Rootkits:
Risks, Issues and Prevention.

Martin Overton, IBM Global Services, UK

Email: *overtonm@uk.ibm.com*
WWW: *http://www.ibm.com/uk*

Tel: *+44 (0) 2392 563442*

Abstract:

Rootkits have been around almost since the start of computing, however over the last two years the threat has changed; no longer is it just a *NIX [Unix/Linux] problem, corporate and academic computers running Microsoft Windows are now an increasing target. We are now at a tipping point; rootkits are no longer a minor annoyance or threat, they are starting to become a major cause for concern.

Many corporate security staff have a rather vague understanding of rootkits, not just what they are, but how they work. Furthermore many have little understanding of the risks to their company or their own home computer.

This paper will explain what rootkits are and how they work. It will also discuss ways to combat them using methods that range from simple security methodologies through to technical solutions.

Disclaimer:
Products or services mentioned in this paper are included for information only. Products and/or services listed, mentioned or referenced in any way do not constitute any form of recommendation or endorsement by IBM or the papers author.

This paper was written for, and presented at, the 2006 Virus Bulletin conference at the Fairmont Queen Elizabeth Hotel, Montreal, Canada between October 11th – 13th 2006.

I would welcome any constructive feedback on this paper and its content.

1. Introduction

This paper has been written for the corporate stream of the conference and therefore it will not delve into very technical details of rootkits and stealth techniques. However, links will be used [where possible] to point the reader to more details on a topic.

Before we get stuck in and look at the relevant areas of concern and interest with regard to rootkits and stealth techniques used by an increasing number of malware, let us quickly cover some terms that will be used throughout the paper. This will hopefully ensure that all readers be they; layman, technician or researcher will be all be able to get something useful from it. Other terms will be defined as required elsewhere in the paper.

To further aid those readers that have a limited knowledge of rootkits and stealth techniques, I will then present a short history of rootkits and stealth techniques and how they evolved to what we see today. Basically we will cover both sides of the same coin; heads [rootkits] and tails [stealth]!

1.1 Definitions:

1.1.1 Rootkit [old]

Here is a definition from Wikipedia[1]:

> "*The term "rootkit" (also written as "root kit") originally referred to a set of recompiled Unix tools such as "ps", "netstat", "w" and "passwd" that would carefully hide any trace of the intruder that those commands would normally display, thus allowing the intruders to maintain "root" on the system without the system administrator even seeing them*."

My own preferred description of what rootkits are appears below:

> "*A rootkit is a collection of tools an intruder brings along to a victim computer after gaining initial access, usually via hacking into the box manually or by getting the a user to execute a Trojan or Worm which will install a backdoor for them to slither onto the system in the first place. A rootkit generally contains network sniffers, log-cleaning scripts, and trojaned replacements of core system utilities. There is however another type which doesn't tend to replace system files, these are: Kernel [LKM] rootkits which subvert the system by attaching themselves to, or by otherwise modifying the kernel of the targeted operating system.*
> *Some examples of such kernel rootkits on Linux include: Knark, Adore, and Rtkit.*[2]"

1.1.2 Rootkit [new]

Here is another definition of Rootkit from Wikipedia:

> "*A set of software tools frequently used by a third party (usually an intruder) after gaining access to a computer system. These tools are intended to conceal running processes, files or system data that helps an intruder to maintain access to a system without the user's knowledge*"

My own preferred description of what modern rootkits are appears below:

> "*A rootkit is a 'stealth' toolkit, library or program code which is used by malware to hide or 'cloak' files, directories and processes used by the malware. The operating system and applications running on the operating system will not be able to see [read/write/execute] files 'stealthed' in this manner when called or otherwise accessed via 'stealthed' [filtered] system calls and APIs, this includes many anti-virus, anti-spyware and personal firewall applications. In conclusion modern rootkits, especially those for Wintel are not rootkits in the traditional sense, and should be called 'stealthkits' where the functionality is external to the malware, and called 'stealth' where the functionality is part of the malware itself.*
> *Some examples of Wintel rootkits include: Hacker Defender, FU and Vanquish.*"

[1] Source: http://en.wikipedia.org/wiki/Rootkit
[2] Source: http://momusings.blogsome.com/2005/04/12/rootkits-revealed/

1.1.3 Stealth

So, now that we know that modern rootkits are really stealthkits or program code or modules to effectively give malware stealth-like abilities to allow it to hide its presence. Just what is 'stealth'?

