Global Information Assurance Certification Paper

Stealth for Survival:
Threat of the Unknown

Author: Ken Dunham, ken@kendunham.org

GCIH Gold Certification

Adviser: Dominicus Adriyanto

Accepted: April 4, 2007

Table of Contents

1. Executive Overview

A dramatic increase in Windows "rootkits" and stealth components emerged by December 2005. A perceived "drop" in malicious code in 2006 is, in part, a reflection of increased stealth and survival of undetected malicious codes in the wild (Doyle, Dunham, 2006). Now, more than ever, both consumers and corporate users need to understand how to identify and remove stealth codes from Windows, including rootkits, browser help objects, and codes that utilize alternate data streams to conceal data on a computer.

The term rootkit originally stemmed from a collection of Unix commands that were modified to conceal an attack (Wikipedia.org, 2007). Today the term is generalized to include any type of kit or component that results in modification of software to control the behavior of software for concealment purposes. There are two fundamental types of rootkits, user-level and kernel-level. User-level rootkits work on the application level and kernel-level work on the kernel level, the core of the operating system itself.

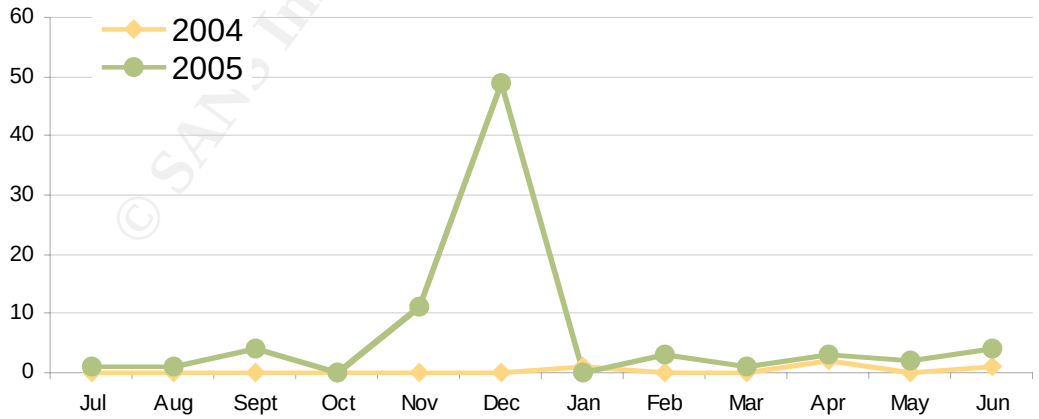


Fig 1: iDefense: Major spike in Windows rootkit components in December 2005.

Popularization of rootkits through several public websites had greatly increased the threat by 2005. iDefense identified a significant spike in rootkit functionality for Windows in December 2005 (Dunham, K. & Doyle, F., 2006). This spike largely represents the use of rootkit techniques and scripts within e-mail worms like that of Feebs and automated bots.

Hackers have concurrently adopted similar stealth components including hostile browser help objects and malicious use of alternate data streams (ADS). These new stealth vectors, now popularized within multiple malicious codes in the wild, greatly increase the complexity and time requirements for staff addressing stealth threats in the wild. Common anti-virus applications do not detect such programs easily, requiring an arsenal of techniques and specialized software to properly root out rootkits and stealth code. Worse, many infections now go undetected for extended periods, maximizing criminal exploitation of compromised resources.

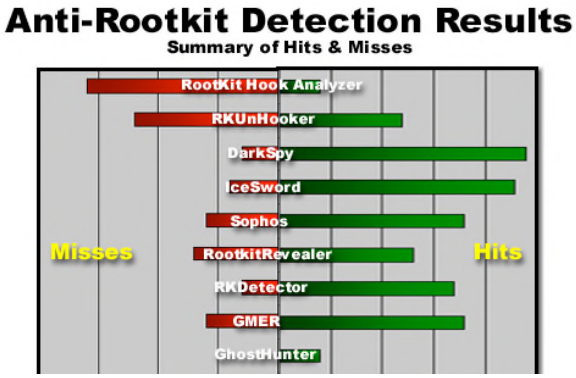


Fig 2: Anti-Rootkit Detection Results

Anti-rootkit programs are a focus of the lab study performed in this research. DarkSpy and IceSword proved to be the most effective anti-rootkit programs available as freeware today (more "hits" or detections of rootkit components). Traditional anti-rootkit scanning programs, like BlackLight, and newer products such as the Sophos Anti-Rootkit program, performed well in lab tests. See a rootkit video at http://www.kendunham.org/rootkit.wmv.

2. Introduction to Stealth and Windows "Rootkits"

The term "stealth" refers to something that is surreptitious or secret. On the Windows operating system there are a wide variety of malicious code techniques utilized for "stealth" to conceal a malicious code infection. Most recently Windows "rootkits", browser help objects (BHOs), and alternate data streams (ADS) are leveraged by hackers to conceal the infection of malicious code on a computer.

The focus of this paper is on Windows rootkits, the most complex of the three stealth components aforementioned. Definitions for BHOs and ADS are below, with additional data on each following the section on Windows rootkits.

2. 1 Browser Help Object (BHO)

A BHO is a plug-in or extension to Internet Explorer 4.X and later. Other browsers have similar features, such as plugins for FireFox, with Internet Explorer calling plugins "browser helper objects" (wikipedia.org, 2007). Each BHO is a Component Object Model (executable) object inside of a Dynamic Link Library (DLL) that is loaded with a new instance of Internet Explorer. For example, the Adobe Acrobat Reader BHO is installed by Acrobat Reader as acroIEhelper.dll. The Google Toolbar is another popular example of a legitimate BHO:

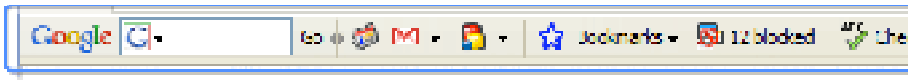


Fig 3: Google Toolbar BHO for Internet Explorer

Malicious BHOs are ideal for information-stealing malicious code. They don't need to run in memory at all times, just when the Internet browser is launched. This normally occurs when the user is online with Internet access. Once online, information-stealing malicious codes commonly upload and download data of choice, steal sensitive information, and further exploit a compromised computer. An excellent example of a BHO installation by malicious code is the MetaFisher family, explained in detail later in this report.

Windows XP SP2 and later supports management of Internet Explorer add-ons, browser help objects (Microsoft Corp., 2004). Select "Manage Add-ons…" from the Tools menu to view any BHOs installed on the computer. An example screenshot of a computer with many legitimate add-ons is below:

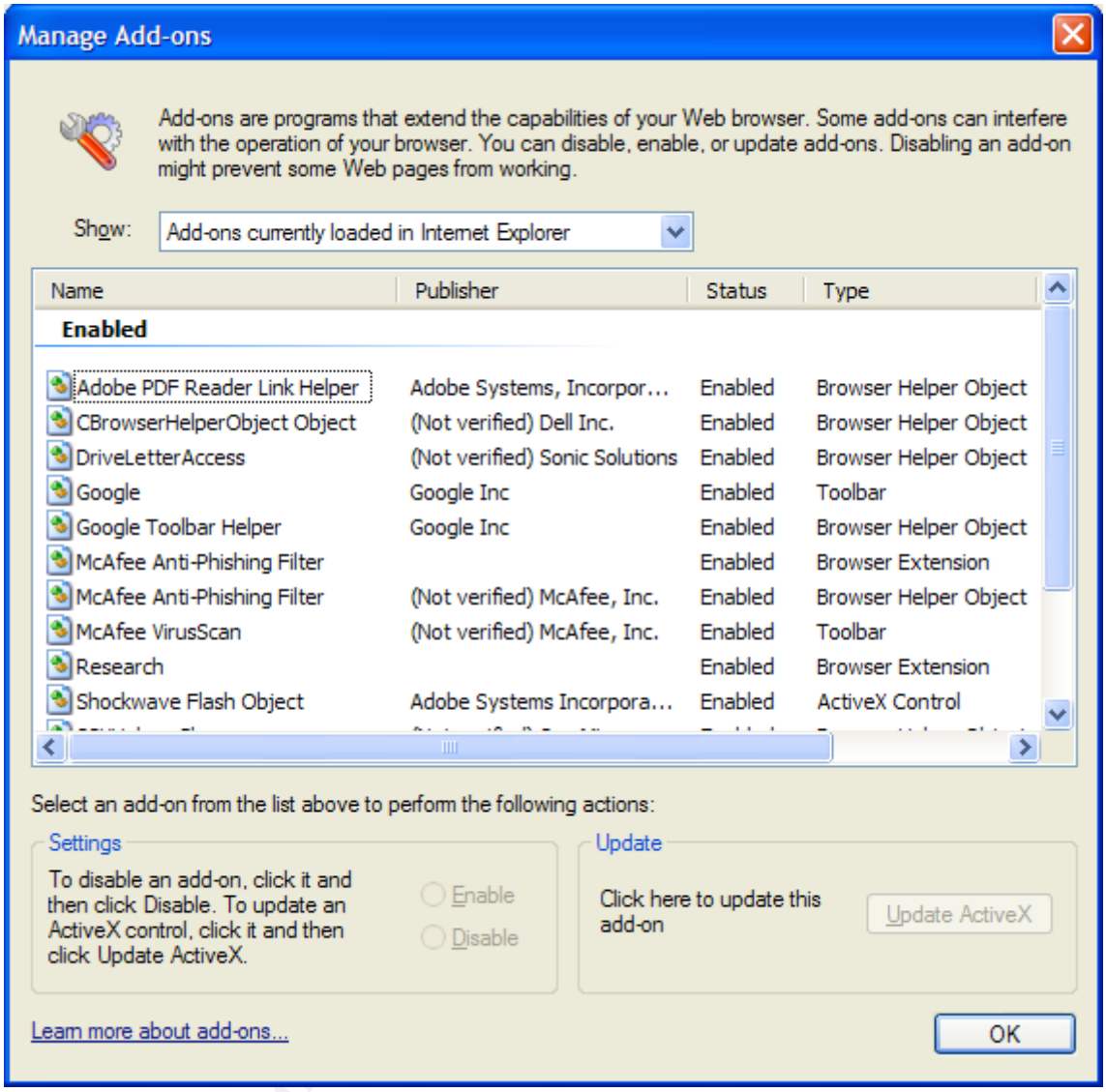


Fig 4: Internet Explorer with legitimate BHOs installed.

A multitude of utilities now exist to search a computer for rogue BHOs, which may best meet the needs of individually configured environments. Popular tools include but are not limited to HijackThis, BHOZapper, BHODemon, and BhoScanner. Again, newer versions of Internet Explorer also natively display a list of BHOs under the "Manage Add-ons…" from the tools menu.

2.2 Alternate Data Stream (ADS)

An extension to Windows NT File System (NTFS), introduced by Microsoft Corp. in 1993, that provides compatibility with files created using Apple's Hierarchical File System (HFS) and provides programmers with alternate data storage. ADS are like a secret location in which to hide data associated with files and folders.

Hackers have been discussing and documenting ADS for several years, including the well-known 29A virus coding group. Benny and Ratter of 29A created the Win2k.Stream in September 2000, popularizing the concept of ADS abuse (Benny, Ratter, 2000). In 2001 Potok, a new worm utilizing ADS, emerged in the wild (McAfee, 2001). Since that time other groups have also shown an interest in ADS exploitation, especially as it relates to the hiding of Trojan files and other data.

In September of 2003 several variants of the CoreFlood Trojan horse family were discovered in the wild (Sophos, 2003). These backdoor Trojan horse programs create a DLL in an ADS file associated with the Windows System directory in an attempt to conceal the Trojan. A top-ten worm for prevalence also emerged around this same time, the Dumaru family. Dumaru worms create a copy of the original executable in an ADS called STR. They then create a copy of the worm on the local file system and hook it into ADS for concealment (Symantec, 2003). Today, Mailbot codes abuse ADS for concealment purposes as the saga continues (F-Secure Corp., 2006).

Anti-virus companies are increasingly moving toward support for ADS scanning but often don't support ADS

scanning by default. Scanning of ADS can decrease system performance, but that is becoming less of an issue as hardware and software improves over time. Below is an example of Kaspersky Anti-virus successfully identifying a Trojan stored within an ADS:

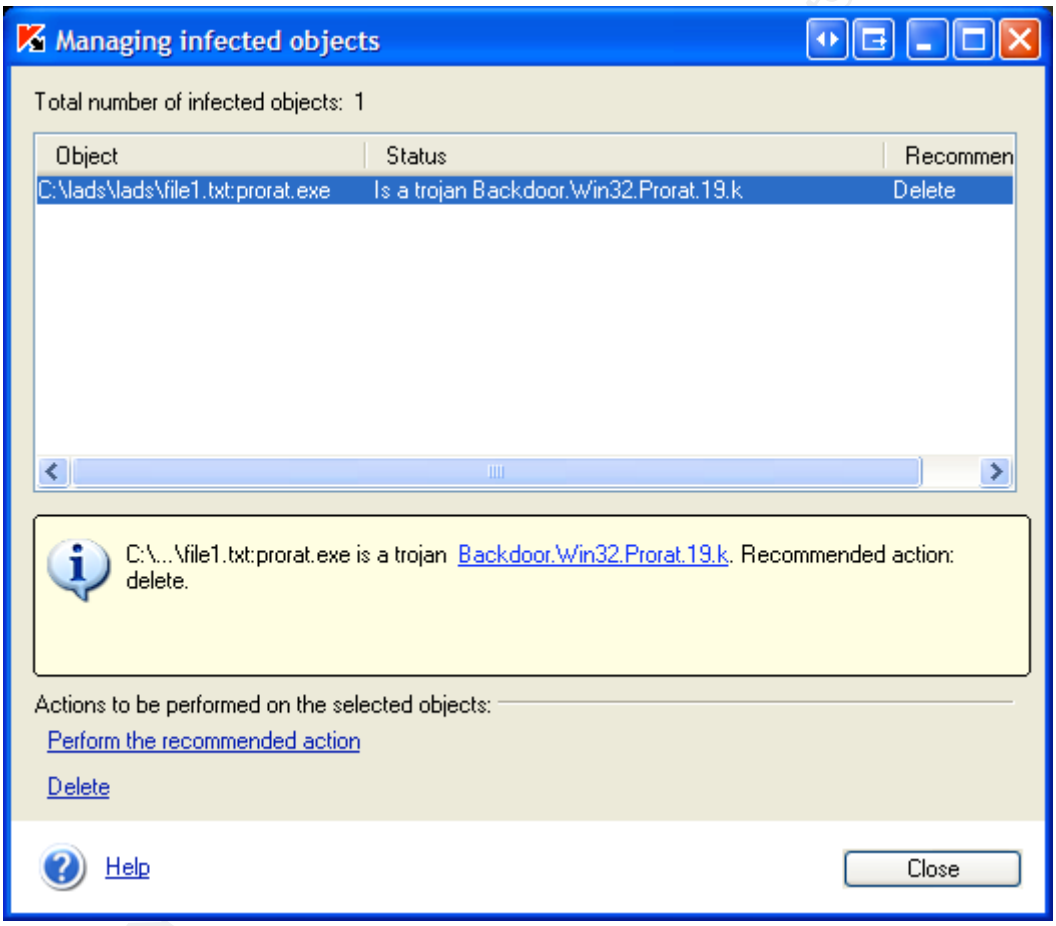


Fig 5: Kaspersky detects the Prorat Trojan hidden in ADS.

Specialized but much less employed programs such as TDS-3 have scanned ADS since 1997. As a result, current ADS-related malicious code threats may go undetected for extended periods until ADS detection software is widely distributed and implemented. In many cases administrators simply give up trying to find questionable applications

that are responsible for identified egress traffic of concern and simply re-image a compromised computer – never identifying the actual threat stored in ADS.

Below is a simple illustration of how an attacker may create an alternate data stream to conceal data. The "readme.txt" visible to the normal "dir" command or Windows shows what appears to be a legitimate readme.txt file. However, looking closer with tools like LADS (Heyne, 2007) we see that an alternate data stream exists recording the credentials of the administrator.

