daylight
Nmap scan report for 192.168.10.114
Host is up (0.00s latency).
Not shown: 131060 closed ports
PORT      STATE      SERVICE
135/tcp   open      msrpc
139/tcp   open      netbios-ssn
445/tcp   open      microsoft-ds
123/udp   open      ntp
137/udp   open      netbios-ns
138/udp   open|filtered netbios-dgm
445/udp   open|filtered microsoft-ds
500/udp   open|filtered isakmp
1900/udp  open|filtered upnp
4500/udp  open|filtered nat-t-ike
MAC Address: 00:01:80:3B:8D:52 (Aopen)

Nmap done: 1 IP address (1 host up) scanned in 23.45 seconds
```

**Figure 55. Black Energy Nmap Port Scan**

Figure 56 and Figure 57 display the output of Netstat and Nmap against a system infected with the Zeus/Zbot rootkit. As can be seen from the output in Figure 56, Netstat detected 11 non-loopback network ports. One difference from the baseline was a listening TCP connection on port 21470, which was a likely backdoor for the Zeus botmaster. Otherwise, there

appear to be no differences between this set and the baseline. In Figure 57, Nmap was able to detect all the open or listening TCP and UDP ports, including the backdoor TCP port. Based on this set, it appears that while Zeus does have a backdoor UDP port, the rootkit does not attempt to hide any of its network activity.

```

Active Connections

Proto Local Address      Foreign Address    State
TCP   0.0.0.0:135         0.0.0.0:0         LISTENING
TCP   0.0.0.0:445         0.0.0.0:0         LISTENING
TCP   0.0.0.0:21470      0.0.0.0:0         LISTENING
TCP   127.0.0.1:1029     0.0.0.0:0         LISTENING
TCP   192.168.10.114:139 0.0.0.0:0         LISTENING
UDP   0.0.0.0:445        *.*               *.*
UDP   0.0.0.0:500        *.*               *.*
UDP   0.0.0.0:4500       *.*               *.*
UDP   127.0.0.1:123     *.*               *.*
UDP   127.0.0.1:1900    *.*               *.*
UDP   192.168.10.114:123 *.*               *.*
UDP   192.168.10.114:137 *.*               *.*
UDP   192.168.10.114:138 *.*               *.*
UDP   192.168.10.114:1900 *.*               *.*

```

**Figure 56. Zeus Netstat Output**

```

Starting Nmap 5.50 ( http://nmap.org ) at 2010-02-01| 22:13 Central Daylight
Nmap scan report for 192.168.10.114
Host is up (0.00s latency).
Not shown: 131059 closed ports
PORT      STATE      SERVICE
135/tcp   open      msrpc
139/tcp   open      netbios-ssn
445/tcp   open      microsoft-ds
21470/tcp open      unknown
123/udp   open      ntp
137/udp   open      netbios-ns
138/udp   open|filtered netbios-dgm
445/udp   open|filtered microsoft-ds
500/udp   open|filtered isakmp
1900/udp  open|filtered upnp
4500/udp  open|filtered nat-t-ike
MAC Address: 00:01:80:3B:8D:52 (Aopen)