Here is a definition from Wiktionary[3]:

> ***"Noun***
>
> *stealth*
>
> *The attribute or characteristic of acting in secrecy, or in such a way that the actions are unnoticed or difficult to detect by others.*
>
> ***Adjective***
>
> *stealth*
>
> *Having the ability to not be detected"*

Here is a definition of malware 'stealth' from Wikipedia[4]:

> "*Some viruses try to trick anti-virus software by intercepting its requests to the operating system. A virus can hide itself by intercepting the anti-virus software's request to read the file and passing the request to the virus, instead of the OS. The virus can then return an uninfected version of the file to the anti-virus software, so that it seems that the file is "clean". Modern anti-virus software employs various techniques to counter stealth mechanisms of viruses. The only completely reliable method to avoid stealth is to boot from a medium that is known to be clean.*"

In the realm of malware 'stealth' is not new, in fact the very first malware written for the IBM PC [and clones] used 'stealth' to hide its presence[5]: The name of this malware is Brain[6].

Here is a short extract from the description of Brain from F-Secure explaining how the stealth function it used works:

> "*The Brain virus tries to hide from detection by hooking into INT 13. When an attempt is made to read an infected boot sector, Brain will just show you the original boot sector instead. This means that if you look at the boot sector using DEBUG or any similar program, everything will look normal, if the virus is active in memory. This means the virus is the first "stealth" virus as well.*"

So, what is the difference between rootkits and stealth?

When we are talking about modern rootkits then my own belief is:

> *Rootkit == Stealthkit == Stealth*

For the purpose of the rest of this paper, please see rootkits and stealthkits [or stealth] as the same when I am discussing *modern* rootkits. However, just to make the paper complete the next section will briefly cover the history of rootkits; from traditional through to the modern ones which I prefer to call 'stealthkits' or 'stealth'. I will then briefly cover a short history of stealth in relation to malware.

---

[3] Source: http://en.wiktionary.org/wiki/stealth
[4] Source: http://en.wikipedia.org/wiki/Computer_virus#Stealth
[5] Source : http://www.research.ibm.com/antivirus/timeline.htm
[6] More data can be found here : http://www.f-secure.com/v-descs/brain.shtml

1.2 A Very Short History of Rootkits – Heads

Back in the late 1980s rootkits started to appear for UNIX based platforms. These were collections of previously disparate tools, many of which had existed for many years, such as backdoors, log filters, sniffers and trojan system binaries. They were in some cases just assembled from these individual components and supplied with some form of install script or instructions on how to install the 'bits' which made up the early kits.

The main functions that 'real' rootkits were created for back in the 1980s were:

- Maintain access – after you had already 'hacked' into a system, usually by installing a backdoor.
- Hide your presence – by altering, replacing or destroying files, such as replacing system binaries, removing log entries and so on.

These early 'Heath Robinson' types of kits were being replaced by more professional and better integrated ones during the middle to late 1990s. Not only that but the functionality of these new breed of rootkits were being expanded to include:

- Attacking [compromise and DoS] of other systems.
- Data mining/capture tools.

Up to this point all of the rootkits had been of the application level type; replaced [trojanised] system binaries were used. Towards the end of the 1990s new methods, such as LKM [Loadable Kernel Module] rootkits and library rootkits started to appear for *NIX systems, including Linux. Some early work was also done on kernel level rootkits which modified the system kernel without the use of LKM techniques.

The edge of the new Millennium brought even more complex and enhanced rootkits to the *NIX world. These new ones included new features yet again, such as:

- Sniffers and protocol decoders[7].

Packet sniffing tools and protocol decoders are easily used and very effective, especially those that have been written to 'decode' password data.

1.2.1 The Next Generation

Not only was the *NIX[8] world still a target but also in 1999 an article written by Greg Hoglund was published in Phrack 55[9] on the possibility of a Windows NT Kernel rootkit. He discusses a 4 byte patch which, as he describes "*removes ALL security restrictions from objects within the NT domain. If this patch were applied to a running PDC, the entire domain's integrity would be violated.*" From this article he went on to develop, as he advertises it: "*The original and first public NT ROOTKIT*[10]". Pandora's Windows box was now open!

This milestone was followed by another in 2003 when a paper was published on the FU rootkit which uses what it describes as 'Direct Kernel Object Manipulation (DKOM)'. This step is described by some as the third generation of 'rootkit' techniques. FU is described thus by its author: "*The FU rootkit can hide processes, elevate process privileges, fake out the Windows Event Viewer so that forensics is impossible*". The latest version can "*even hide device drivers*".

F-Secure describe FU as:

[7] Used for finding paaswords, sensitive data and logon credentials, to be later mis-used.
[8] A good history of *NIX rootkits has been written by iDefense, entitled: *An Overview of UNIX Rootkit*s which can downloaded from here:
http://www.rootsecure.net/content/downloads/pdf/unix_rootkits_overview.pdf
[9] The article can be found here: http://www.phrack.org/phrack/55/P55-05
[10] Source: http://www.rootkit.com/project.php?id=11

> "*Fu is one of the most widely utilized rootkits in the wild. Other malware, such as rbot and sdbot variants, have used its features to hide themselves. Fu is a kernel-mode rootkit that modifies kernel data structures, which allows it to hide e.g. processes*[11]."