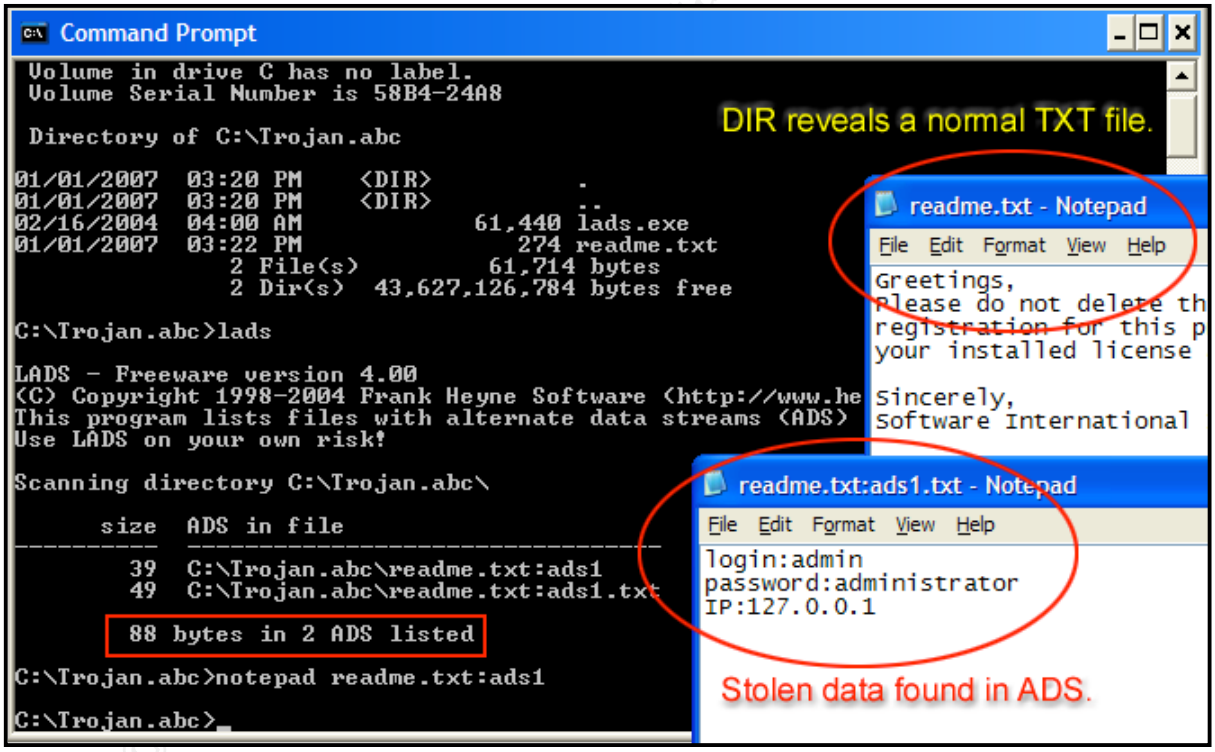


Fig 6: Alternate data Streams conceal data on a drive.

For a quick hands-on exercise (Dunham, 2004) to create an alternate data stream on an NTFS drive simply do the following:

1. Create a folder named "ADS" on your C:\ drive.

2. Download a free copy of LADS, an ADS detection utility, from http://www.heysoft.de and extract it into the C:\ADS directory.
3. Select "Run…" from the Start menu, type in "cmd" (without quotes) and press Enter.
4. Type "CD C:\ADS" and press enter to navigate to the ADS directory.
5. Type "DIR" to view the contents of the directory.
6. Type "echo "hidden data goes here" > adstest.txt:ads1" and press enter to create both a file and hidden data inside of ADS.
7. Type "lads" and press enter to view the newly created ADS.
8. Delete the ADS directory when done to delete test files and the ADS created with this test.

To store an executable within an ADS simply associate it with a file or directory, or create one, such as:

```
Type notepad.exe > readme.txt:notepad.exe
```

If you're not auditing for alternate data streams you may not find data stored there by an insider or malicious code attack. Many tools now exist to find and delete or manage data within ADS today. Additional information on ADS mitigation comes later in this report.

3. What is a Rootkit?

A rootkit is a set of programs used to manipulate software to conceal its presence on a computer. SANS defines (SANS, 2007) a rootkit as follows:

> "A collection of tools (programs) that a hacker uses to mask intrusion and obtain administrator-level access to a computer or computer network."
>
> http://www.sans.org/resources/glossary.php

Notice that the SANS definition adds an additional level of intent or functionality, using a rootkit to obtain administrator-level access. The concept of "administrator-level" access is in this definition to accommodate the historical function of rootkits, to gain and maintain "root" on a system. While accurate in the traditional sense, this is not always the case today, where malicious code already obtains "root" and then uses a stealth or "rootkit" component to conceal the infection. The anti-virus industry has been plagued with categorization of evolving malicious code for years, where traditional definitions no longer accurately describe cutting edge attack techniques and/or technology.

Historically, the traditional rootkit is composed of a variety of Unix commands, such as passwd, ls, ps, netstat, and others that are modified and recompiled to conceal specific ports, accounts, or other information. This helps to reveal the presence of hostile code or an intrusion on a computer. In this way a suite of tools, or "**kit**", enables the attacker to maintain **root** on a compromised computer.

The general public today generalizes usage of the term "rootkit" to denote code used to conceal an attack by

controlling user views of various registries, processes, directory lists, and more. No longer do "rootkits" have to be a collection of tools used for stealth. This is especially true on the Windows operating system, where only one component of a malicious code attack may be used to conceal a process or perform stealth functions of choice.

The key factor to embrace today is that Windows "rootkits" are all about stealth, accomplished through one or more tools or components attempting to manipulate the user and the operating system.

3.1 Introduction to Windows Ring Permissions

There are two fundamental types of Windows rootkits: **user-level** and **kernel-level** rootkits (Dunham, K. & Doyle, F., 2006). User-level rootkits, aka userland rootkits, operate on what is known as "ring 3", while kernel-level rootkits operate on what is known as "ring 0".

Intel x86 hardware employs a ring architecture to manage access control. Four rings are used in the ring system, with ring 0 being the lowest level of all rings where the kernel resides (Answers.com, 2007). Userland rootkits may operate in rings three (userland rootkit) or ring zero (kernel rootkit). Rings one and two are not utilized on the Intel x86 architecture. Therefore, only rings 0 and three matter on a Windows operating system, with ring zero access the target of an attacker.

MS-DOS is unable to support privileged levels, effectively running at ring 0 (all or nothing OS).

Rootkit authors know that drivers, such as a print driver, require temporary access to ring zero to install a driver (Hougland, G, & Butler, J, 2006). Rootkit actors use this to

their advantage to manipulate memory management for installation of stealth code:

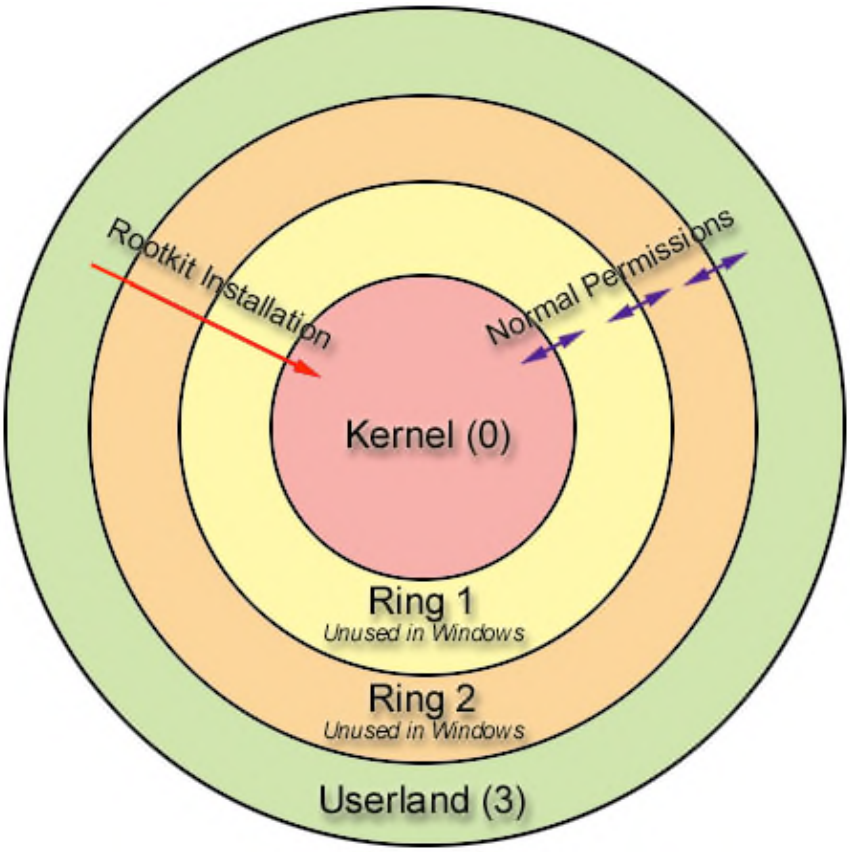


Fig 7: A Windows kernel rootkit installs self on ring 0 from ring 3.

The central processing unit (CPU) hardware attempts to find addresses via multiple tables. Many rootkits attempt to manipulate data within various tables in attempt to install and/or manipulate the operating system. The following is a list of common tables related to CPU processing and rootkit abuse (Hougland, G, & Butler, J, 2006):

- Global Descriptor Table (GDT) to map addresses
- Local Descriptor Table (LDT) to map addresses
- Page Directory to map addresses

- Interrupt Descriptor Table (IDT) for finding interrupt handlers

The Windows operating system (OS) also uses a table for handling system calls, the System Service Dispatch Table (SSDT).

Each process under Windows may have a separate lookup table to find memory pages for memory. Different lookup tables can cause memory to be viewed entirely by each process. Each process has a unique and independent view of memory.

When instructions are processed the CPU first performs a descriptor check, then a page directory check, and then a page check before a page is marked as read-only by the CPU. The OS doesn't use a descriptor check but relies upon the ring structure for management of access controls. The User/Supervisor bit in the page table check is used alone to control access to memory, and thus the kernel by ring zero software. Translation of memory addresses is handled by the page-table directory.

3.2 User (Application)-Level Rootkits

The key word here is "application," where Windows user-level rootkits work on the application level. Windows user-level rootkits are **commonly installed as a DLL file** or an injected thread into an existing process (Dunham, K. & Doyle, F., 2006). Malicious code commonly attempts to register the hostile DLL file as legitimate to then enable it to control various function calls made by Windows or other programs.

User-level rootkits, commonly referred to as userland rootkits, are often performed through simple process injection

and hooking application program interfaces (APIs) (Hougland, G, & Butler, J, 2006).

Programmers utilize the Windows Application Programming Interface (WinAPI) to code a variety of functions within Windows. Most of these functions reside within three Dynamic Link Library (DLL) files: gdi32.dll, kernel32.dll, and user32.dll. These three DLL files contain various functions and commands used to control graphical commands, file and memory access, and application control, respectively.

Rootkits attempt to control function calls made to the three aforementioned DLL files. For example, a user-level rootkit may install itself, register a hostile DLL, and then attempt to intercept and manipulate or "hook" a series of WinAPI calls. To control any software, including Windows, from reporting back to the user a directory listing of files the rootkit may attempt to hook WinAPI calls related to FindFirstFile, FindNextFile, and others.

3.3 Kernel Level Rootkits

The kernel is the very core of an operating system, responsible for process management, file access, security, and memory management.

Windows Kernel-level rootkits have the same basic functionality of user-level rootkits but accomplish the concealment of the code through the kernel rather than on the application level. **Look for SYS or VXD files** and rootkit functionality when identifying kernel-level rootkits (as opposed to just DLL files with user-level rootkits).

In general, kernel-level rootkits are more robust and difficult to detect than user-level rootkits. Kernel-level

rootkits function on a deeper level, the kernel of the operating system.

3.4 Common Windows Rootkit Families

There are many families of Windows rootkits in the wild today. Below is a list of rootkit families for 2006, where the first variant of each family discovered in the wild with rootkit functionality is listed below. Thus, any variant after that date, such as variants of Feebs following Feebs.T, likely contain rootkit functionality. Earlier variants may NOT contain rootkit functionality, dependent upon the family. While this list is fairly comprehensive, it is a work in progress due to the evolving nature of stealth code discovery in the wild to date.

AFXrootKit.A, Agent.AAV, AluRoot.A, Arkdoor.A, Bagle.HG, Berbew.F, Blubber.A, Bomka.I, Chsh, Comxt.A, Darkmoon.A, Dasher.B, Denutaro.A, Detnat.A, Dloader.AJE, Dropper.AR, Embed.C, FakeMcPatch, FBSD, Feebs.T, Feutel.BB, Fkit.A, FreeBSD.A, FuRootkit, Gdiw.A, GiftCom.A, Ginwui.A, Goldun.W, GrayBird.AE, HacDef.084, HackDef.B, Haxdoor.G, Hupigon.P, Inrg.A, IRCFlood.G, Isen, Kalshi, KassBot.K, Kelar.B, Knark, Lecna.E, LinkOptimizer.A, Login.A, Lrk5.A, MadTol.B, Mailbot.E, MarktMan.A, MDropper.G, Mitglied.A, Mose, Mytob.NC, NtRootK, NtRootKit.D, Opanki.Az, OrderGun.A, Padmin.10, PcClinet.AP, PPdoor.F, PrintMonitor.A, ProAgent.20, PsGuard, PWSteal.C, QoolAid.C, QukArt.U, Raleka, RBot.AWG, Ripgof.A, RKFu.A, Rkit.A, RkProc.A, RootKit.H, RusDrp.A, Rustock.A, RyeJet.A, SdBot.VD, Small.IZ, Spybot.TR, Suckit.A, Sun2, Sunback.A,

SunOS.A, TdiRoot.A, Theales.A, TileBot.AK, Totmau.A, Uprootkit.A, XCPDRM.A, Xsvix.A

3.5 Selected Rootkit Techniques

3.5.1 Browser Help Objects

Stealth codes can easily install as a BHO. However, BHO detection programs easily identify BHOs. Additionally, recent versions of Internet Explorer make it easier for users to identify installed BHOs, increasing transparency of installed plugins including possible malicious code instances. BHOs are unable to auto-start without execution of the browser. This is actually advantageous for the attacker who only wants to run code in memory while online, for uploading of stolen data or connecting back to a compromised computer.

3.5.2 Rootkit Driver Installation

A fairly common method used for Windows rootkit installation today is through userland rootkits installed as a driver. To survive a reboot a new Windows registry key may be entered under the standard HKLM/…/RUN key. Once a rootkit is installed it can be configured to conceal this auto-start hook in the Windows registry when a user attempts to investigate the registry.

3.5.3 Process Injection

Injection is another common method for userland rootkits, injecting a malicious process into an existing process, executable path or existing file such as a DLL (Kuster, 2003).

For example, *CreateRemoteThread()* (Shelton, 2006) may be used to add a thread into the memory space of an existing process on a computer.

Explorer.exe, the Windows explorer process always running under Windows, is a common injection target. Injection is only used for legitimate processes (e.g. explorer.exe and others).

If a system-wide resource, such as taskman.exe or explorer.exe, is injected the rootkit gains global visibility and increased privileges for restricted function calls. The rootkit NTIllusion (SecuriTeam, 2005) makes use of these principles to hijack Windows XP privileges from a non-administrative account. As the rootkit increases its privilege base, the same hooking functions allow it to hook into increasingly more valuable targets including system API calls.

Hooking calls enables the rootkit to return only values that will not disclose its presence. Thus, through a series of function calls and browsing the internal Windows process tables and data structures, even a user-mode rootkit can achieve full administrative access while maintaining complete stealth.

Windows attempts to prevent code injection by checking for applications without permission attempting to perform read or write functions in memory. *SetWindowsHookEx()* (Cummings, 2006) function is one method leveraged by hackers to inject code. This technique attempts to hook the message event handler of an application. When Windows maps an associated DLL with an application, the DLL is hooked into the memory space of the application. In this way a malicious DLL file can be hooked into an existing application's memory space, thereby injecting the code and not running as an independent process on the operating system.

3.5.4 API Hooking

Once code injection has occurred API hooking (Microsoft Corp., 2007) is performed to manipulate behavior on a compromised computer (Volker, 2001). Hackers typically attempt to patch the Import Address Table (IAT) (Hougland, G, & Butler, J, 2006) to point to the hostile code instead of the true entry point of the target API.

A global hook is sometimes employed to ensure that a rootkit remains completely hidden at all times from all graphical processes within Windows. This may be accomplished by using the SetWindowsHookEx for WH_CBT events. This is limited to processes using user32.dll so it won't work for console programs such as cmd, netstat, and others. As a result the rootkit must also be able to hook into processes to avoid detection via console programs. This is achieved by hooking into the CreateProcessW function into all injected processes. This trick also enables the rootkit to inject into Task Manager when ALT-CTRL-DEL is pressed.

3.5.5 New Interrupt Descriptor Table (IDT)

IDT is inside the INIT section of ntoskrnl. Rootkits may also create a new interrupt table to hide modifications to the original Interrupt Descriptor Table (IDT). Once a rootkit is installed API hooking may be leveraged to further control the computer. API hooks are excellent for hiding files from a user, such as concealing a directory containing hostile files and stolen log data. While API hooks are not that difficult to detect they are easy to leverage and commonly implemented within code in the wild today.

3.5.6 Direct Kernel Object Manipulation (DKOM)

A kernel-level rootkit exploits the Loadable Kernel Module (LKM) (Hougland, G, & Butler, J, 2006) to obtain direct access to hardware. This causes the DLL to obtain access to ring 0, the kernel. A kernel-level rootkit obtains ring 0 access by installing itself as a device driver requiring Direct Kernel Object Manipulation (DKOM). This technique is great for hiding processes, hiding device drivers, hiding ports, elevate privileges, and skewing forensics. Mainstream media, in both hardcopy print and Internet web page forums, has helped to popularize this technique.

3.5.7 CR0 Trick

The "CR0 trick" (James, 2007), as documented, is used to disable read-only settings of the System Service Descriptor Table (SSDT) and Interrupt Descriptor Table (IDT) tables. This makes it possible for a rootkit to install (write permissions required) in an area that would otherwise be read-only. The following Windows registry keys are modified when the CR0 trick is utilized:

HKLM\System\CurrentControlSet\Control\Control\Sessions Manager\Memory Management\EnforceWriteProtection = 0

HKLM\System\CurrentControlSet\Control\Control\Sessions Manager\Memory Management\DisablePagingExecutive = 1

3.5.8 Sysenter eip

Mailbot (Rustock) is the first code in the wild to utilize the Sysenter EIP technique (Symantec, 2006). According to an online tutorial, Sysenter EIP utilizes the following strategy (sic):

> "I use ZwQueryInformationProcess to retrieve a pointer to PROCESS_BASIC_INFORMATION structure. Now, from here we obtain the PebBAseAddress field; from this we retrieve a pointer to a RTL_USER_PROCESS_PARAMETERS structure called ProcessParametes. From this structure is possible to extract the ImagePathName.Buffer field (the type is WCHAR)."