Nmap done: 1 IP address (1 host up) scanned in 21.81 seconds

```

**Figure 57. Zeus Nmap Port Scan**

## 8.0 Conclusions

Rootkits are a significant threat to information security, as was observed in the set of system performance observations in this thesis research. The resulting network and system performance impacts, as well as the potential loss of sensitive information, can be disastrous for an individual user or organization. This thesis analyzed a large number of different rootkit detection applications and techniques in order to determine the best methods to neutralize the most recent rootkit threats.

The results of the ARK scans highlight the need to use actively-developed tools in attempting the detection and removal of the latest rootkits. Out of the 28 ARK tools that were used in the research, the top 8 were all still being updated to reflect the most recent trends in malware development. Additionally, the best performing tools were those that utilized multiple detection techniques to identify malware. Most notably, the common characteristics of the top performing ARK tools were the use of heuristics-based detection, as well as hooking Windows services to provide better ARK tool self-protection and real-time detection of malware. Some follow-on testing demonstrated that heuristics did make a difference in detecting some rootkit components which were not detected by other methods such as memory or filesystem cross-view scanning.

In addition to performing a large number of ARK scans, the network-based "cross-view" rootkit detection method was demonstrated by comparing the output of a local, API-driven application (Netstat) versus an external port

scanner (Nmap). The method was demonstrated to be successful in detecting the hidden ports from the Hacker Defender rootkit. However, it appears that none of the modern rootkits included in the thesis research make an attempt to hide their network port activity. With the exception of the Rustock rootkit, each of the others (TDL3, Black Energy, and Zeus/Zbot) appeared to have active backdoor ports able to connect to remote servers. While seemingly counterintuitive to the idea of a stealth rootkit, this finding does seem to agree with recent analysis by subject matter experts [18].

While it appears that the network-based procedure may not always detect the presence of a rootkit, it should still be included in the standard practice of a forensic investigator. The fact that 3 of the 4 rootkits were observed to have active backdoor ports in place would likely arouse suspicion and further investigation, which could lead to the detection of the malware. Additionally, this type of "cross-view" technique could be automated and used in concert with Intrusion Detection Systems such as Web Tap [19] to provide more complete coverage from a network perspective.

## **8.1 Future Research**

Based on the results of the ARK scans, further research should focus on developing an optimal set of heuristic-based rules to detect rootkit activity, which maximizes the rate of detection while minimizing the rate of false positives. By focusing on dynamic behavior, it is likely that an ARK developer will keep up with the latest threats and provide better overall security.

## REFERENCES

- [1] Alien Registry Viewer (3.5.567) [Software], 2010, Available from <http://lastbit.com/arv/>
- [2] "AV-Comparatives," AV-Comparatives, Web, Accessed on 21 March 2011, <http://www.av-comparatives.org/>.
- [3] "Backdoor.Tdss.565 and its modifications (aka TDL3)", Dr. Web, 2009, Retrieved from [http://www.drweb.com/static/BackDoor.Tdss.565\\_\(aka%20TDL3\)\\_en.pdf](http://www.drweb.com/static/BackDoor.Tdss.565_(aka%20TDL3)_en.pdf)
- [4] IceSword (1.22en) [Software], Available from <http://www.softpedia.com/get/System/System-Info/IceSword.shtml>
- [5] Kaspersky Internet Security (2010) [Software], Available from <http://www.kaspersky.com/>
- [6] MalwareBytes Anti-Malware (1.50) [Software], Available from <http://www.malwarebytes.org/mbam.php>
- [7] Microsoft Security Essentials (2.0.657.0) [Software], Available from [http://www.microsoft.com/security\\_essentials](http://www.microsoft.com/security_essentials)
- [8] Microsoft. (June 14, 2010). "Microsoft Security Intelligence Report," Vol. 8.
- [9] Netstat application [Software], Available from <http://en.wikipedia.org/wiki/Netstat>
- [10] Partimage Is Not Ghost (PING) (3.00.04) [Software], Available from <http://ping.windowdream.com/>
- [11] RegSnap (7.0.2084) [Software], 2010, Available from <http://lastbit.com/regsnap/>
- [12] "Ring (computer security)", Retrieved from [http://en.wikipedia.org/wiki/Ring\\_\(computer\\_security\)](http://en.wikipedia.org/wiki/Ring_(computer_security))
- [13] Rootkit Unhooker (3.8.388.480 SR2) [Software], Available from <http://www.antirootkit.com/software/RootKit-Unhooker.htm>
- [14] "ThreatExpert Report - TDSS sample," Web, Accessed on 21 March 2011, <http://www.threatexpert.com/report.aspx?md5=ccab8b016f9372fc47eeb83df0f63dd9>.