However, in all these cases so far, these rootkits are separate to any malware that may make use of them to enable them to become 'stealthed'.

This scenario started to change as so-called 'rootkit' functionality [stealth] was added to malware binaries at the compile stage [as a module, library or other code block]. Malware authors had decided to re-discover[12] 'stealth' and use it in their Windows creations.

In 2005, a surprising finding suddenly catapulted 'rootkits' back into the news from their relative obscurity of the last two decades. This finding would start a media sensation, law suits and lots of red faces at a large multi-national company. Who was this company? For those of you who slept through the last quarter of 2005, it was Sony.

Yes, Sony was found to be using 'rootkit' techniques on a number of their Music CDs. The technology was supplied by First 4 Internet, known as XCP (Extended Copy Protection). This technology was used by Sony for the purpose of Digital Rights Management [DRM]. I am not going to go through the rights and wrongs of the Sony case as you can find plenty of other analysis elsewhere. Here are a few links to get you started:

- http://www.sysinternals.com/blog/2005/10/sony-rootkits-and-digital-rights.html
- http://en.wikipedia.org/wiki/2005_Sony_CD_copy_protection_controversy

My only comment about the Sony case is that the problem was that the technology they used was flawed as it allowed other files, not from Sony or First 4 Internet to use the technique to hide from security tools and the Operating System. That was a very bad mistake to make, aside from the ethical dimension.

Not only were Sony caught using so-called 'rootkit' techniques, so were Symantec as they hide files using a similar technique in one of their products[13]. Kaspersky were also chastised for using a different technique to hide data; this being the use of the ADS[14] feature of Windows NTFS, which I will cover later in this paper.

1.3 A Very Short History of Stealth – Tails

Back in the dawn of malware for the IBM PC and clones we saw the birth of malware for the platform with Brain, as previously covered in the last section, stealth techniques were used by Brain to try and stop the malware from being discovered and neutralised.

Brain is a boot sector virus, which infects when a floppy disk [5.25"[15]] is left in the drive of a PC and the PC attempts to boot from it. At this point Brain is memory resident; stealing 7KB and if a hard drive is installed then the partition sector of the HD becomes infected. All other floppy disks used in the infected system that are not write protected will also become infected by Brain.

It took until 1989 until we saw the same level of development in stealth for file viruses. The first full-stealth file virus was Frodo[16].

Here's a snippet from the description of Frodo from F-Secure:

[11] Source: http://www.f-secure.com/v-descs/fu.shtml
[12] Stealth has been used in a small number of Windows malware since 1997 in Cabanas [semi-stealth] and Maslan.A in 2004 [There are many DOS viruses which used stealth techniques to hide from the Operating System and Anti-Virus tools]
[13] The Symantec product was Norton SystemWorks. The actual component involved was NProtect.
[14] ADS stands for Alternate Data Streams. This will be covered later in the paper.
[15] 360KB capacity storage media.
[16] Source: http://www.virus-scan-software.com/virus-scan-help/answers/the-history-of-computer-viruses.shtml

> "*The Frodo virus is one of the most advanced "stealth" viruses, and is quite successful in hiding from detection. If it is active in memory and you look at the directory, the virus will show you the original length of any infected program. In addition, it will intercept most attempts to read infected files, so only the original, non-infected file will be seen.*"

From here on [1989] stealth became a common and increasingly a must-have feature for any self-respecting malware by its author.

Then Windows 95B appeared bringing a new file system [FAT32] with it, and things changed…

1.3.1 The Next Generation [Take 2]

FAT32 arrived and caused lots of problems not only for malware but also for anti-virus programs too. The problem was so large and lots of misinformation and myths were flying around about FAT32 and its mystical properties that I decided to write a paper for Virus Bulletin '97 to address the problem.

The paper was entitled: *'FAT32 - a new problem for anti-virus or viruses?*[17]'. The paper covers the impact of FAT32 (Part of Windows 95B/98) on computer viruses and anti-virus software. The paper included test results of a number of file and boot viruses, many of them included used stealth techniques and the results were intriguing as many of the viruses that used medium or advanced stealth would no longer work under Windows 95B or Windows 98. This included the much vaunted 'Frodo'.

So, was this the end of stealth, at least with regard to Windows?

For many years it seemed so. Stealth techniques did seem to be ignored for Windows by malware authors, with the obvious exception of Cabanas that is. However, malware that uses 'stealth' techniques are now commonly referred to as using 'rootkit' techniques or functionality. The first example of this new wave of malware using 'rootkit' techniques as part of its own code is Maslan.A. However, it should be called 'stealth' as that is what it achieves.