3.5.9 Directly Modify the Kernel

Modification may also be made to the on-disk kernel (Hougland, G, & Butler, J, 2006). A permanent modification to the kernel is highly advantageous to an attacker but extremely difficult compared to other attack techniques. No device drivers or auto-start hooks are required when the rootkit is installed into the kernel directly. Checksum integrity scans and similar counter-measures pose threats to this type of rootkit, but are not commonly utilized today since this type of rootkit is not popular in the wild. It is also possible to modify the boot loader to patch the kernel before it loads, avoiding checksum challenges.

3.5.10 Persistent System BIOS Rootkits

John Heasman has proven that persistent BIOS rootkits are possible in his paper released November 2006 (Heasman, 2006). This technique involves modification of Advanced Configuration

and Power Interface (ACPI) tables within the BIOS to enable a rootkit to overwrite kernel code and data structures.

3.6 Availability of Rootkits Today

Rootkits are traditionally a collection of tools used to modify data presented to the user of a computer. Today rootkit components on Windows are the latest trend, where malicious code includes a "rootkit"-like component as part of the malicious code installation. The graph below identifies rootkits reported by iDefense for 2005. The number of new rootkits remained low throughout 2005, with a significant spike in November and December 2005.

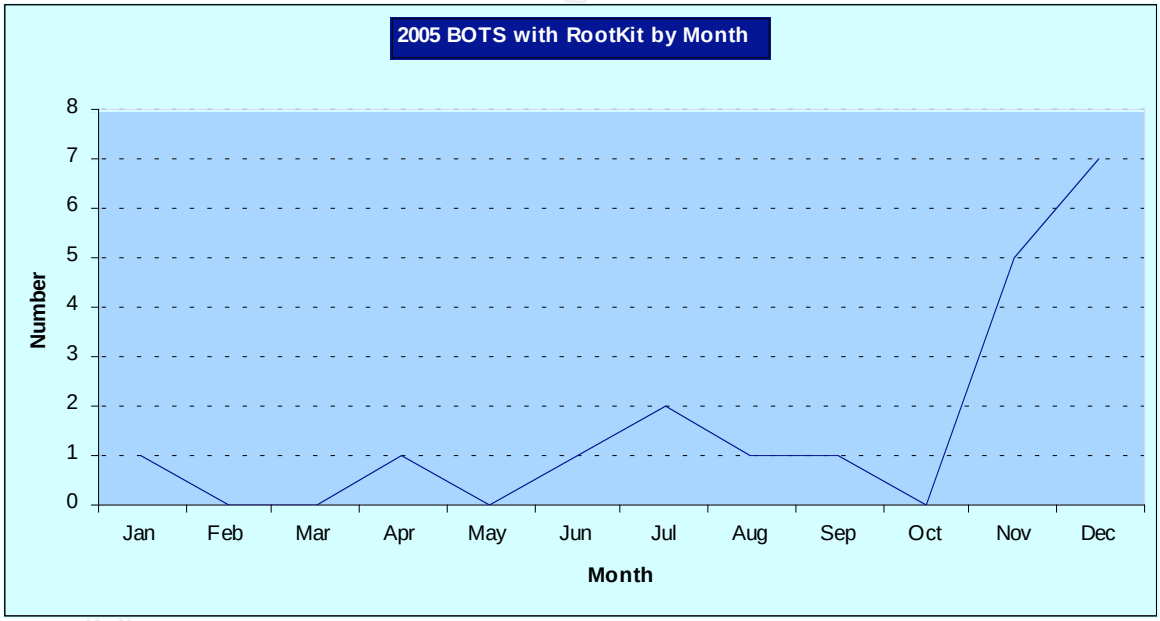


Fig 8: VeriSign/iDefense 2005

At the same time rootkit components shot through the roof with worms like Feebs, in December 2005 (Dunham, 2006). The graph below shows a significant increase in "rootkit or rootkit components" in December 2005.

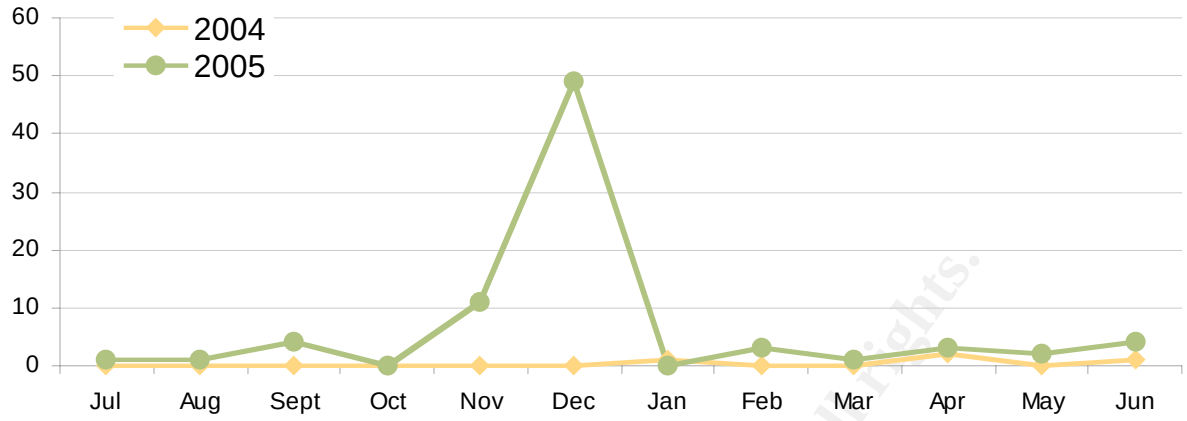


Fig 9: VeriSign/iDefense 2005

The number of codes containing rootkit components tripled in the first quarter of 2006 compared to the first quarter of 2005. While this number is still relatively low overall, it shows a significant increase in codes in the wild utilizing rootkit components. To complicate matters, rootkit components are increasingly difficult and unlikely to be detected, making it highly likely that this threefold increase in detected rootkits in early 2006 is actually far less than actual prevalence of rootkit components in the wild at that time.

4. Descriptions of Selected Stealth Codes

Descriptions of selected stealth codes are below to familiarize administrators with selected threats facing networks at the time of this writing. These threats are generally representative of rootkit and BHO/ADS threats in the wild today.

Dozen new families of code have emerged in 2005 and 2006 as containing rootkit functionality for the first time, such as variants of Bagle, HackDef, GinWui, Mailbot, RBot, Myfip, Opanki, Haxdoor and many others. This is exacerbated by those that spread through targeted attacks involving zero-day exploits, like that of GinWui, or rapid exploitation of new vulnerabilities, like that of Opanki exploiting MS06-040 in the summer of 2006.

4.1 FURootKit

The FURootKit is an excellent example of a kernel-rootkit that uses DKOM techniques. FURootKit was developed by Jamie Butler (a.k.a. Fuzen), as a proof-of-concept for DKOM as part of his research for the book, "*Rootkits: Subverting the Windows Kernel".*

The name "FU" rootkit is a play on words. In UNIX, "SU" is used to "switch users" to elevate privileges. FU sounds similar but also has a negative connotation that can't be missed by an American audience.

Rootkits.com advertises FURootKit as a code that can:

> "*…hide processes, elevate process privileges, fake out the Windows Event Viewer so that forensics is impossible, and even hide device drivers (NEW!). (Look, Mom, no hands!) It*

does all this by Direct Kernel Object Manipulation (TM); no hooking!"

FURootKit is a collection of source code that hackers use to create customized kernel-level rootkits. All of the files and libraries included with FURootKit are very well documented, facilitating use of such code in the underground as well as for research. As a result, FURootKit code is commonly found within bots.

FURootKit consists of two primary files: an executable and a SYS file. The SYS file is a primary clue that a kernel-rootkit may exist.

4.2 Feebs

Feebs is a new e-mail worm family that first appeared in the wild on Dec. 20, 2005, installing itself as a user-level rootkit (Dunham, 2006). Just nine weeks later 30 variants existed in the wild. By the spring of 2006 a new Feebs variant emerged every two days or less! This is a high degree of multi-variant prevalence compared to most threats to date. While it's impossible to track all minor variants of this family to date, naming systems and correlation to date of signatures indicates several hundred variants within the first six months following initial discovery!

Feebs is nothing special on the front cover, until you look closer. It's not a common mass mailer, but one that injects copies of itself into legitimate e-mails sent out on an infected computer. It is a heavily encoded polymorphic worm with many features including rootkit functionality. Due to the heavy encoding, most reports on Feebs worms to date have been sparse

in reporting details of this worm family. No single in-depth report exists to identify every single feature of Feebs.

As a general introduction to the Feebs worm family, Feebs worms do the following:

- Randomized HTML e-mail worm that spreads via HTA and ZIP extensions.
- Creates %System%\MS[random characters].exe (53,259 bytes).
- Creates %System%\MS[random characters]32.dll.
- Modifies the Windows registry to auto-start upon Windows startup, to conceal the worm, to disable security, and change the start page of Internet Explorer.
- May delete various files as well as create others used in the spreading of the worm.
- Create copies of itself in directories with the string values of "download, upload, income, or share" directories as a ZIP file containing an executable called websetup.exe.
- Terminates services related to security services and disables settings for Windows firewall.

An aggregate analysis of the family code has resulted in the identification of the following features highlighted in this report:

- Heavily encrypted and weak polymorphic code
- E-mail worm, P2P worm, downloader, and backdoor Trojan horse
- Port 80 Web Server

- Steals information and monitors traffic to targeted financial URLs
- Start page change
- Rootkit functionality

Feebs worms are able to hide files, Windows registry keys, processes, and network connections by injecting a hostile DLL component into all running processes via the svchost.dll file. It also attempts to hooks the system library functions as part of playing the man in the middle of all such activities to conceal the infection. It may also attempt to conceal P2P copies of the worm in explorer window containing the text "_new!_full+crack.zip". It also attempts to conceal the userinit.exe download in the Recycled directory, similar to the technique first popularized by the SirCam worm several years ago.

4.3 MetaFisher

MetaFisher is one of the most sophisticated bots in the history of computing (Dunham, 2006). MetaFisher installs itself as a BHO on a compromised computer. Attackers deploy MetaFisher in a variety of ways, including a targeted attack that took place in the spring of 2005 illustrated below:

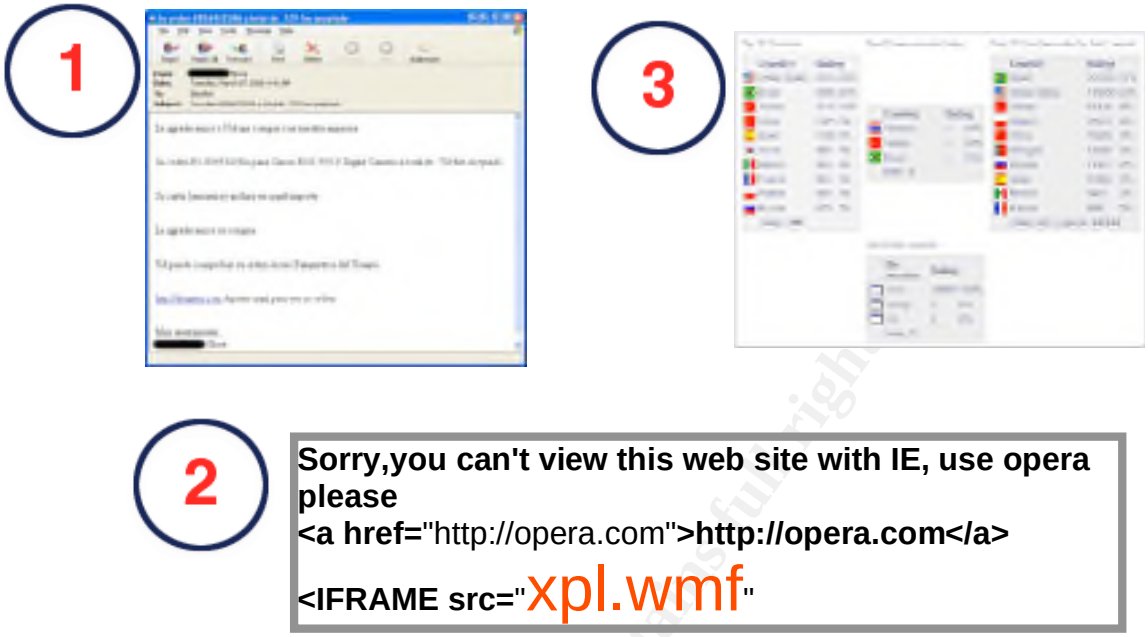


Fig 10: VeriSign/iDefense: Stages of MetaFisher Attack

For this specific attack, targeted e-mails are sent to consumers of targeted Spanish banks. E-mails use social engineering to prompt the user to visit a website that contains a windows Metafile (WMF) exploit as highlighted in orange in step 2 above. If a vulnerable version of Internet Explorer is used to browse the website, the consumers' computer is silently infected with a MetaFisher browser help object code.

Step three is the view of the attacker who authenticates to a PHP command and control center to view infections by country and update bots and steal information as desired. A configuration page, utilized by hackers, is below:

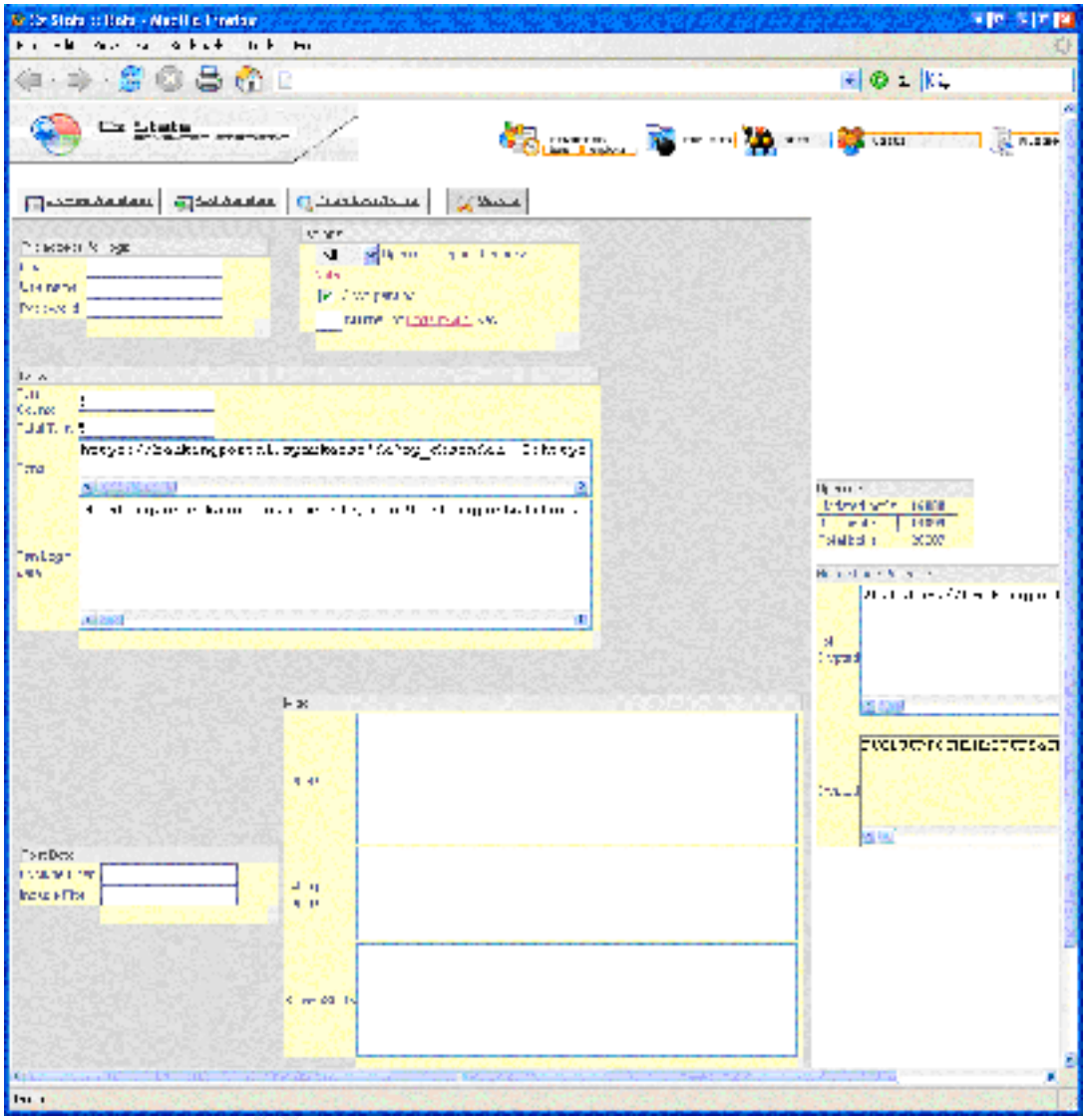


Fig 11: VeriSign/iDefense: MetaFisher Configuration Page

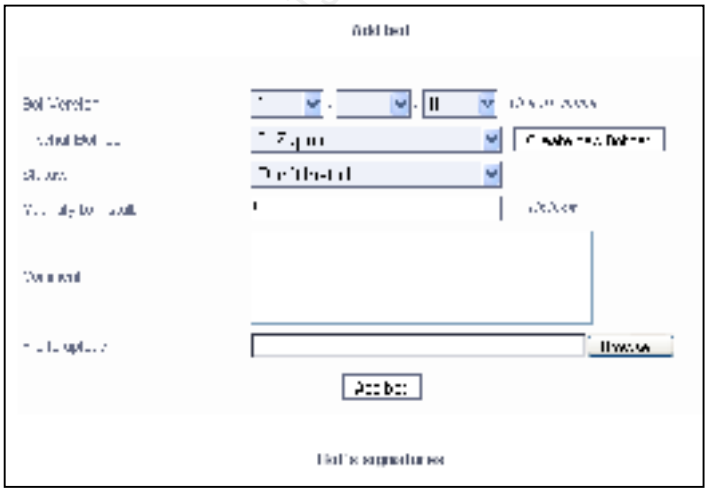


Fig 12: Bots are easily updated through website.