- [15] Tripwire [Software], Available from <http://www.tripwire.com>
- [16] WinDiff (5.2) [Software], 1992, Available from <http://en.wikipedia.org/wiki/WinDiff>
- [17] Bensalleeh H, Ormerod T, and al e, "On the Analysis of the Zeus Botnet Crimeware Toolkit," *Proceedings from Eighth Annual Conference on Privacy, Security, and Trust*, Ottawa, ON, Canada: IEEE Press, 2010.
- [18] Blunden B, *The Rootkit Arsenal: Escape and Evasion in the Dark Corners of the System*. Plano, TX: Wordware Publishing, 2009.
- [19] Borders K, and Prakash A, "Web Tap: Detecting Covert Web Traffic," *Proceedings from 11th ACM Conference on Computer and Communications Security*, New York, NY: 2004.
- [20] Cox A, and Golomb G, "The "Kneber" Botnet: A Zeus Discovery and Analysis", Netwitness, Feb. 18, 2010, Retrieved from

- <http://www.netwitness.com/resources/downloads/2011-the-kneber-botnet>
- [21] Davis J. *"Using Wireshark to Create Network-Usage Baselines,"* Georgia Tech Research Institute, 2007.
- [22] Davis M, Bodmer S, and LeMasters A, *Hacking Exposed: Malware and Rootkits*. New York: McGraw-Hill, 2009.
- [23] Falliere N, and Chien E, *"Zeus: King of the Bots"*, Symantec, Feb. 20, 2010, Retrieved from [http://www.symantec.com/content/en/us/enterprise/media/security\\_response/whitepapers/zeus\\_king\\_of\\_bots.pdf](http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/zeus_king_of_bots.pdf)
- [24] Gmerek P, GMER (1.0.15.15281) [Software], 2010, Available from <http://www.gmer.net/>
- [25] Gorman S, and Perez E, (December 22, 2009), "FBI Probes Hack at Citibank," *Wall Street Journal*.
- [26] Hay P, *"Spam Volumes Drop After Spamit Shakeup"*, M86 Security, October 2010, Retrieved from <http://labs.m86security.com/2010/10/spam-volumes-drop-after-spamit-shakeup/>
- [27] Horn D, Darik's Boot and Nuke (DBAN) (1.0.7) [Software], Available from <http://www.dban.org/>
- [28] Ionescu A, *"NT Internals,"* Accessed on <http://www.ntinternals.org/index.php>.
- [29] Jogie N, *"Rootkit Analysis – Hiding SSDT Hooks"*, Securabit, March 31, 2010, Retrieved from <http://www.securabit.com/2010/03/31/rootkit-analysis-hiding-ssdt-hooks/>
- [30] Jones A, (September 30, 2010), "Zeus Trojan and Money Moles: More Reasons to Scan Email Carefully," *Wall Street Journal*.
- [31] Lagerweij B, Bart's Preinstalled Environment (BartPE) (3.1.10a) [Software], 2006, Available from <http://www.nu2.nu/pebuilder/>
- [32] Lanstein A, *"An Overview of Rustock"*, FireEye Malware Intelligence, Mar. 19, 2011, Retrieved from <http://blog.fireeye.com/research/2011/03/an-overview-of-rustock.html>
- [33] Lyon G, Network Mapper (Nmap) Application (5.51) [Software], Available from <http://www.nmap.org>
- [34] Macdonald D, *"Zeus: God of DIY Botnets"*, Oct. 14, 2009, Retrieved from <http://www.fortiguard.com/analysis/zeusanalysis.html>
- [35] Matrosov A, and Rodionov E, *"TDL3: The Rootkit of All Evil?"*, ESET Security, 2009, Retrieved from <http://www.eset.com/resources/white-papers/TDL3-Analysis.pdf>
- [36] Mendrez R, *"Revisiting the King of Spam"*, M86 Security, July 23, 2010, Retrieved from <http://www.m86security.com/labs/traceitem.asp?article=1362>
- [37] Ries C, *"Inside Windows Rootkits"*, Vigilant Minds, Inc., May 22, 2006, Retrieved from <http://madchat.fr/vxdevl/library/Inside%20Windows%20Rootkits.pdf>



- [38] Russinovich M, Rootkit Revealer (1.7) [Software], Available from <http://technet.microsoft.com/en-us/sysinternals/bb897445.aspx>
- [39] Russinovich M, and Solomon D, *Windows Internals, Fifth Edition*. Redmond, WA: Microsoft Press, 2009.
- [40] Son N, "TDL3: Part I. A detailed analysis of TDL rootkit 3rd generation", CMCInfosec, 2009, Retrieved from <http://www.layer8howto.net/wordpress/wp-content/uploads/2010/02/tld3-analysis-paper.pdf>
- [41] 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.
- [42] Stewart J, "Black Energy Version 2 Analysis", SecureWorks, March 3, 2010, Retrieved from <http://www.secureworks.com/research/threats/blackenergy2/?threat=blackenergy2>
- [43] Stewart J, "Rustock DDoS Attack", SecureWorks, Feb. 2007, Retrieved from <http://www.joestewart.org/rustock-ddos.html>
- [44] Szor P. (2010). *U.S. Patent No. 11/271327*. Washington, DC: U.S. Patent and Trademark Office.
- [45] Treit R, "Some Observations on Rootkits", Microsoft, January 7, 2010, Retrieved from <http://blogs.technet.com/b/mmpc/archive/2010/01/07/some-observations-on-rootkits.aspx>
- [46] Williams J, "Operation b107 - Rustock Botnet Takedown", Microsoft Malware Protection Center, Mar. 17, 2011, Retrieved from <http://blogs.technet.com/b/mmpc/archive/2011/03/18/operation-b107-rustock-botnet-takedown.aspx>
- [47] Winsrvprf, "Interpreting CPU Utilization for Performance Analysis", Windows Server Performance Team Blog, Aug. 6, 2009, Retrieved from <http://blogs.technet.com/b/winserverperformance/>
- [48] Yegulalp S. Review: Six Rootkit Detectors Protect Your System. In: *Information Week*.