Will we see a similar scenario when Vista starts to become the dominant versions of Windows? Only time will tell.

2 The Size of the Problem

Right, now you know about rootkits and stealth and the history behind both and the current misnaming of stealth as rootkit when discussing modern malware that uses techniques to hide from the Operating System and all applications running on the OS. Hopefully you will agree that what we are really talking about is the next generation of 'stealth' or 'Stealth:TNG' if you prefer. Let us use that name for the purpose of discussing what most security people are currently calling 'rootkit techniques'.

So, how big a problem is 'Stealth:TNG'?

A recent study published in Australia by AUSCERT [2006 Australian Computer Crime and Security Survey] claims that[18]:

- "*About 1 in 5 reported trojan or rootkit infections,which is considered to be high given that such malware cannot self-propagate.*"
- "*Of those that suffered trojan or rootkit infections,most were from public sector organisations (60%) compared to the private sector (40%).*"
- *"2005 saw a marked increase in the use of rootkits on the Windows platform, blending with more traditional malware such as trojans to increase their life expectancy and this trend continues into 2006. AusCERT has observed an increasing number of online ID theft trojans using rootkit functionality."*

[17] It can be downloaded from here : http://arachnid.homeip.net/papers/VB97-FAT32.pdf
[18] Source: http://www.auscert.org.au/images/ACCSS2006.pdf

I think that they missed the reason for the jump in malware using 'Stealth:TNG' [aka Rootkit functionality]. The reason is that the malware authors are using this technology irrespective of the type of malware, be it Trojan, spyware, bots or worms. In the case of worms and many bots they are self-propagating, or at least human assisted via social engineering.

The bad guys, be they malware authors 'doing it for fun' or cyber criminals doing it for money are actively using 'Stealth:TNG' to make their wares harder to detect. We will almost certainly see a re-run of the 'stealth' wars we saw back in the 1980s and 1990s between the malware authors and the security community. This war; from the 'good guys' side should be led by the anti-virus companies as they already have experience in battling and countering 'stealth' that they can leverage to protect their customers.

Furthermore, McAfee also published the first part of a white paper on rootkits[19], and in it they came up with some interesting statistics on the size and complexity of the problem. They claim:

- "*In just three short years, the use of stealth techniques in malicious software (malware) has grown by more than 600 percent.*"
- "*From 2000 to 2005, rootkit complexity grew by more than 400 percent, and year-over-year, Q1 2005 to 2006, complexity has grown by more than 900 percent.*"
- "*The share of Linux-based techniques has gone from a high of roughly 71 percent of all malware stealth components in 2001 to a negligible number in 2005, while the number of Windows-based stealth components has increased by 2,300 percent in the same period.*"

3 Risks

Risk is a difficult area to cover. Most of us understand what constitutes a risk, however everyone's perspective on risk is different; one person's acceptable risk [such as mountaineering, motorcycle racing, bungee jumping, or keeping venomous animals (snakes, tarantulas and/or scorpions)] is another person's unacceptable risk. So, I will cover the areas of risk assuming that we are dealing with a company/institution/person who is inherently risk sensitive or averse.

So, let us for clarities sake look at each major area of risk that 'Stealth:TNG' specifically brings to the table; a relative feast awaits us.

Please note that many of the areas of risk are the same for many classes of malware and related security threats. I have covered many of these in my earlier papers on 'Bots and Botnets' and 'Spyware: Risks, Issues and Prevention'. I would suggest that you read these as I will only cover newer or specific risks that apply to rootkits, stealthkits and Stealth:TNG in this section[20].

3.1 Loss of Ownership

This risk is much the same irrespective of what malware or security breach has taken place on one or more systems in your organisation. Once your security is breached and someone else has their code or remote access to your computers, these computers no longer belong to you, they now belong to the bad guys and girls. Rootkits, Stealthkits and Stealth:TNG just make the job of detecting the security breach and misuse of the infected/affected systems that much harder.

3.2 Seeing The Invisible

This is the main problem with malware which either uses an existing [or new] stealthkit or includes Stealth:TNG in the code of the malware itself.

The whole purpose of these techniques and tools are to make the malware invisible [at least partially] to the Operating System and more importantly, from the malware authors perspective, any security

[19] The paper is entitled 'Rootkits, Part 1 of 3: The Growing Threat. It can be found here: http://download.nai.com/products/mcafee-avert/WhitePapers/AKapoor_Rootkits1.pdf
[20] All my published papers and articles can be found here: http://arachnid.homeip.net/papers/

software which runs on top of it. This includes anti-virus, anti-spyware, personal firewall and last but not least anti-rootkit tools.