Updates to the bot, including targeted transaction authorization number (TAN) attacks, are easily performed through this command-and-control, PHP-based website.

The hostile BHO component of MetaFisher attempts to download commands from the command-and-control center via TCP port 80-encrypted communications disguised as a ZIP file. This makes it possible for MetaFisher to successfully evade standard intrusion detection efforts, flags that may be set for standard bots, egress and ingress filtering, and the ability to control hundreds of thousands of bots easily through just one command and control site.

MetaFisher also performs state-of-the-art phishing attacks by generating new fields for Spanish banks, stealing TAN numbers from German bank consumers, and performing local phishing attacks against United Kingdom banks. Local man-in-the-middle attacks are based upon the environment, within a fully authenticated secure sockets layer (SSL) transaction, completely undermining trusted security measures to date.

4.4 Rustock.B

Rustock is a well-known rootkit popularized to the public in part by publications from Symantec (Symantec Corp., 2006). It's special amongst rootkits because it has NO PROCESS. It may also utilize ADS to conceal data. It avoids hooks of native APIs, a common technique amongst Windows rootkits today. Rustock also controls special IRP functions to avoid SSDT detection by anti-rootkit programs. Rustock is also polymorphic and has anti-rootkit software features to prevent BlackLight, RootkitRevealer, RKDetector, GMER, Endoscope, DarkSpy, and Anti-Rookit from detecting it. In other words, Rustock is a major upgrade for Windows rootkits. Symantec theorizes that it originates out of Russia.

When executed, Rustock.B installs lzx32.sys into the Windows System32 directory. It also attempts to modify the

Windows registry and then hooks the MSR_SYSENTER code to patch several areas of the Windows Kernel to change functioning of several APIs:

- ZwOpenKey
- ZwEnumerateKey
- ZwQueryKey
- ZwCreateKey
- ZwSaveKey
- ZwDeviceIoControlFile
- ZwQuerySystemInformation
- ZwInitializeRegistry

4.5 Haxdoor (Nuclear Grabber)

Haxdoor is a Trojan that employs kernel-level rootkit techniques to conceal the processes, files, and changes to the Windows registry. This is a somewhat famous rootkit because it was used in the Swedish Nordea bank heist (Espiner, 2007). A customer Haxdoor attack launched against consumers of the bank leading to what McAfee calls the "biggest ever" online bank heist.

Haxdoor typically spreads via seeded e-mails (Sophos Plc., 2006). Once executed, it installs itself in the Windows System directory as dat, dll, and sys files. Specifically, five files are identified as rootkits for Haxdoor-DH:

- qo.dll
- qo.sys
- svjvpn.sys
- svjvpn.dll

- svkvpn.sys

It also modifies the Windows registry to run code exported by svkvpn.dll upon Windows startup:

- HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify\svkvpn DllName svkvpn.dll
- HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify\svkvpn Startup ER03Sb5fex
- HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify\svkvpn Impersonate

The rootkit file svjvpn.sys is registered as a new system device driver called svjvpn and display name of "MCRT accelerator", via the Windows registry key:

- HKLM\SYSTEM\CurrentControlSet\Services\svjvpn\

Once installed on a system, Haxdoor successfully hides all process, registry, and file data related to the protected rootkit files:

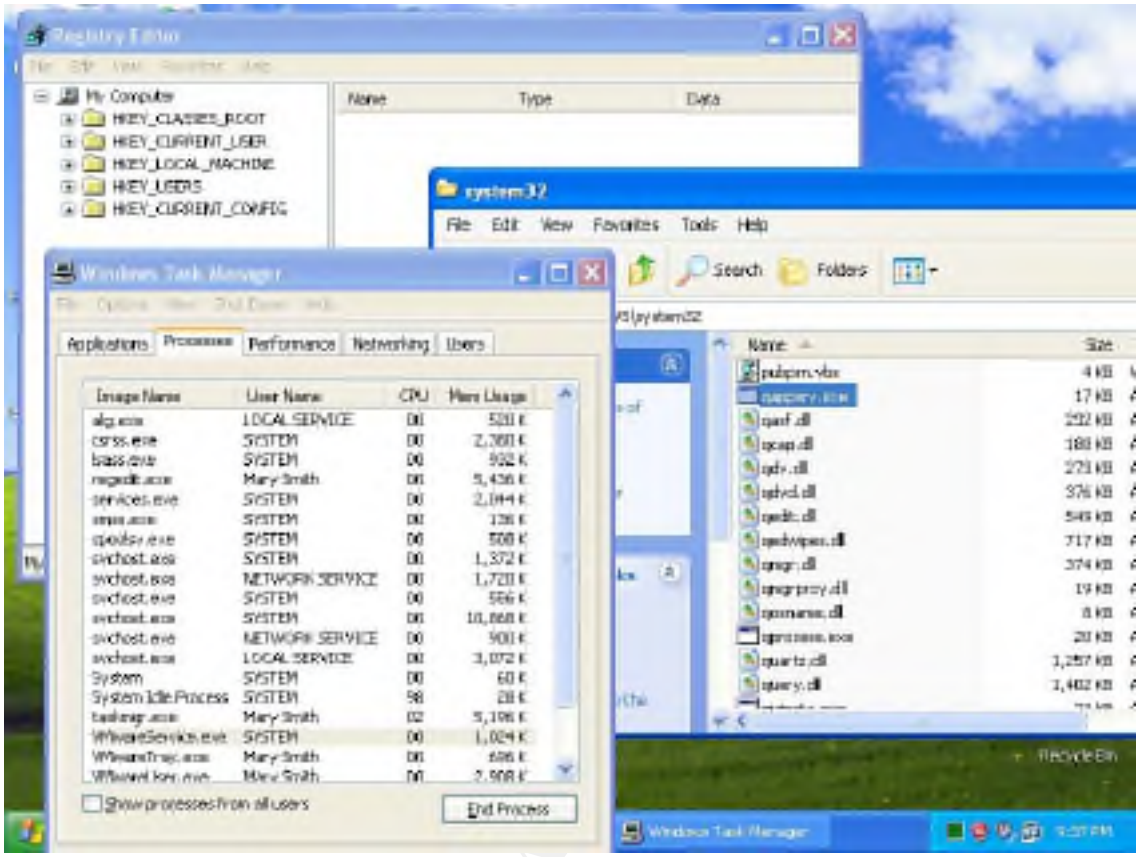


Fig 13: Haxdoor conceals presence of rootkit files on drive.

5. Anti-Rootkit Programs

A host of anti-rootkit programs have emerged in the past two years (see appendix for screenshots and links). My personal annotations for each program reviewed in this document are below, authored in the first person after analyzing rootkits on an infected system. See Appendix A for screenshots of each anti-rootkit tool identified in the table below.

Tools vary in terms of functionality and ease of use. Two basic categories exist: scanners and analysis tools. Scanners function in a variety of ways but are easy to use and produce a final report of discovered rootkit processes and/or files. Analysis tools produce lists that require more advanced user interpretation and analysis to locate rootkit components.

Some analysis tools color code possible rootkits in red and allow for sorting of data, while others are rudimentary with no such options. Of the reviewed analysis tools, IceSword is by far superior because it is more comprehensive, color codes items, and allows for sorting of data to quickly find newly created items and can identify ports utilized by a rootkit on an infected computer.

Compatibility issues exist between two or more anti-rootkit programs run at the same time. When run concurrently they may detect one another as a possible rootkit or may cause instability. Running IceSword and GMER together repeatedly crashed the VMWare test system.

5.1 BitDefender Rookit Uncover v1.0 Beta 2

Process | File | Cleaner

This is an easy-to-use scanner that is relatively fast. From a user standpoint it is very similar to F-Secure's Blacklight program. I prefer the user interface for the report because it immediately displays the full path of the hidden items to be renamed, important for recovery after restart.

5.2 Ghost Hunter Anti Rootkit 1.0.0.0

Process | Cleaner

This is a great program for just finding the hidden process on a computer. It highlights the hidden process in red and also allows for renaming and cleaning of the rootkit. It's not as robust as a tool like IceSword but is easy to use, has a nice GUI, and is fast.

5.3 GMER 1.0.12

Process | File | Registry | Cleaner

GMER is a program that took me by surprise. What it lacks in the GUI it makes up for in functionality. This highly

functional, easy to use scanner, does it all including scans for ADS and other areas of the computer. I especially appreciate how it displays more technical information about each rootkit component it discovers on the computer. The cleaner component of this program is not as obvious – just right-click on any item to restore, delete, or kill as appropriate.

5.4 RKDetector v2.0 Beta

File

This program is not very intuitive and is not designed for just rootkits. It does a good job of highlighting in red and performing a text dump of hidden files found on the system. It also has options for analyzing browser, recovery, ADS, and registry. To get started type "C:" or whatever path you like and click "Go". This program requires a larger screen mode and has issues with the GUI not scrolling or layering correctly at times.

5.5 Rootkit Revealer v1.7

Registry | File

This is one of the most well-known anti-rootkit programs on the market today. It's a pioneer of the field and it's free. It just reports registry and file information with no cleaning capabilities built into the program. It's great for a second opinion and a fast scan of a system. It's the first place I go when I want hostile registry data related to a rootkit.

5.6 Sophos Anti-Rootkit Version 1.2 (data 1.01)

Process | File | Registry | Cleaner

Sophos gives the user control to determine what they want to scan or not scan, something we don't see with other easy to use scanners for anti-rootkit. It's trivial to use, robust in

detection capabilities, and offers cleaning. It is also best at identifying the exact malicious code name in tests I performed.

5.7 Blacklight 2.2.1055.0

File | Process | Cleaner

Blacklight is easy to use and does a great job at finding and helping users clean rootkits from a system. It takes a bit longer than some scanners to run. Only icons on the "clean hidden items" screen clues the researcher into what was discovered, to the left of each found item. To see the full path of files being renamed you have to double-click on each, which is not as user-friendly for experts wanting to capture a sample after renaming. This program requires regular updating from the website, set to expire every few months.

5.8 IceSword 1.2.0

Process | File | Registry | Cleaner

This is a great program for the more advanced user wanting to investigate a possible rootkit infection through several possible angles. Instead of operating as a scanner, IceSword offers the ability to look for a variety of stealth components through process, registry, startup, BHO, and many more. I particularly like this program for two features, process and port identification of rootkits. The rootkit process is highlighted in red and easily identified. The port scanning capabilities are excellent for identifying port activity with a rootkit in real-time. While registry and file browsing are possible, it's like searching for a needle in a haystack. In the end look for REN files related to the rootkit.

5.9 Darkspy 1.0.5.0

Process | File | Cleaner

This program also includes a “supermode,” which requires installation and restart to implement. I used just the basic mode for all the tests I performed. It quickly highlights discovered rootkit processes in red and displays their PID and EPROCESS data. Simply left-click on the process to see the full path. Right-click to kill or force kill the process. You can also browse the file system and registry looking for REN files related to found rootkits. This powerful program also includes port monitoring capabilities too. The “cleaner” component of this program is related to list and discover and kill/rename capabilities performed by the more advanced user. Unfortunately, this program has no sorting or color coding options, making it much more difficult to use compared to IceSword, and it doesn't monitor ports in real-time (refresh required).

5.10 RKUnHooker 3.0.80.290

Process | File | Cleaning

This program requires an installation first, rather than simply running an executable like all others in this study. It is much more technical than most other programs, with tabs like “SSDT Hooks Detector/Restorer” and “Code Hooks Detector” and others. On the default tab for SSDT Hooks Detector/Restorer sort the list by Hooked to quickly see rootkits that may exist on the system. The same is true for similar columns in other tabbed areas, such as process status to find hidden items. Once a rootkit is found right-click on it for lots of options, such as unhooking, killing, dumping the process or driver, wiping the file and more. To top it off, just go to “Report” and click on

Scan to get a full sanitized report of possible rootkits on the system – sweetness.

5.11 Rootkit Hook Analyzer 2.00

Process | File

This is another program requiring an installation prior to use. After clicking on "Analyze" click the "Show hooked services only". It efficiently lists the service name, syscall, address, and related module. You can also export the findings, helpful for the researcher taking notes. Under modules look for Paths with "???", by sorting, to quickly find possible rootkit entries.

6. Anti-Rootkit Tests

Scan times vary between Windows scanning anti-rootkit programs. Most scan within just a few seconds or minutes, dependent upon what it is scanning for and performance of each program. Only GMER takes several minutes to complete. Results for scanning the same system for a rootkit are below:

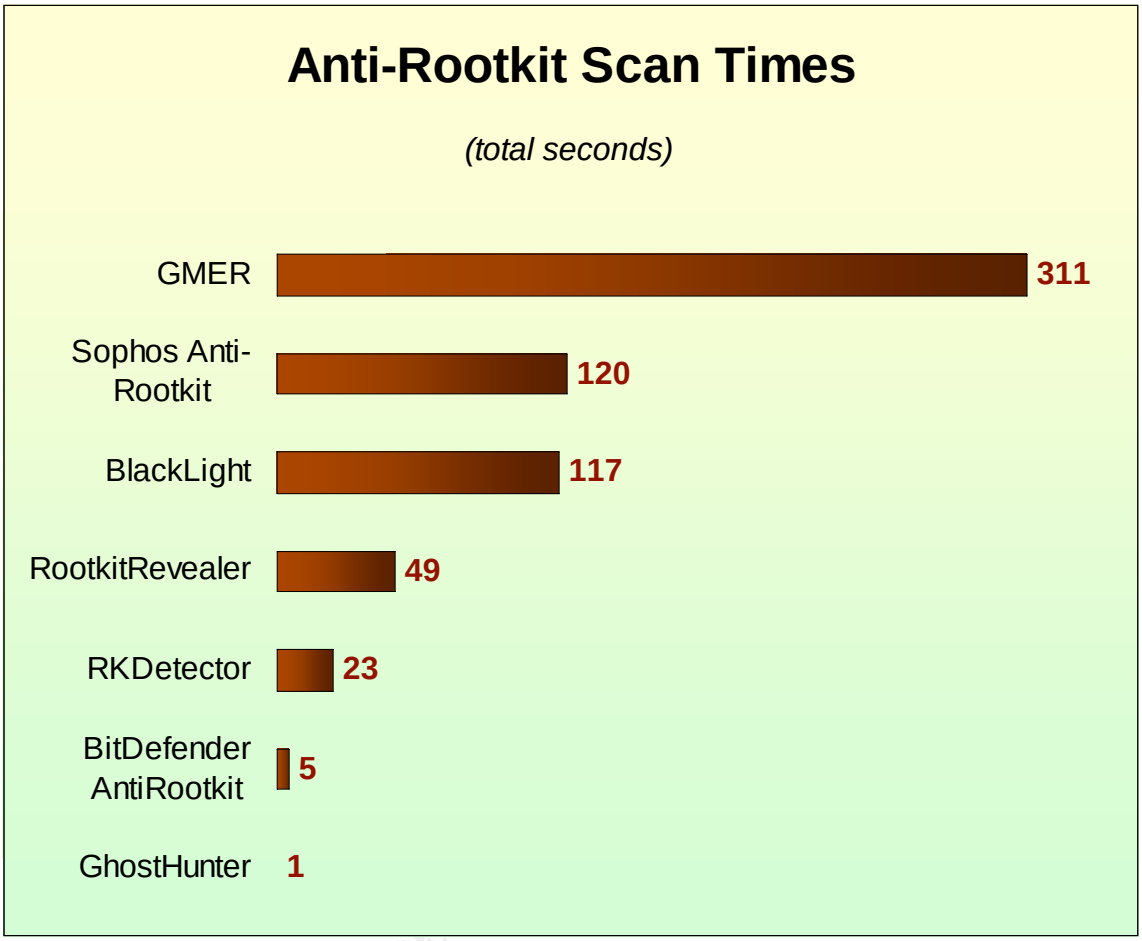


Fig 14: Scan times vary for anti-rootkit programs.

GMER performs an immediate scan upon execution to identify if a hidden process exists. If found it recommends a full system scan, which then takes several minutes to complete. It is the fastest for an immediate triage scan of the system but the slowest for an in-depth scan.

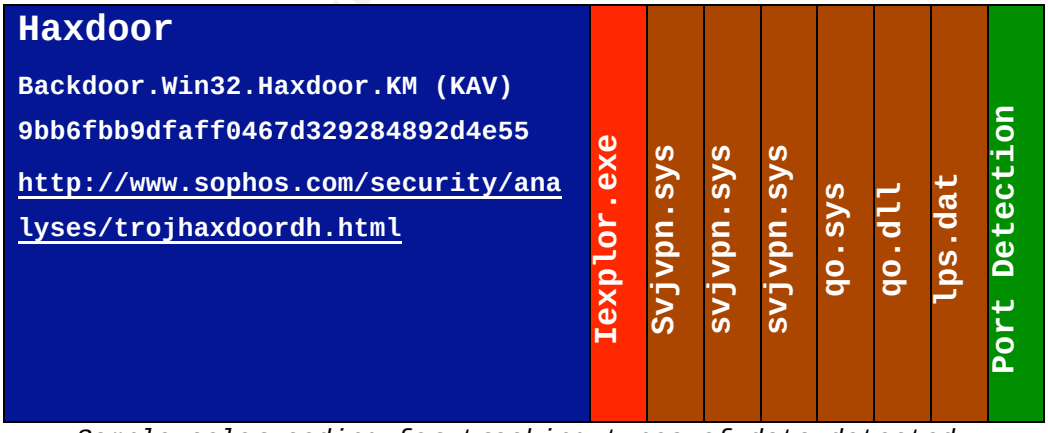
6.1 Anti-Rootkit Test Results

Are they any good, and can they really detect rootkits that exist in the wild today? The matrix below reveals the test results of scans performed on test systems configured for each of the anti-rootkit solutions and rootkits identified in the

matrix. Tests performed for all samples are within Windows XP SP2 VMWare. Items identified for testing are only related to rootkit components for processes, files, and ports. Other data, such as Windows registry or non-rootkit components, are not included in this study.

An example video exists at http://www.kendunham.org/rootkit.wmv to illustrate how rootkits infect and hide on a system and how a researcher may find them using anti-rootkit tools.

The table for each malicious code tested contains the primary family name, Kaspersky (KAV) anti-virus name, MD5 value, and one public report if available. The table is also color coded based upon the type of data potentially detected by anti-rootkit programs. Processes are color coded in red. File data is color coded in brown. Port data is colored in green. Grayed out areas are not applicable to specific programs. In the Haxdoor example below there is one process, several files, and a port related to the malicious code:



Sample color coding for tracking types of data detected.

Tests are repeated at least twice to verify functionality and detection by each program. In cases where incomplete detection takes place, such as finding a hidden prefetch file

but none of the primary rootkit files, it is marked as an undetected rootkit. Each specific file related to the rootkit, process, or port, is marked as either "X" for detected or "U" for undetected.

Note that file names may be dynamic from what is shown on the tables. For example, Feebs always creates a MS** file in the Windows System32 directory, where ** are randomized characters. Repeated tests, for validation, baseline the rootkit data manually prior to testing with each rootkit program to ensure integrity in test results. "*Shared Files" is for many files.

Haxdoor **Backdoor.Win32.Haxdoor.KM (KAV)** **9bb6fbb9dfaff0467d329284892d4e55** **http://www.sophos.com/security/analyses/trojhaxdoordh.html**	explorer.exe	Svjvpn.sys	svjvpn.sys	svjvpn.sys	qo.sys	qo.dll	lps.dat	Port Detection
BitDefender AntiRootkit 6.0.2900.2180	X	X	X	X	X	X	X	
BlackLight 2.2.1055.0	X	X	X	X	X	X	X	
GhostHunter 1.0.0.0	X							
GMER 1.0.12.12011	X	X	X	X	X	X	X	
RKDETECTOR 2.0.0.1		X	X	X	X	X	X	
RootkitRevealer 1.71.0.0		X	X	X	X	X	X	
Sophos Anti-Rootkit 1.2.2	X	X	X	X	X	X	X	
IceSword 1.2.0.0	X	X	X	X	X	X	X	X
DarkSpy 1.0.5.0	X	X	X	X	X	X	X	X
RKUnHooker 3.0.80.290	X	X	U	U	U	U	U	
RootKit Hook Analyzer 2.00		X	U	U	U	U	U	

Special Haxdoor Note: Test results changed significantly for Haxdoor when run on the same system but with the Internet Explorer 7 update. Multiple applications did not execute or work properly. Only GhostHunter, GMER, and RootkitRevealer worked under this configuration.

Feebs **Worm.Win32.Feebs.q (KAV)** **e85829e35b2b5c023deb126dd8014145** **http://www.sophos.com/virusinfo/analyses/w32febsq.html**	Svchost.exe	Mszh.exe	Msxl32.dll	Msha	*Shared files	Port Detected
BitDefender AntiRootkit 6.0.2900.2180	X	X	X	X	X	
BlackLight 2.2.1055.0	U	X	X	X	U	
GhostHunter 1.0.0.0	X					
GMER 1.0.12.12011	U	U	U	U	U	
RKDETECTOR 2.0.0.1		X	X	X	X	
RootkitRevealer 1.71.0.0		U	U	U	U	
Sophos Anti-Rootkit 1.2.2	U	U	U	U	U	
IceSword 1.2.0.0	U	X	X	X	X	X
DarkSpy 1.0.5.0	X	X	X	X	X	X
RKUnHooker 3.0.80.290	X	U	X	X	X	
RootKit Hook Analyzer 2.00		U	U	U	U	

Hupigon **Backdoor.Win32.Hupigon.j (KAV)** **MD5: fde735c7354da0f1e85009c1209ead42** ***No public report on this variant.***	Iexplore.exe	System.dll	System.exe	System_Hook.dll	SystemKey.dll	Port Detection
BitDefender AntiRootkit 6.0.2900.2180	X	U	U	U	U	
BlackLight 2.2.1055.0	X	X	X	X	X	
GhostHunter 1.0.0.0	X					
GMER 1.0.12.12011	X	X	X	X	X	
RKDETECTOR 2.0.0.1		X	X	X	X	
RootkitRevealer 1.71.0.0		X	X	X	X	
Sophos Anti-Rootkit 1.2.2	X	X	X	X	X	
IceSword 1.2.0.0	X	X	X	X	X	X
DarkSpy 1.0.5.0	X	X	X	X	X	X
RKUnHooker 3.0.80.290	X	U	U	U	U	
RootKit Hook Analyzer 2.00		U	U	U	U	

Lecna Backdoor.Win32.Lecna.k (KAV) MD5: 037797242115dfafc2d24ba615fc8ac2 http://www.sophos.com/virusinfo/analyses/trojlecnaf.html	Winword.exe	USBTest.sys	Winword.exe
BitDefender AntiRootkit 6.0.2900.2180	X	U	X
BlackLight 2.2.1055.0	X	U	U
GhostHunter 1.0.0.0	X		
GMER 1.0.12.12011	X	X	X
RKDETECTOR 2.0.0.1		U	X
RootkitRevealer 1.71.0.0		U	X
Sophos Anti-Rootkit 1.2.2	X	U	X
IceSword 1.2.0.0	X	U	X
DarkSpy 1.0.5.0	X	U	X
RKUnHooker 3.0.80.290	X	X	X
RootKit Hook Analyzer 2.00		X	X

Rustock.B Trojan-Clicker.Win32.Costrat.l (KAV) MD5: c6a5c476d0d896e67c1352428fca3e8b http://www.symantec.com/security_response/writeup.jsp?docid=2006-070513-1305-99&tabid=1	Lzx32.exe	Hidden module
BitDefender AntiRootkit 6.0.2900.2180	U	U
BlackLight 2.2.1055.0	U	U
GhostHunter 1.0.0.0		
GMER 1.0.12.12011	X	U
RKDETECTOR 2.0.0.1	U	U
RootkitRevealer 1.71.0.0	U	U
Sophos Anti-Rootkit 1.2.2	X	X
IceSword 1.2.0.0	U	U
DarkSpy 1.0.5.0	U	U
RKUnHooker 3.0.80.290	U	X
RootKit Hook Analyzer 2.00	U	U

Goldun **Trojan-Spy.Win32.Goldun.le (KAV)** **MD5: d4cae0b06e35f4c0069efe9de7872d88** **http://www.sophos.com/security/analyses/trojgoldunea.html**	**K53l0ck.sys**	**t848rom.dll**
BitDefender AntiRootkit 6.0.2900.2180	**X**	**X**
BlackLight 2.2.1055.0	**X**	**X**
GhostHunter 1.0.0.0		
GMER 1.0.12.12011	**X**	**X**
RKDETECTOR 2.0.0.1	**X**	**X**
RootkitRevealer 1.71.0.0	**X**	**X**
Sophos Anti-Rootkit 1.2.2	**X**	**X**
IceSword 1.2.0.0	**X**	**X**
DarkSpy 1.0.5.0	**X**	**X**
RKUnHooker 3.0.80.290	**X**	**U**
RootKit Hook Analyzer 2.00	**X**	**U**

Summary of Rookit Results

In reviewing rootkit results the primary focus is to identify software best able to detect one or more rootkit components on a computer. If at least one rootkit component is detected the analyst is then able to further explorer the computer for possible rootkit and related stealth code in a directed manner.

An analysis of each code, by each program, for capabilities supported by the software is in the graph below. For example, GhostHunter only scans for hidden processes, not hidden files or registry data. GhostHunter is then evaluated on how effective it is at identifying hidden processes for each code. In the case of hidden files, each program is expected to identify all hidden files for maximum detection. In multiple cases programs only detect a few of the files instead of all hidden files on a system.

Overall DarkSpy and IceSword did the best job of reliably detecting rootkits on a computer. IceSword is far superior from a user standpoint, highlighting hidden processes in red and includes additional analyst-like features such as sorting capabilities by date created. Both programs require more advanced knowledge of a system but are excellent for skilled employees. Scanners did a good job of detecting rootkits, especially GMER, often overlooked by many because of the more technical GUI interface.

Total "hits" identifies those programs that flagged files and processes correctly as a possible rootkit. Total "misses" identifies those programs that failed to identify a file or process as a rootkit.

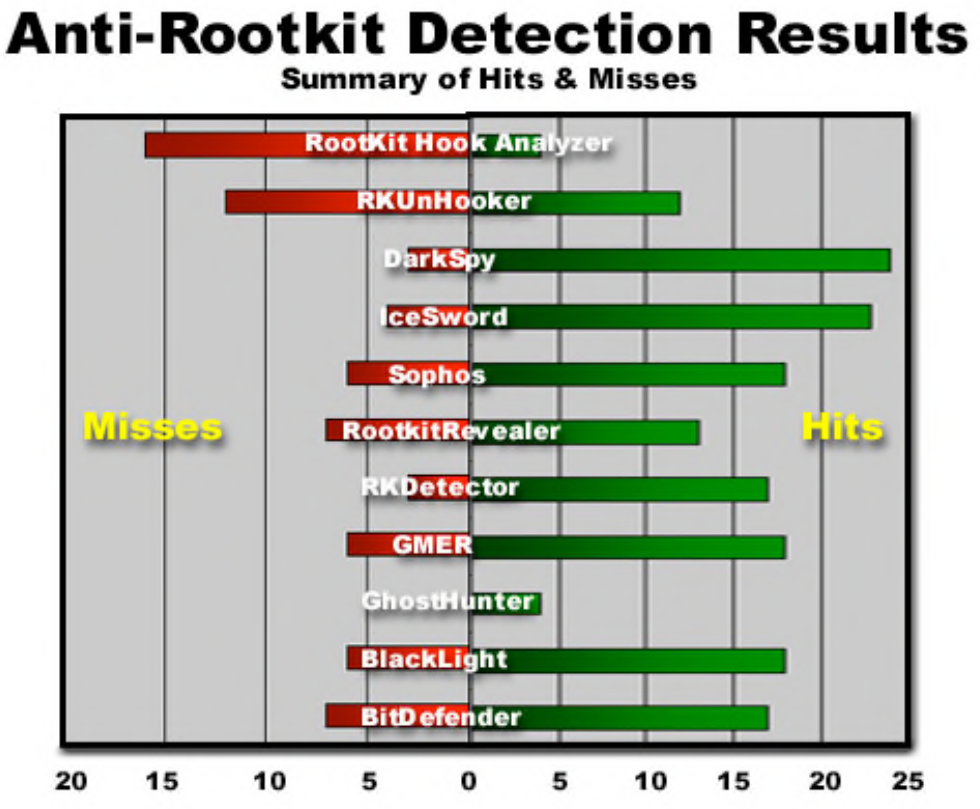


Fig. 15: Rootkit detection results reveals performance of products.

DarkSpy and IceSword have the best overall detection results, with 24 and 23 hits overall. RKUnHooker and RootKit Hook Analyzer missed the most with 12 and 16 misses respectively. For scanners, BlackLight, GMER, and Sophos products scored the best in this limited test, with BitDefender, and RKDetector coming in just one detection behind leading products. This substantiates that most anti-rootkit scanners have similar performance overall, but that some are better at detection specific rootkits than others. Additionally, some have different detection capabilities, such as Sophos being able to detect hidden Windows registry data and IceSword detecting hidden port activity.

When tracking how well each component did for what it could have potentially detected, we have the following percentage graph indicating total effectiveness for each product:

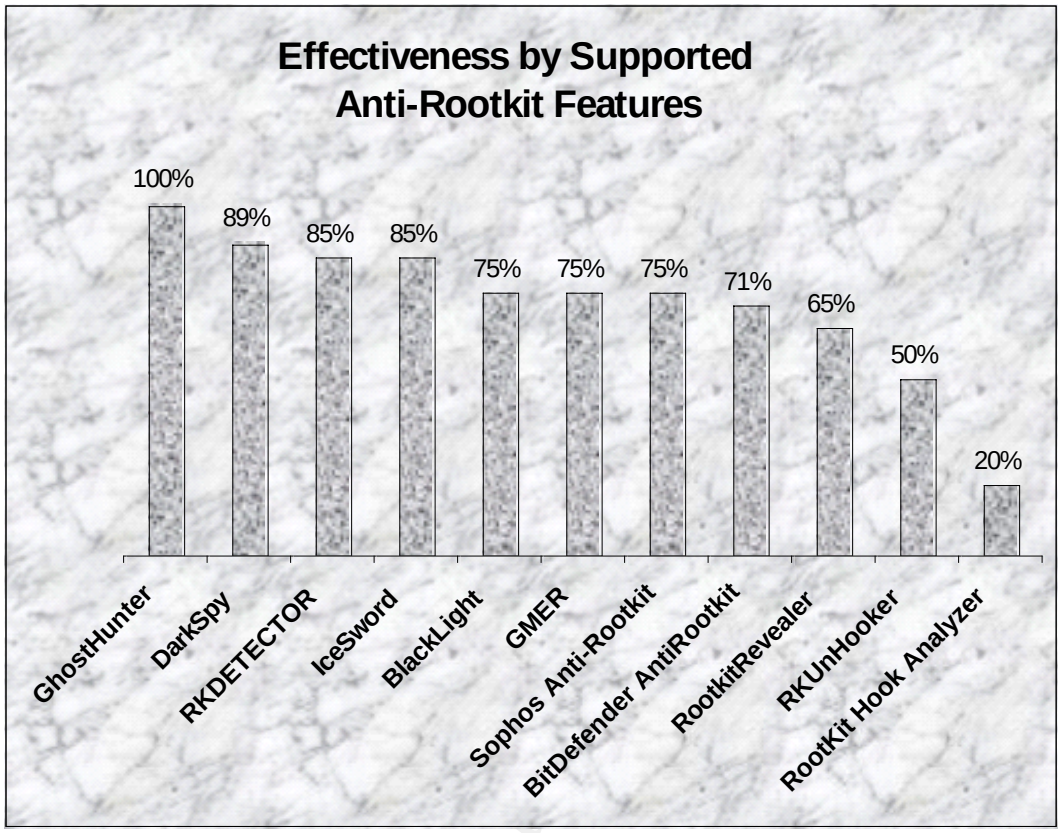


Fig. 16: Effectiveness of each product for supported anti-rootkit features.

This graph proves that most products do a good job at their promoted capabilities. Only RootkitRevealer, RKUnhooker, and RootKit Hook Analyzer performed at levels of 65% or less for detection capabilities. RootkitRevealer is slightly disadvantaged as a forerunner in the market, freeware commonly targeted by attackers for counter-measures today.

7. Rooting Out Rootkits

As the Director of the Rapid Response Team for VeriSign iDefense, I regularly serve as an extension of the team for the largest networks in the world. What we do best is to rapidly analyze new threats to identify behavioral information related to the sample. This then empowers an administrator to audit hosts and network traffic for malicious code. But the most challenging part for administrators is identification and capture of the rootkit sample.

The focus of rootkit mitigation, in this section, is on identification of the threat. Obviously, prevention against attack is best-practices advice, involving both training and technology. But once a suspect computer is discovered, what does the administrator do?

It's not uncommon for administrators to notice network traffic and then attempt to locate questionable or clearly malicious programs on a computer. Once administrators discover a file, they submit it to anti-virus software for signature updates in hopes of installing updated signatures to detect and/or remove it from additional computers. The problem with this approach when it comes to rootkits is that the rootkit may successfully undermine all host-based solutions.

A better strategy with rootkit infections is to do the following:

- Identify the root cause of the infection. Is there a vulnerability being exploited? Is it user-interaction based? Why did the host become infected and are others in the network likely infected?

- Capture suspect files, analyze them and submit them to anti-virus vendors. Confirm malicious files and then perform aggregate research and behavioral analysis to fully understand the nature of the threat. If you can identify which anti-rootkit programs work well against the threat, within a test environment, use those on suspect hosts.
- Recognize that it's highly likely that additional codes are installed on the computer, especially when rootkits (more sophisticated codes) are involved in the attack. **Re-imaging is the only good option to maintain integrity for many of the threats facing networks today**. That said, it's still critical to fully analyze the threat to best identify all possible vectors for identifying the threat across the network, such as egress traffic, processes or files that may be visible on hosts, etc.
- Aggressively monitor mitigated hosts and the network to ensure mitigation of malicious code.