Reliably detecting malware is not a trivial task, with the exception of the simplest malware or those that are 'in-your-face' in that they make it obvious that they have infected your system, such as with obvious payloads, such as: pop-ups, mass-mailing, fast infecting, data corruption or encryption, file content modification or deletion, and so on.

However most malware will, to achieve it's ends will give itself away, be it that it modifies or replaces files, or make a companion of an existing file, network activity, such as with phone-home techniques, mass-mailing, registry modification, memory usage, cpu usage, and so on. However, most 'normal' applications do this too, so how do you detect the bad without detecting the 'good'. In the case of Stealth:TNG and Stealthkits, the answer is obvious, these techniques and tools try to hide, so to detect their presence we need to find out that something is being hidden. A number of techniques can be used to do this. You can also make sure that the malcode using such techniques isn't running; if it isn't running it can't hide – it is visible.

I will cover some of the ways that this can be achieved in the section on 'Solutions' later in this paper.

3.3 He Who Runs First wins

As hinted at above, the problem with malware that is running on a system is that is may be able to hide its presence because it is running, and because of that it can subvert the Operating System and any software that is running on the Operating System.

This means that malware which uses Stealth:TNG which gets to run first [before the security software] is effectively invisible from not only the Operating System but also to all software which runs on it.

So, if the security software gets loaded first and can detect the malware , it probably will and stop it from being able to run and install itself.

This is the state of play when we are discussing known malware, the rules may well be different if we are discussing new malware which is not yet in the detection database for the anti-malware product. However, all is not lost, there is a chance that other 'generic' techniques such as heuristics, emulation, sandboxing or behaviour blocking may still be able to detect it and stop it.

I will cover some of the ways that this can be achieved in the section on 'Solutions' later in this paper.

3.4 Loss of Confidence

If a malware infected system is traced back to 'your' company [organisation or establishment], what would be the impact on 'your' companies credibility? What about a loss of confidence from other companies that you partner with, what about the potential loss of faith by your customers?

These are real concerns and they shouldn't be underestimated. In this case there is such a thing as bad publicity no matter who tells you otherwise.

3.5 Other Tricks

3.5.1 ADS

What is ADS [Alternate Data Streams]?

Here is a definition from 'FAQ: Alternate Data Streams in NTFS[21]'

"In NTFS, a file consists of different data streams. One stream holds the security information (access rights and such things), another one holds the "real data" you expect to be in a file. There may be another stream with link information instead of the real data stream, if the file actually is a link. And

[21] Source: http://www.heysoft.de/nt/ntfs-ads.htm

there may be alternate data streams, holding data the same way the standard data stream does."

How do you make and use ADS?

Simple really, you can do it from the command prompt. Let's say that you want to add an ADS to `Notepad.exe`. This is how you would do it:

```
Type malware.exe > notepad.exe:malware.exe
```

If you want to now execute the file in the ADS of `Notepad.exe` from Windows XP you would do it like this:

```
start .\notepad.exe:malware.exe
```

The host file doesn't have to be an executable, you can use any file type you like and even directories to serve as the host for ADS hidden files.

This will even work for files hosted on a web server even if they are remotely accessed. The bad guys could hide malware in what looks like a HTML or other text file and call that file hidden in the ADS in this way:

```
http://myserver.nosuchdomain.com/index.htm:malware.exe
```

As you can see the possibilities of misuse are quite large and somewhat worrying. In fact the first malware to use this technique is known as Virus.Win32.Stream.a[22]. There are also modern malware which use ADS along with other 'Stealth:TNG' techniques, an example of this is Mailbot.AZ[23] (aka Rustock.A).

The good news is that most anti-virus programs can now detect malcode hidden using ADS.

3.5.2 The Human Element

I've written about the 'Human Element' in a number of previous papers and articles for Virus Bulletin [and others]. This is probably the hardest area to deal with. Not only with those wonderful people that like to 'double-click' or 'double-click-and-enter-the-password-and-double-click-again' on attachments they receive; to infect their computers, but also those that are more 'security-savvy' such as administrators, support staff, and so on.

Why am I covering this here? Well, unfortunately there will be some [hopefully a small number] of people in your organisation that will:

- Want to play with new malware and malware creation tools. '*Just to see if it works*', of course.
- Be there to carry out industrial espionage.
- Are disgruntled, because they believe they are underpaid, put upon, under-valued, being bullied or are going to be, or have been sacked.

These individuals may well want to create and plant a rootkit [or other malware] on your systems as a form of revenge or to make money from selling your secrets. However, most of them will not have the ability or time, to learn the skills required to write a rootkit or other stealth malware. Never fear, someone has created a 'point-and-click' rootkit creation kit for them.

In other words, you don't have to be a rocket-scientist or a professional programmer to create a rootkit or stealth malware.

[22] A description of Stream can be found here: http://www.viruslist.com/en/viruslist.html?id=4078
[23] A good description of this can be found here: http://www.f-secure.com/v-descs/mailbot_az.shtml

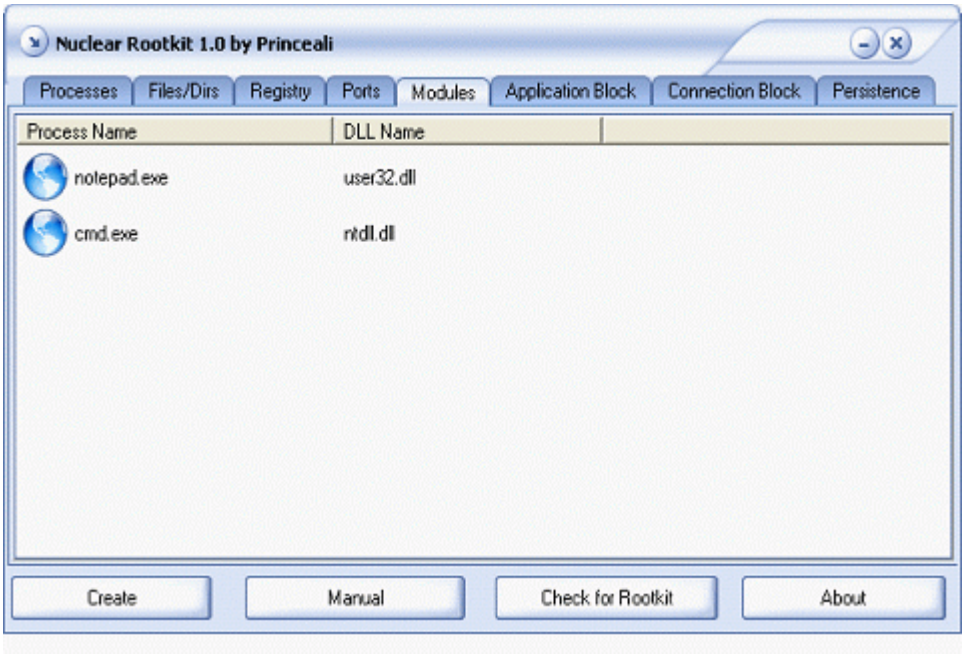


Figure 1a: Nuclear Rootkit Creator Creation Screenshot

You can see in figure 1a the main screen for the 'Nuclear Rootkit 1.0' creation tools. The shown tab is for the modules. You can clearly see the other tabs that allow you to configure the behaviour, such as which files/directories to hide, processes to hide, ports to hide and so on. So, we have configured the behaviours we want and now click on the 'Create' button. If everything goes well we will be presented with the following dialog box.

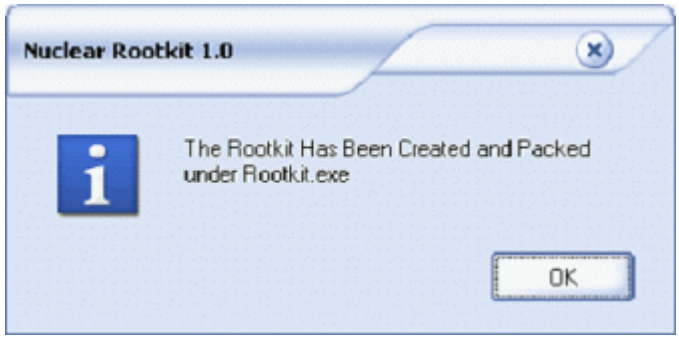


Figure 1b: Nuclear Rootkit Creator Completed Dialog Screenshot

Now we have our new rootkit, ready to be renamed and deployed as the bad guy or girl sees fit.

4 How they work

This section will cover how 'real' rootkits, stealthkits and stealth techniques work. Please do not consider this a full, exhaustive, discussion on these techniques and technologies as that is beyond the scope of this paper.

4.1 'Real' Rootkits

The main purpose of a traditional *NIX rootkit is to allow intruders to come back to the compromised system later and access it without being detected. A rootkit makes this very easy by installing a backdoor remote-access trojan [daemon or application aka a RAT].

Application level rootkits come with modified system binaries that replace the existing ones on the target system. Why do they use modified system binaries? To hide their presence and the actions of those who are using the rootkit of course. In other words, to cover their tracks…..out of sight is out of mind!

Technique Used	Tools/Malware Using This Technique
Patching or replacing binaries on disk	Early Unix rootkits

Table 1: Techniques used by 'real' rootkits

The screenshot below shows a '`ps`' [Process Listing] from a Linux system. The list on the left shows a compromised system which been 'rooted' but the 'rootkits' Trojanised system binaries have not yet been installed. You can clearly see all processes, including '`nc`[24]' and several other 'shells'.