A host of rootkit detection programs have emerged in the wake of rootkit development over the past few years, as evident in test results revealed in this report. Anti-virus programs may detect various components of code, such as the e-mail component of Feebs, but still not detect the rootkit component by default. Many Trojans and other malicious codes attempt to disable or delete hundreds of security software services on a computer when infecting to protect against identification and mitigation. This fact, coupled with a lack of strong rootkit detection capabilities built into any mainstream anti-virus product for default configuration, creates a great opportunity for criminals to maintain stealth for survival and financial

gain. Programs like the Enterprise edition of Encase offer anti-rootkit response and forensic capabilities, gaining favor with some.

Analysis of network communications is critical for anti-rootkit efforts today. Many times successful rootkit attacks are first noticed via a SNORT trip, IDS/IPS, or Firewall log file. Administrators then identify the IP associated with the trip and investigate the host. Several anti-rootkit programs may then be used to successfully detect most rootkits in the wild today, as proven with lab results in this report.

In more serious case, where persistent network activity is identified with no clear process or file association, mounting the drive for analysis on another machine is necessary to remove the control of the rootkit over scanning or analysis results. Forensic review of a disk is done through hash checks, anti-virus and anti-rootkit scans, and manual inspection of Windows, Windows/System32, and other locations. Questionable or overtly hostile data found in a forensic investigation often leads to proper threat identification and mitigation.

In summary, my personal approach is to use a variety of tools to help narrow in on a possible rootkit. If egress activity is identified, then Windows Task Manager, FPort, or Process Monitor should quickly reveal the process responsible for the egress traffic. If no such process is found then running a quick anti-rootkit scan and IceSword quickly find most rootkits that may exist on a computer. If nothing is found then mounting the drive for manual scanning and inspection is the last and most timely step to take in responding to a possible rootkit threat. As information is identified during this entire process, various research paths are then initiated, such as

scanning a found malicious sample, testing it within VMWare to identify behavioral information, and more.

7.1 DiskMount for VMWare Analysis

For networks supporting virtual systems DiskMount is an excellent utility for analyzing VMWare images (wmare.com). A few quick practical instructional points for using VMWare DiskMount are below:

- You install it on the host machine to analyze guest operating systems infected with code.
- The guest operating system must be shut down within VMWare so that the host can mount it.
- Click Start/Run… and type cmd to open a command window.
- Browser to the DiskMount directory and drag and drop vmware-mount into the command window. Then enter a space, "z:" (no quotes), a space, and then drag and drop path to the VMware image to be mounted (this mounts it as the Z drive on the host). I prefer to sort my VMWare directory by last modified to locate the proper snapshot to load.

  e.g. vmware-mount LETTER: "C:\PATH\NAME.vmdk"
- Browse to the mounted drive letter on the host. Look for files in common locations, such as Windows and Windows/System32, and sort by creation date to quickly find most if not all of the files created by the rootkit.
- When done use the /d or /f options to unmount the drive.

  e.g. vmware-mount /d z:

note: /d never seems to work for me, so I regularly use /f to forcefully dismount the drive.

CAUTION: I have found that DiskMount may corrupt snapshots of a VMWare image. Always have a full backup of a VMWare image before using DiskMount.

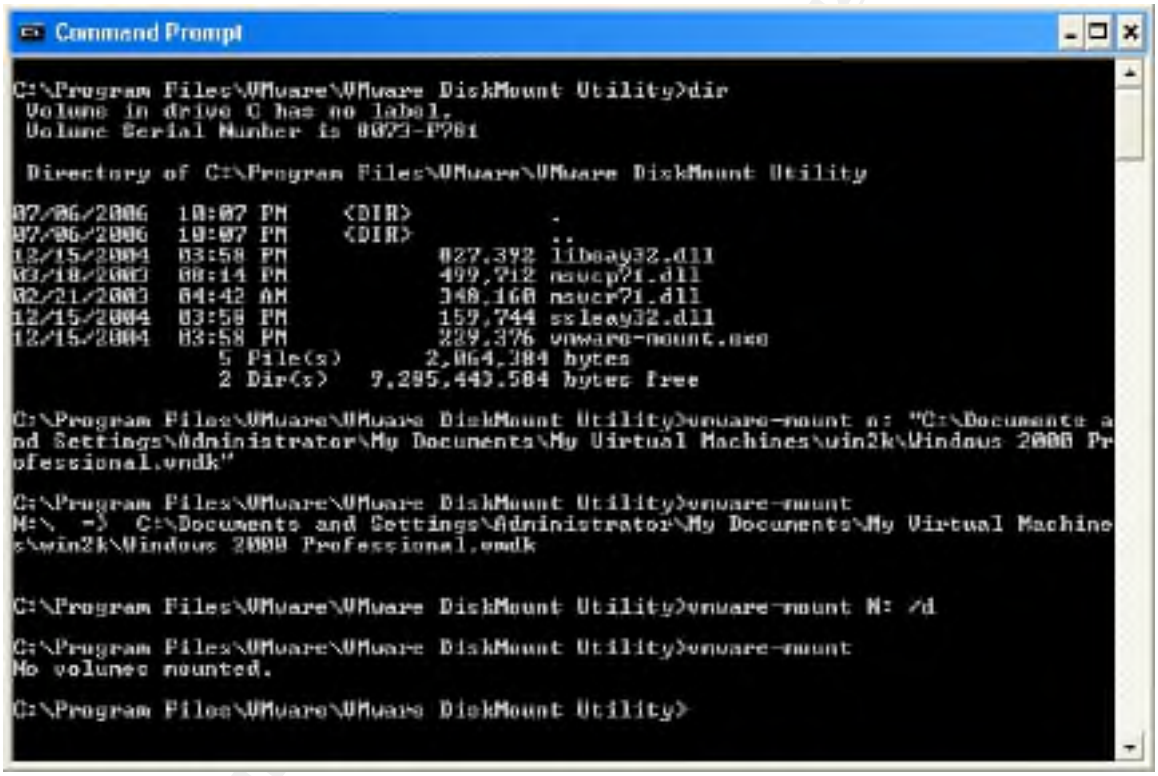


Fig 17: VeriSign/iDefense: Using DiskMount to mount a virtual volume.

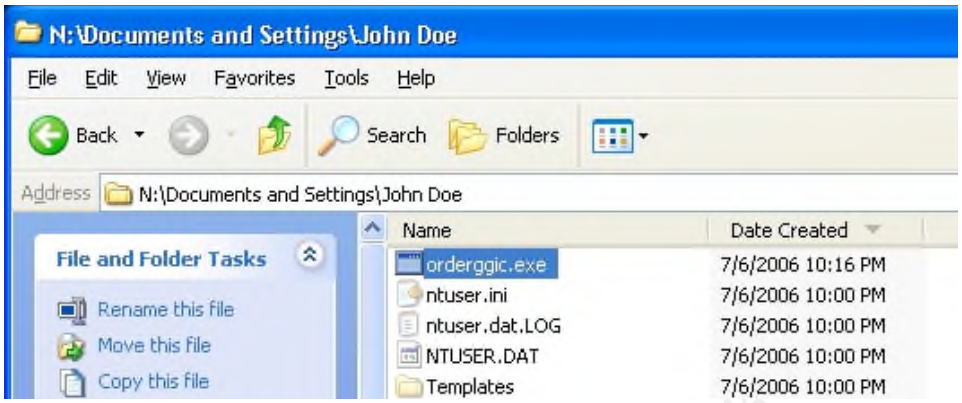


Fig 18: VeriSign/iDefense: Rootkit files are visible via DiskMount.

7.2 Re-Image the Disk: Do I Have Too?

Many organizations attempt to manually remove known components of an attack to mitigate a malicious code incident. The problem with this approach is the lack of integrity introduced to a system upon compromise, especially with stealthy codes and rootkits.

The longer an attacker has control over a system the more likely it is that additional new undetected code is uploaded to the computer. It is a common practice amongst hackers to upload private code to infected computers. This "private code" is not spammed out but carefully controlled, only sent to infected computers with disabled anti-virus. Even if updated anti-virus is able to detect the original source of the infection the newer "private code" goes undetected and survives in a manually cleaned system.

Many organizations today are moving toward full image reinstallations to fully mitigate all malicious codes that may reside on a system. Leading organizations are also considering deployment of virtualization environments, where all data is stored on a centralized server and hosts are simple virtual

machines that easily rebooted into a clean state for each time of use.

7.3 Patch Guard

Microsoft has released Patch Guard for x64-based Windows, with multiple improvements impacting rootkit attempts (Microsoft Corp., 2007). According to Microsoft Patch Guard prohibits drivers on x64-based systems from doing the following:

- Modifying system service tables, for example, by hooking KeServiceDescriptorTable
- Modifying the interrupt descriptor table (IDT)
- Modifying the global descriptor table (GDT)
- Using kernel stacks that are not allocated by the kernel
- Patching any part of the kernel (detected only on AMD64-based systems)

Patch Guard successfully protects against popular rootkits such as Hacker Defender (Rutkowska, 2006). It is not a silver bullet, undermined by more sophisticated rootkits.

7.4 Use Anti-virus Software

Most anti-virus programs now offer some level of anti-rootkit support. At a minimum, detection of non-rootkit files serves as an indicator that additional codes may be installed on a computer. Anti-rootkit techniques and tools need to be implemented into daily operating procedures for responding to suspect and known malicious code incidents on a computer.

7.5 Browser Help Objects

BHOs are easily detected today as stand-alone threats. The problem exists where BHOs are part of a sophisticated attack involving rootkit technology, multiple codes, etc. For the average BHO attack today simply use upgraded Internet Explorer to view installed BHOs. Select "Manage Add-ons" from the Tools menu to display a list of all BHOs, aka add-ons or extensions. Then identify each and perform research to identify those that should not be on a system. Having a baseline of registered BHOs for an installation is critical in facilitating incident response within a large network.

There are also many good third-party products useful in identifying BHOs. Keep in mind that FireFox and other browsers are increasingly coming under attack, so auditing for extensions in alternative browsers is important where such solutions are employed.

If a hostile BHO is detected it's trivial to click on the item and then "disable" within Internet Explorer. Full research is in order to identify if the BHO is related to other threats or codes. For example, some BHOs are installed as a result of exploitation from a website where multiple codes are installed. Again, any malicious behavior on a computer may serve as an indicator of additional issues not yet discovered on the host.

7.6 Alternate Data Streams

Information stored in ADS is not fully visible in the directory using traditional DOS commands such as DIR. To do this, run a tool like LADS and then the DIR command within the command prompt to view any ADS within that directory as shown in the sample image below.

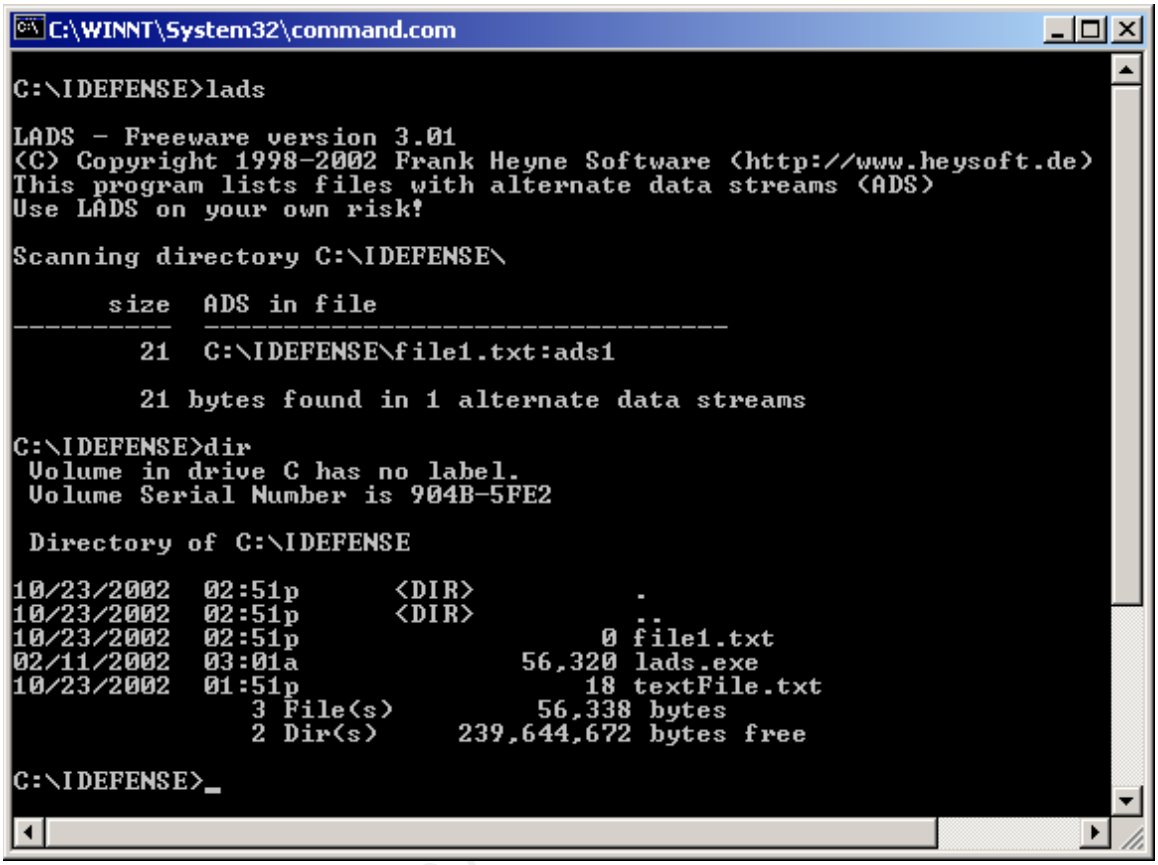


Fig 16: VeriSign/iDefense: An Alterative Data Stream called ads1 is revealed using the LADS freeware program.

A variety of ADS identification and management utilities exist today including but not limited to:

- **ADS Locator**
  http://www.safer-networking.org/en/tools/tools_ads.html
- **ADS Spy GUI Scanner**
  http://www.spywareinfo.com/~merijn/downloads.html
- **Crucial ADS GUI Scanner**
  http://www.crucialsecurity.net//index.php?option=com_content&task=view&id=95&Itemid=137

- **LADS**
  http://www.heysoft.de/nt/ep-lads.htm
- **ScanADS command line tool**
  http://www.kodeit.org/products/scanads/
- **Shredator**
  http://www.shredator.com/commondownloadpage.asp
- **Streams**
  http://www.microsoft.com/technet/sysinternals/utilities/streams.mspx
- **The Forensic ToolkitTM v2.0**
  http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/proddesc/forensic-toolkit.htm

ADS are slowly being exploited by malicious actors to conceal infections and store data in a hidden location. There are cases where ADS based code existed on a computer and the administrator didn't know how to search ADS for data, and as a result, re-imaged the computer before fully understanding the threat. Analysis of ADS is critical for forensics or in working with law enforcement, to ensure that all evidence is properly analyzed and collected.

8. Concluding Comments

Criminal motives drive cyber-attacks today. It's not about 15 minutes of fame anymore. It's about multi-billion dollar fraud operations by organized professionals. Stealth is here to stay, with criminals aggressively countering known security measures and techniques in a competitive fashion with the world of cyber-security. Advancements in stealth are expected to continue and become increasingly complex over the next 18 months.

Today the use of ADS continues to be limited, largely due to the limited benefit of using ADS to conceal data on a computer. It takes additional effort on the part of the attacker and doesn't reap that much in terms of benefits. Today it is far more common for an attacker to copy and paste anti-security process routines into source code to then attempt to terminate processes that may detect the malicious code on a computer. This is far easier, popularized, and effective compared to storing data in ADS to avoid anti-virus detection. As a result, competitive counter-measures limit the use of ADS by criminals to date.

Browser Help Objects and extensions in general are a cause for concern, especially as hostile BHO attacks are on the rise for the Windows platform. Increased transparency has greatly improved the ability for consumers and enterprise to audit for hostile BHOs and mitigate respectively. This is still a major growth area of attack and fairly stealthy for the average user who doesn't understand nor even know about BHOs in most cases – let alone how to identify a malicious BHO. Simple education, use of popular freeware tools, and simple Internet searches for

potentially hostile CLSIDs is all that is required for baseline instruction regarding BHOs.

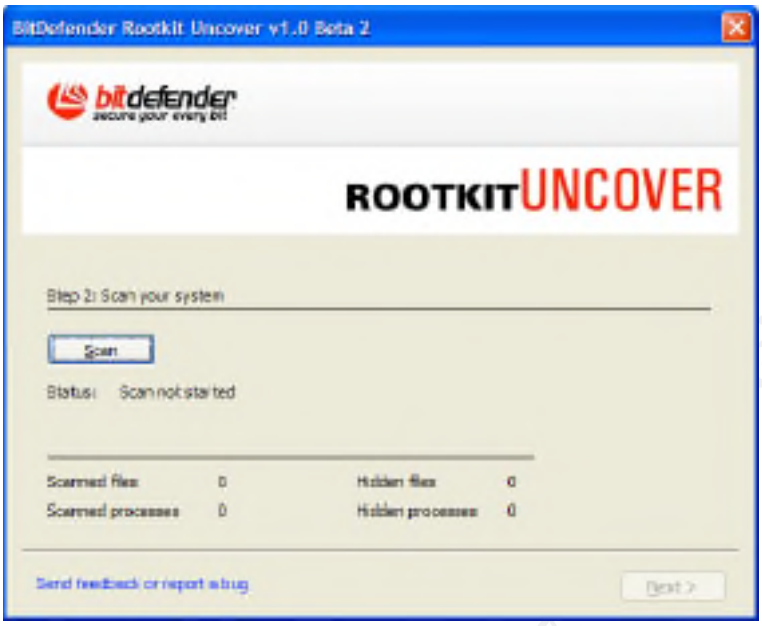
Windows rootkits are increasingly common and sophisticated. Approximately 10 percent of cases handled by the author involve sophisticated codes that are anti-VMWare, requiring use of a goat lab machine or other sophisticated analysis tools. The advent of Rustock in the wild, fueled by Russian criminals, is troubling. This highly sophisticated rootkit is not easily detected by popular and trusted anti-rootkit tools. This forces the administrator to improve egress monitoring of traffic from hosts and have a large suite of tools available for anti-rootkit analysis. In some cases expensive forensic and incident response procedures may be required to fully identify potential rootkit codes. Fortunately, rootkits that control the computer on a deeper level are not in the wild at this time and are only proof-of-concept.