Now, compare this against the '`ps`' output on the right. You can see that those processes are now missing as the '`ps`' binary have been replaced by a Trojanised copy which is designed to hide certain pre-defined processes.

PS Not Trojaned

```
PID TTY          TIME CMD
    1 ?        00:00:00 init
    4 ?        00:00:00 events/0
   94 ?        00:00:00 pdflush
  929 ?        00:00:00 udevd
 2663 ?        00:00:00 portmap
 2687 ?        00:00:00 syslogd
 2793 ?        00:00:00 rpc.statd
 3607 ?        00:00:00 nfsd
 3608 ?        00:00:00 lockd
 3609 ?        00:00:00 rpciod/0
 3619 ?        00:00:00 rpc.mountd
 3740 ?        00:00:00 crond
 3776 ?        00:00:00 smbd
 3777 ?        00:00:00 nmbd
 3820 ?        00:00:00 lisa
 3887 ?        00:00:00 login
 3888 tty2     00:00:00 bash
 3889 tty3     00:00:00 nc
 3891 tty4     00:00:00 mingetty
 3893 tty5     00:00:00 mingetty
 3895 tty6     00:00:00 mingetty
 3956 tty1     00:00:00 bash
 5583 pts/1    00:00:00 bash
 8992 pts/1    00:00:00 su
 8995 pts/1    00:00:00 bash
 9037 pts/1    00:00:00 ps
```

PS Trojaned

```
PID TTY          TIME CMD
    1 ?        00:00:00 init
    4 ?        00:00:00 events/0
   94 ?        00:00:00 pdflush
  929 ?        00:00:00 udevd
 2663 ?        00:00:00 portmap
 2687 ?        00:00:00 syslogd
 2793 ?        00:00:00 rpc.statd
 3607 ?        00:00:00 nfsd
 3608 ?        00:00:00 lockd
 3609 ?        00:00:00 rpciod/0
 3619 ?        00:00:00 rpc.mountd
 3740 ?        00:00:00 crond
 3776 ?        00:00:00 smbd
 3777 ?        00:00:00 nmbd
 3820 ?        00:00:00 lisa
 3887 ?        00:00:00 login
 3891 tty4     00:00:00 mingetty
 3893 tty5     00:00:00 mingetty
 3895 tty6     00:00:00 mingetty
 3956 tty1     00:00:00 bash
 5583 pts/1    00:00:00 bash
 8992 pts/1    00:00:00 su
 8995 pts/1    00:00:00 bash
 9037 pts/1    00:00:00 ps
```

Figure 1c: Screenshot Showing Linux PS Listings

As you can clearly see the replacement '`ps`' is working and hiding certain processes as requested.

[24] 'NC' is 'netcat', which is often used as a remote access Trojan [RAT], although it does have many legitimate uses too.

4.2 Stealthkits

These are modern 'rootkits' which enable, either by design or lack of design, files, directories, registry entries, processes and some cases users and ports to be hidden from the Operating System and security tools running on the Operating System.

In this group I would include tools such as:

- Sony [XPC] DRM software.
- FU and FUtoo
- Rdriv
- NTRootkit
- Hacker Defender

It should be mentioned that many of the bot families are being bundled with stealthkits such as FU and Rdriv. I expect that this is a transitory phase between bots [and other malware] without any stealth functionality and bots [and other malware] that will have stealth built in. In other words we will see more malware, including bots using Stealth:TNG. If you want to learn more about bots and botnets then I would suggest that you grab a copy of my VB2005 paper[25] on this subject, as a starting point.

Technique Used	Tools/Malware Using This Technique
Hooking in memory by replacing pointers in handler table	Ntrootkit Sony BMG DRM
Hooking in memory by modifying function entry point	Hacker Defender
In-memory data structure manipulation	FU (2003) by Butler

Table 2: Techniques used by modern rootkits [aka stealthkits]

A typical stealthkit will allow files and folders that have a specific prefix, such as '_root_' to be hidden from the Operating System and most software applications using normal system calls and APIs. You can see this very option in the screenshot below.

Figure 1d: NTRootkit Screenshot

This technique effectively acts as a file/directory/process filter allowing any entries in the system handler table [system API, entry point, data structure] to be hidden or removed before the data is passed back to the program which requested the data.

[25] The paper can be found here: http://arachnid.homeip.net/papers/VB2005-Bots_and_Botnets-1.0.2.pdf

These API filters are usually run as 'User Mode' processes; however the more advanced ones can run at system level [Kernel Mode].

Here is an example:

In all the following examples, green arrows[26] show unfiltered traffic or data and red arrows are used to show filtered/modified traffic and data.