This report proves that no single program for stealth analysis will do it all. A suite of tools is required, with an understanding of each, to properly identify threats that may exist on a computer. Each tool may render different results, providing clues to possible stealth code on a computer. By learning how to use each tool for its strengths, researchers are best enabled to quickly triage suspect computers for possible stealth code attacks.

9. Appendix A – Screenshots of Windows Anti-Rootkit Tools

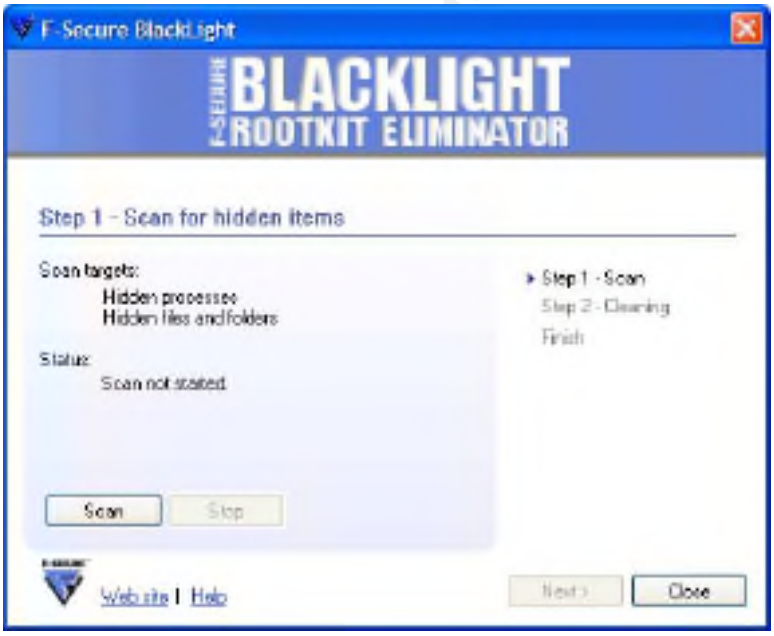
BitDefender AntiRootkit 6.0.2900.2180

http://beta.bitdefender.com/login.php



BlackLight 2.2.1055.0

http://www.f-secure.com/blacklight/

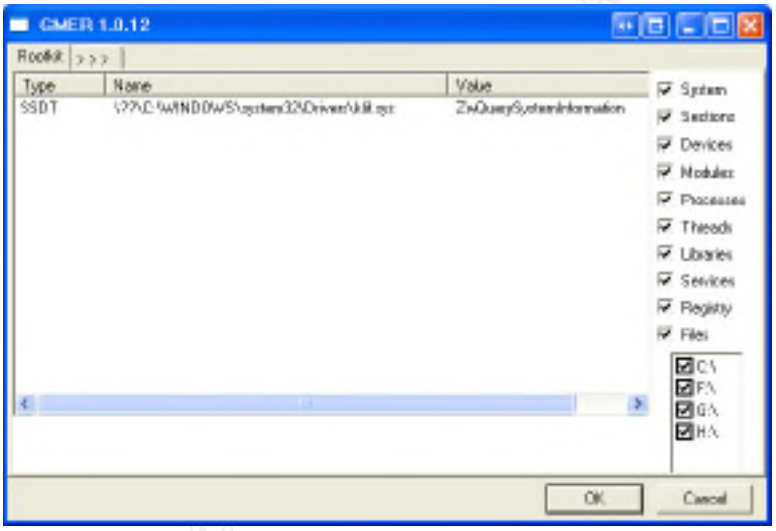


GhostHunter 1.0.0.0

paolo.monti@effetime.it

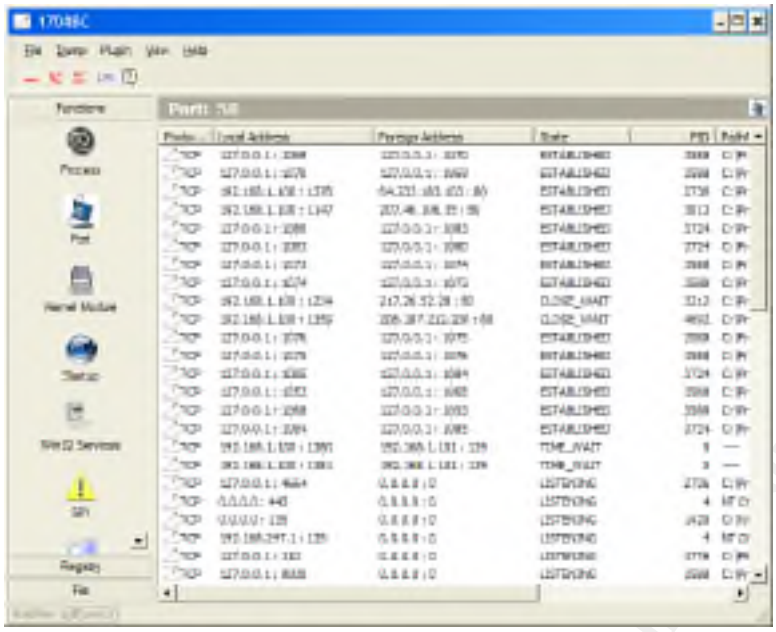
GMER 1.0.12.12011

http://martijnc.be/tools/mieeeeeeeeeeeeeeeeep/gmer.htm



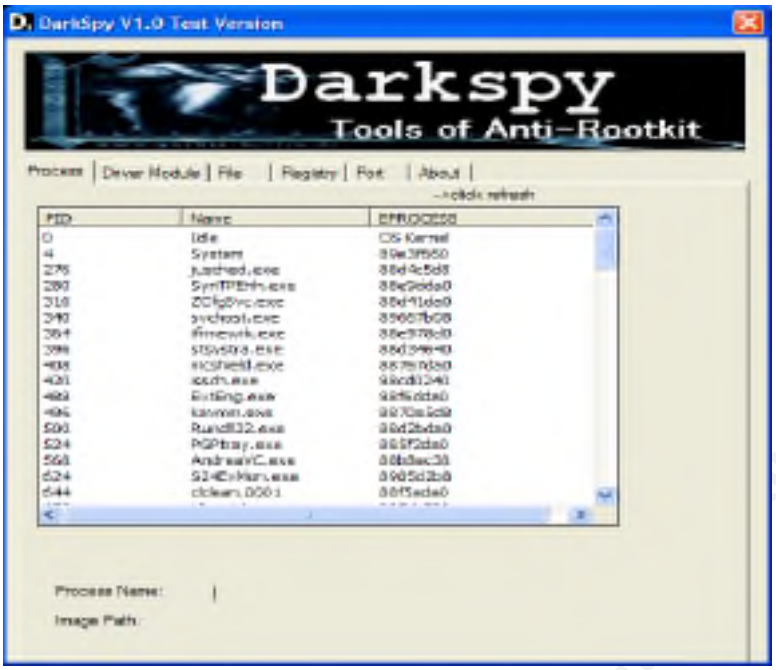
IceSword 1.2.0.0

`http://202.38.64.10/%7Ejfpan/download/IceSword120_en.zip`



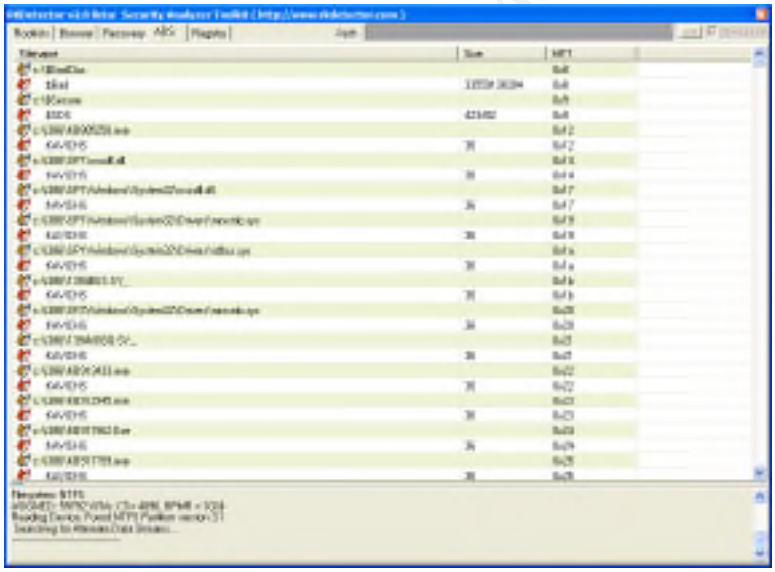
DarkSpy 1.0.5.0

http://www.rootkit.com/newsread.php?newsid=474



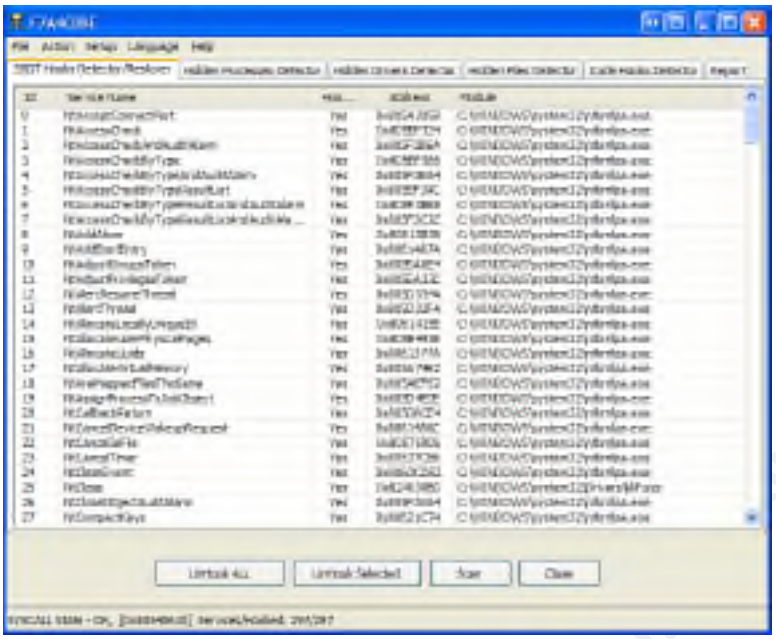
RKDETECTOR 2.0.0.1

http://www.rkdetector.com/



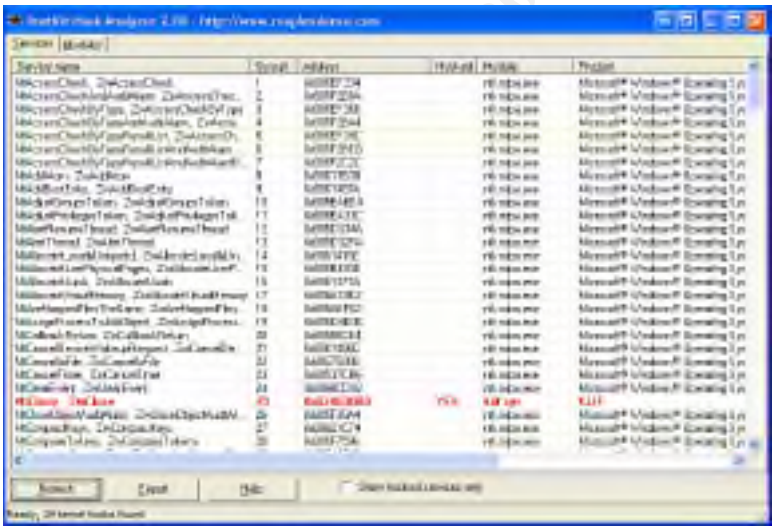
RKUnHooker 3.0.80.290

http://rkunhooker.narod.ru/



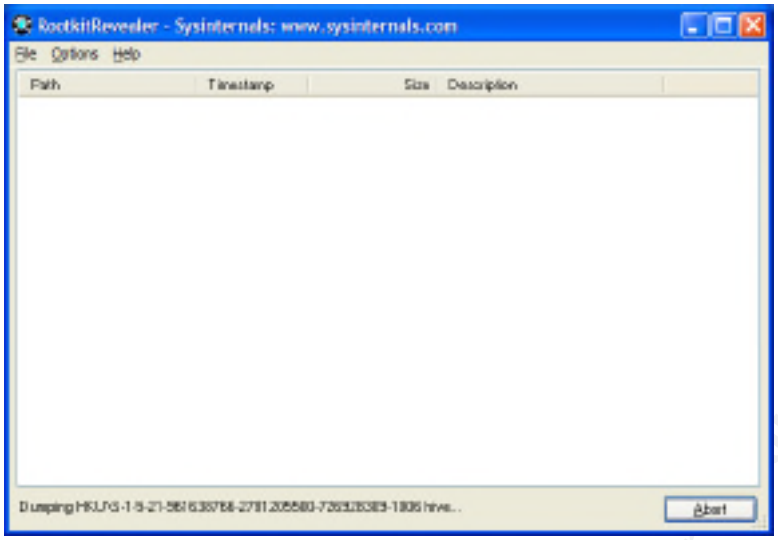
RootKit Hook Analyzer 2.00

http://www.resplendence.com/hookanalyzer



RootkitRevealer 1.71.0.0

http://www.microsoft.com/technet/sysinternals/utilities/RootkitRevealer.mspx



Sophos Anti-Rootkit 1.2.2

http://www.sophos.com/products/free-tools/sophos-anti-rootkit.html

10. Appendix B - Multiscan Results

Multiscan results, courtesy of AV-Test.org, for each malicious code sample used in testing are below. Only those programs that detect something suspicious or an actual family/variant name are included in the results below (non-detections removed). This information helps to cross-correlate samples via names assigned by different vendors.

Scan report of: FEEBS.exe

@Proventia-VPS	Malicious (Cancelled)
AntiVir	Worm/Feebs.A
Avast!	Win32:Feebs-M [Wrm]
AVG	Worm/Generic.GC
BitDefender	Win32.Worm.Feebs.Q
ClamAV	Worm.Feebs.AE
Command	W32/Feebs.N (exact)
Dr Web	Win32.HLLM.Graz
eSafe	Win32.Polipos.sus
eTrust-VET	Win32/Feeb.K
eTrust-VET (BETA)	Win32/Feeb.K
Ewido	Worm.Feebs.q
F-Prot	W32/Feebs.N (exact)
F-Secure	Worm.Win32.Feebs.q
F-Secure (BETA)	Worm.Win32.Feebs.q
Fortinet	W32/Feebs.Q!worm
Fortinet (BETA)	W32/Feebs.Q!worm
Ikarus	Worm.Win32.Feebs.Q
Kaspersky	Worm.Win32.Feebs.q
McAfee	W32/Feebs.gen@MM
McAfee (BETA)	W32/Feebs.gen@MM
Microsoft	Win32/Feebs.gen@mm
Nod32	Win32/Mocalo.N worm
Norman	W32/Feebs.AA
Panda	W32/Feebs.G.worm
Panda (BETA)	W32/Feebs.G.worm
QuickHeal	Worm.Feebs.q
Rising	Worm.Feebs.ap
Sophos	W32/Feebs-Gen
Symantec	W32.Feebs
Symantec (BETA)	W32.Feebs
Trend Micro	WORM_FEEBS.AX
Trend Micro (BETA)	WORM_FEEBS.AX
UNA	Worm.Win32.Feebs.q
VBA32	Embedded.MalwareScope.Worm.Feebs.1 (suspected)
VirusBuster	Worm.Feebs.G
WebWasher	Worm.Feebs.A

Scan report of: HAXDOOR.exe

@Proventia-VPS	Malicious (Cancelled)
AntiVir	BDS/Haxdoor.KM
Avast!	Win32:Trojan-gen. {Other}
AVG	BackDoor.Generic3.KBB (Trojan horse)
BitDefender	Backdoor.Haxdoor.KM
ClamAV	Trojan.Haxdoor-117
Command	W32/Haxdoor.LH@bd
Dr Web	BackDoor.Haxdoor.340
eSafe	Win32.Haxdoor.il
eTrust-VET	Win32/Haxdoor!generic
eTrust-VET (BETA)	Win32/Haxdoor!generic
Ewido	Backdoor.Haxdoor.km
F-Prot	W32/Haxdoor.LH@bd
F-Secure	Backdoor.Win32.Haxdoor.km
F-Secure (BETA)	Backdoor.Win32.Haxdoor.km
Fortinet	BDoor.BAC!tr.bdr
Fortinet (BETA)	BDoor.BAC!tr.bdr
Ikarus	suspicious
Kaspersky	Backdoor.Win32.Haxdoor.km
McAfee	BackDoor-BAC.gen.e trojan
McAfee (BETA)	BackDoor-BAC.gen.e trojan
Microsoft	Backdoor:Win32/Haxdoor!B69B
Nod32	Win32/Haxdoor trojan
Norman	W32/Haxdoor.AXG
Panda	Bck/Haxdoor.MN
Panda (BETA)	Bck/Haxdoor.MN
QuickHeal	Backdoor.Haxdoor.km
Rising	Backdoor.Haxdoor.ww
Sophos	Troj/Haxdoor-DB
Symantec	Backdoor.Haxdoor
Symantec (BETA)	Backdoor.Haxdoor
Trend Micro	BKDR_HAXDOOR.IL
Trend Micro (BETA)	BKDR_HAXDOOR.IL
UNA	Backdoor.Haxdoor.4949
VBA32	Backdoor.Win32.Haxdoor.km
VirusBuster	novirus:Packed/FSG
WebWasher	Trojan.Haxdoor.KM

Scan report of: GOLDUN.exe

@Proventia-VPS	Malicious (Cancelled)
AntiVir	TR/Spy.Goldun.LE
Avast!	Win32:Goldun-EN [Trj]
AVG	PSW.Goldun.DN (Trojan horse)
BitDefender	Trojan.PWS.Agent.CL
ClamAV	Trojan.Spy.Goldun-141
Command	W32/Dropper.gen2
Dr Web	Trojan.PWS.GoldSpy
eSafe	Trojan/Worm [100] (suspicious)
eTrust-VET	Win32/Haxdoor!generic
eTrust-VET (BETA)	Win32/Haxdoor!generic
Ewido	Logger.Goldun.le
F-Prot	W32/Dropper.gen2
F-Secure	Trojan-Spy.Win32.Goldun.le
F-Secure (BETA)	Trojan-Spy.Win32.Goldun.le

Fortinet	Spy/Goldun
Fortinet (BETA)	Spy/Goldun
Ikarus	suspicious
Kaspersky	Trojan-Spy.Win32.Goldun.le
McAfee	PWS-Goldun.dr trojan
McAfee (BETA)	PWS-Goldun.dr trojan
Microsoft	Trojan:Win32/HideDrv.gen!sys
Nod32	Win32/Spy.Goldun.HP trojan
Norman	W32/Goldun.ABM
Panda	Suspicious file
Panda (BETA)	Suspicious file
QuickHeal	Suspicious (warning)
Rising	Trojan.Spy.Goldun.yr
Sophos	Mal/Packer
Symantec	Trojan.Goldun
Symantec (BETA)	Trojan.Goldun
VBA32	Trojan-Spy.Win32.Goldun.le
VirusBuster	novirus:Packed/FSG
WebWasher	Trojan.Spy.Goldun.LE

Scan report of: RUSTOCK.exe

@Proventia-VPS	Malicious (Cancelled)
AntiVir	TR/Click.Costrat.E.2
Avast!	Win32:Costrat-L [Trj]
AVG	Clicker.CVC (Trojan horse)
BitDefender	Trojan.Clicker.Costrat.O
ClamAV	Trojan.Clicker-3
Command	W32/Trojan.JQB (destructive program)
Dr Web	Trojan.Spambot
eSafe	Win32.Costrat.l
eTrust-INO	Win32/Rustock.7ri!Trojan
eTrust-INO (BETA)	Win32/Rustock.7ri!Trojan
eTrust-VET	Win32/Rustock.Q
eTrust-VET (BETA)	Win32/Rustock.Q
Ewido	Hijacker.Costrat.l
F-Prot	W32/Trojan.JQB (destructive program)
F-Secure	Trojan-Clicker.Win32.Costrat.l
F-Secure (BETA)	Trojan-Clicker.Win32.Costrat.l
Fortinet	Adware/Costrat
Fortinet (BETA)	Adware/Costrat
Kaspersky	Trojan-Clicker.Win32.Costrat.l
McAfee	Spam-Mailbot.c trojan
McAfee (BETA)	Spam-Mailbot.c trojan
Microsoft	Rustock (threat-c)
Nod32	Win32/Rustock.NAJ trojan
Norman	W32/Smalldrp.KEP
Panda	Trj/Clicker.SU
Panda (BETA)	Trj/Clicker.SU
QuickHeal	TrojanClicker.Costrat.l
Rising	Trojan.Clicker.Costrat.r
Symantec	Backdoor.Rustock.B
Symantec (BETA)	Backdoor.Rustock.B
Trend Micro	TROJ_COSTRAT.L
Trend Micro (BETA)	TROJ_COSTRAT.L
UNA	TrojanClicker.Win32.Costrat.241B
VBA32	Trojan-Clicker.Win32.Costrat.l

VirusBuster	Trojan.CL.Costrat.X
WebWasher	Trojan.Click.Costrat.E.2
YY_Spybot	Smitfraud-C.,,Temporary file

Scan report of: HUPIGON.exe

@Proventia-VPS	Malicious (Cancelled)
AntiVir	BDS/Hupigon.I.2
Avast!	Win32:Hupigon-BX [Trj]
AVG	BackDoor.Generic.G (Trojan horse)
BitDefender	Backdoor.Hupigon.E
ClamAV	Trojan.Hupigon-1357
Command	W32/Hupigon.I@bd
Dr Web	BackDoor.Pigeon.204
eTrust-VET	Win32/Pigeon!generic
eTrust-VET (BETA)	Win32/Pigeon!generic
Ewido	Backdoor.Hupigon.j
F-Prot	W32/Hupigon.I@bd
F-Secure	Backdoor.Win32.Hupigon.j
F-Secure (BETA)	Backdoor.Win32.Hupigon.j
Fortinet	W32/HUPIGON.J!tr.bdr
Fortinet (BETA)	W32/HUPIGON.J!tr.bdr
Kaspersky	Backdoor.Win32.Hupigon.j
McAfee	BackDoor-AWQ.b trojan
McAfee (BETA)	BackDoor-AWQ.b trojan
Microsoft	Backdoor:Win32/Hupigon!94A4
Nod32	Win32/Hupigon trojan (variant)
Norman	W32/Hupigon.OY
Panda	Backdoor Program.AP
Panda (BETA)	Backdoor Program.AP
Rising	Backdoor.Gpigeon.cx
Sophos	Troj/Feutel-Gen
Symantec	Backdoor.Graybird.L
Symantec (BETA)	Backdoor.Graybird.L
Trend Micro	BKDR_HUPIGON.GEN
Trend Micro (BETA)	BKDR_HUPIGON.GEN
VBA32	MalwareScope.Backdoor.Hupigon.11
VirusBuster	Backdoor.Hupigon.Gen.2
WebWasher	Trojan.Hupigon.I.2
YY_Spybot	Jupilites,,Installer

Scan report of: LECNA.exe

@Proventia-VPS	Malicious (Cancelled)
AntiVir	TR/Lecna.D
Avast!	Win32:Trojan-gen. {Other}
AVG	BackDoor.Generic.JYT (Trojan horse)
BitDefender	Backdoor.Lecna.K
Command	W32/Lecna.J@bd
Dr Web	BackDoor.Lecnac
eSafe	Trojan/Worm [101] (suspicious)
eTrust-INO	Win32/Lecna.A!Trojan
eTrust-INO (BETA)	Win32/Lecna.A!Trojan
eTrust-VET	Win32/Lecna.A
eTrust-VET (BETA)	Win32/Lecna.A
Ewido	Backdoor.Lecna.k
F-Prot	W32/Lecna.J@bd

F-Secure	Backdoor.Win32.Lecna.k
F-Secure (BETA)	Backdoor.Win32.Lecna.k
Fortinet	W32/Lecna.CSB!tr.bdr
Fortinet (BETA)	W32/Lecna.CSB!tr.bdr
Ikarus	Backdoor.Win32.Lecna.d
Kaspersky	Backdoor.Win32.Lecna.k
McAfee	BackDoor-CSB trojan
McAfee (BETA)	BackDoor-CSB trojan
Microsoft	Backdoor:Win32/Lecna.D
Nod32	Win32/Lecna.A trojan
Norman	W32/Lecna.B
Panda	Trj/Ranky.FY
Panda (BETA)	Trj/Ranky.FY
QuickHeal	Backdoor.Lecna.d
Rising	Backdoor.Lecna.a
Sophos	Troj/Lecna-D
Symantec	Downloader
Symantec (BETA)	Downloader
Trend Micro	BKDR_LECNA.E
Trend Micro (BETA)	BKDR_LECNA.E
UNA	Backdoor.Lecna.9884
VBA32	BackDoor.Lecnac
VirusBuster	Backdoor.Lecna.B
WebWasher	Trojan.Lecna.D

11. References

Abhijit, Using CR0 register in a driver. Web site: http://www.tutorials-blog.com/nt/Using-register/

Anonymous, ZwWriteFile with Synchronization. Retrieved March 12, 2007, Web site: http://www.tutorials-blog.com/nt/nt-8.html

Answers.com, (2007). ring. Web site: http://www.answers.com/topic/ring-computer-security

Atarasco (2007). RK detector. Web site: http://www.rkdetector.com/

Bartheld, Volker (2001, May 29). HOOK - A How To for setting system wide hooks. Web site: http://www.codeguru.com/Cpp/W-P/system/misc/article.php/c5685/

Benny, Ratter, (2005, September). Win2k.Stream. 29A Issue, Issue 5, Web site: http://vx.netlux.org/29a/29A-5.html

BitDefender/Softwin (2007). BitDefender Antirootkit Beta. Web site: http://beta.bitdefender.com/login.php

Cardmagic (2007). Introduction To DarkSpy. Web site: http://www.rootkit.com/newsread.php?newsid=474

Cummings, Chris (2006). An introduction to hook procedures. Web site: http://delphi.about.com/library/bluc/text/uc063001a.htm

Dunham, Ken (2006). Top Threats & Trends of 2005: A Forward Looking View.

Dunham, Ken (2006, May 24). Anatomy of MetaFisher Attacks.

Dunham, Ken (2006, March 3). Feebs Parts 1 & 2.

Dunham, Ken (2004, October 29). Introduction: Alternate Data Streams (ADS).

Dunham, Ken (2006, May 24). Metafisher Trojan Activity.

Dunham, Ken (2006, June 6). OrderGun.A Rootkit Analysis.

Dunham, Ken (2007, April 4). Rootkits Video. Web site: http://www.kendunham.org/rootkit.wmv

Dunham, K. & Doyle, F. (2006, September 6). Malicious Code Year-to-Date Trends. Web site: http://labs.idefense.com/presentations/online/

Dunham, K. & Doyle, F. (2006, August 13). Rootkits and Other concealment Techniques in Malicious Code. Web site: http://labs.idefense.com/presentations/online/

Dunham, Ken (2006). VMWare Disk Mount Tutorial.

Doyle, Fred (2006). BOTs and the Rootkits they Carry.
Crucial Security (2007). Crucial ADS GUI Scanner. Web site: http://www.crucialsecurity.net//index.php?option=com_content&task=view&id=95&Itemid=137

Espiner, Tom (2007, January 19). Swedish bank hit by 'biggest ever' online heist. Web site:

http://news.zdnet.co.uk/security/0,1000000189,39285547,00.htm

F-Secure Corp. (2007). F-Secure Blacklight. Web site: http://www.f-secure.com/blacklight/

F-Secure Corp., (2006, May 27). F-Secure Rootkit Information Pages: Mailbot.AZ. Web site: http://www.f-secure.com/v-descs/mailbot_az.shtml

Florio, Elia (2006, June 29). Raising the Bar: Rustock.A and Advances in Rootkits. Web site: http://www.symantec.com/enterprise/security_response/weblog/2006/06/raising_the_bar_rustocka_advan.html

Foundstone Inc. (2007). The Forensic ToolkitTM v2.0. Web site: http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/proddesc/forensic-toolkit.htm

Heasman, John (2006, November 15). Implementing and Detecting a PCI rootkit.

Heyne, Frank List Alternate Data Streams (LADS) Software. Web site: http://www.heysoft.de

Hoang, Mimi (2006, November 8). Handling Today's Tough Security Threats. Web site: http://www.symantec.com/enterprise/security_response/weblog/2006/11/handling_todays_tough_security_1.html

Hougland, G, & Butler, J (2006). *Rootkits: Subverting the Windows Kernel*.Stoughton, Massachusetts: Addison-Wesley.

James (2007). Using CR0 register in a driver. Web site: http://www.tutorials-blog.com/nt/Using-register/

Kdm (2007). NTIllusion: A portable Win32 userland rootkit. Web site: http://www.phreakmail.com/phrack62-3.html

KodeIT (2007). ScanADS command line tool. Web site: http://www.kodeit.org/products/scanads/

Kuster, Robert (2003, August 4). Three Ways To Inject Your Code Into Another Process. Web site: http://www.codeguru.com/Cpp/W-P/system/processesmodules/article.php/c5767/

Martijnc (2007). GMER. Web site: http://martijnc.be/tools/mieeeeeeeeeeeeeeeeep/gmer.htm

McAfee, (2001, July 7). VBS/Potok@MM. Retrieved February 22, 2007, Web site: http://vil.nai.com/vil/content/v_99147.htm

Merijn (2007). ADS Spy GUI Scanner. Web site: http://www.spywareinfo.com/~merijn/downloads.html

Microsoft Corp. (2004, August 4). Control Internet Explorer Add-ons with Add-on Manager. Web site: http://www.microsoft.com/windowsxp/using/web/sp2_addonmanager.mspx

Microsoft Corporation, Hooks. Web site: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/WinUI/WindowsUserInterface/Windowing/Hooks.asp

Microsoft Corp. (2007, January 22). Patching Policy for x64-Based Systems. Web site:

http://www.microsoft.com/whdc/driver/kernel/64bitpatching.mspx

Microsoft Corp. (2007). RootkitRevealer. Web site: http://www.microsoft.com/technet/sysinternals/utilities/RootkitRevealer.mspx

Microsoft Corp. (2007). Streams. Web site: http://www.microsoft.com/technet/sysinternals/utilities/streams.mspx

Monti, Paolo (2007). GhostHunter Beta. Web site: paolo.monti@effetime.it

Parc Data Systems (2007). Shredator. Web site: http://www.shredator.com/commondownloadpage.asp

Pjf USTC (2007). IceSword 1.20en Public Version. Web site: http://202.38.64.10/%7Ejfpan/download/IceSword120_en.zip

Resplendence (2007). RootKit Hook Analyzer. Web site: http://www.resplendence.com/hookanalyzer

RK Unhooker (2007). RK Unhooker. Web site: http://rkunhooker.narod.ru/

Rutkowska, Joanna (2006, December 28). Stealth malware – Towards Verifiable OSes. Web site: http://invisiblethings.org/papers/towards_verifiable_systems.ppt

Safer Networking Limited (2007). ADS Locator. Web site: http://www.safer-networking.org/en/tools/tools_ads.html

SANS, (2007). Glossary of Terms Used in Security and Intrusion Detection. Web site: http://www.sans.org/resources/glossary.php

SecuriTeam, KDM (2005). Analysis of a win32 Userland Rootkit. Web site: http://www.securiteam.com/securityreviews/5FP0E0AGAC.html

Shelton, Tim (2006). Rootkit Basics. Web site: http://dc214.unspecific.com/notes/rootkit_basics.ppt

Sophos Plc, (2003, October 21). Troj/CoreFloo-C. Retrieved February 22, 2007, Web site: http://www.sophos.com/virusinfo/analyses/trojcoreflooc.html

Tutorials-blog.com, I can not get the right symbol files! Web site: http://www.tutorials-blog.com/nt/nt-8.html

Sophos Plc. (2006, April 3). W32/Feebs-Q. Web site: http://www.sophos.com/virusinfo/analyses/w32feebsq.html

Sophos Plc. (2006, September 8). Troj/Goldun-EA. Web site: http://www.sophos.com/security/analyses/trojgoldunea.html

Sophos Plc. (2006, October 16). Trojan/Haxdoor-DH. Web site: http://www.sophos.com/security/analyses/trojhaxdoordh.html

Sophos Plc. (2005, November 18). Troj/Lecna-F. Web site: http://www.sophos.com/virusinfo/analyses/trojlecnaf.html

Sophos Plc. (2007). Sophos Anti-Rootkit. Web site: http://www.sophos.com/products/free-tools/sophos-anti-rootkit.html

Symantec Corp., (2003, August 16). W32.Dumaru@mm. Web site: http://securityresponse1.symantec.com/sarc/sarc.nsf/html/w32.dumaru@mm.html

Symantec Corp. (2006, July 5). Backdoor.Rustock.B. Web site: http://www.symantec.com/security_response/writeup.jsp?docid=2006-070513-1305-99&tabid=1

Symantec Corp., (2006, June). Raising the Bar: Rustock.A and Advances in Rootkits. Web site: http://www.symantec.com/enterprise/security_response/weblog/2006/06/raising_the_bar_rustocka_advan.html

VeriSign iDefense (2006, December). Major Threats and Trends Impacting the 2007 Cyber Security Landscape. Web site: http://labs.idefense.com/rss/intelligence.rss.php?type=researchreports

VMWare Inc., VMWare Disk Mount. Web site: http://www.vmware.com/download/ws/drivers_tools.html

Wikipedia.org, (2007). Browser Helper Object. Web site: http://en.wikipedia.org/wiki/Browser_Helper_Object

Wikipedia.org, (2007). Rootkit. Web site: http://en.wikipedia.org/wiki/Rootkit