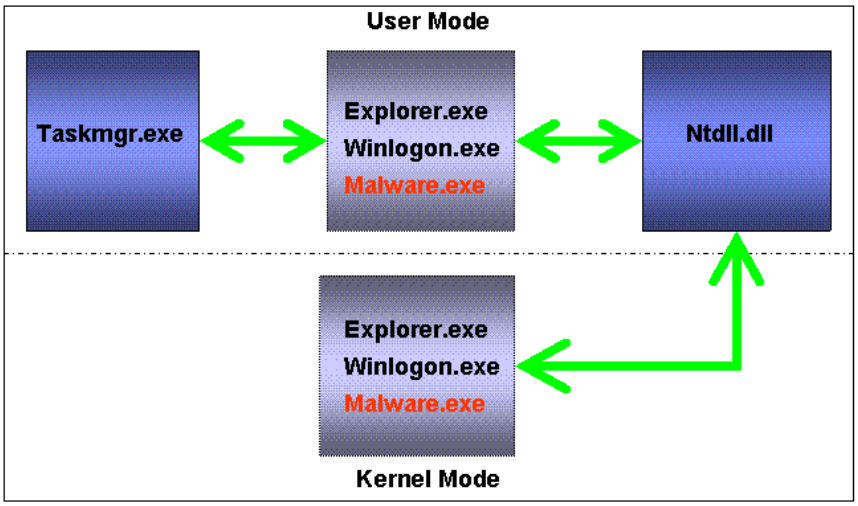


Figure 2: Malware infected system without Stealthkit/Stealth:TNG user mode filter

The above diagram shows the process list of an system infected with a standard worm/virus which is not, at this point, using a Stealthkit or Stealth:TNG functionality.

As you can see task manager [Taskmgr.exe] sees the same process list at both the kernel and user mode levels when using user mode API calls. All traffic and data flow is green; nothing is hidden, including the malware.

Now, let's look at what happens when a Stealthkit is added and told to hide the file and process known as '`Malware.exe`'.

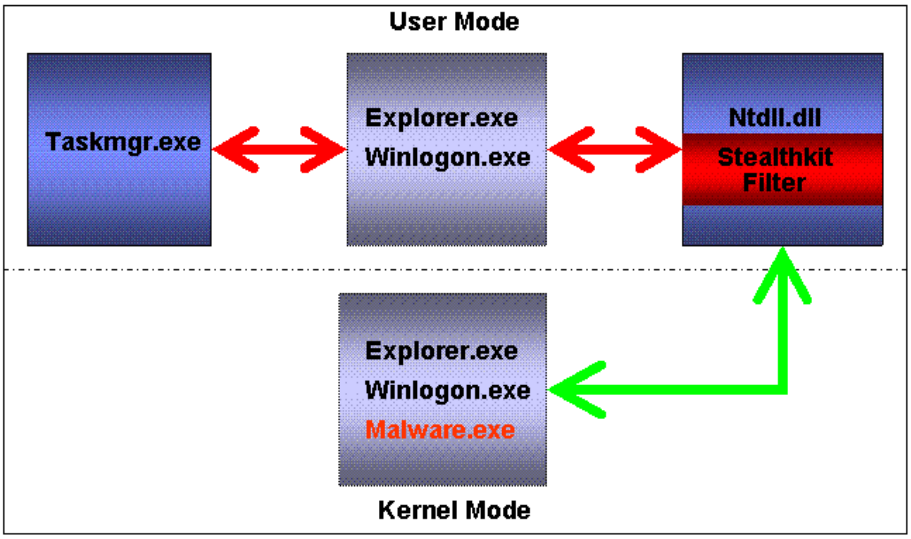


Figure 3: Malware infected system with Stealthkit/Stealth:TNG user mode filter

Now we see that because a Stealthkit has been introduced to the system and configured to hide the file

[26] For those reading in black and white, the green arrows will appear to be light grey and the red arrows will appear to be dark grey.

and process known as '`Malware.exe`'[as well as itself] that some of the data and traffic flow has turned from green to red.

As this Stealthkit is a 'User Mode' one [it runs with the same rights as the logged on user rather than Administrator rights] any requests made by the user will be filtered. This is why the process and file '`Malware.exe`' is not hidden in kernel mode, but it is hidden [filtered] in user mode. In the above example task manager will not be able to see '`Malware.exe`' for the currently logged on user.

Furthermore, many of this class of Stealthkit will not be able to load themselves in 'Safe Mode' or when the 'Administrator' logs on[27]. At this point they are effectively disabled and therefore any previously Stealthed files [including the Stealthkit] are visible and can be detected and removed.

The situation is somewhat different when we look at Kernel Mode Steatlhkits.

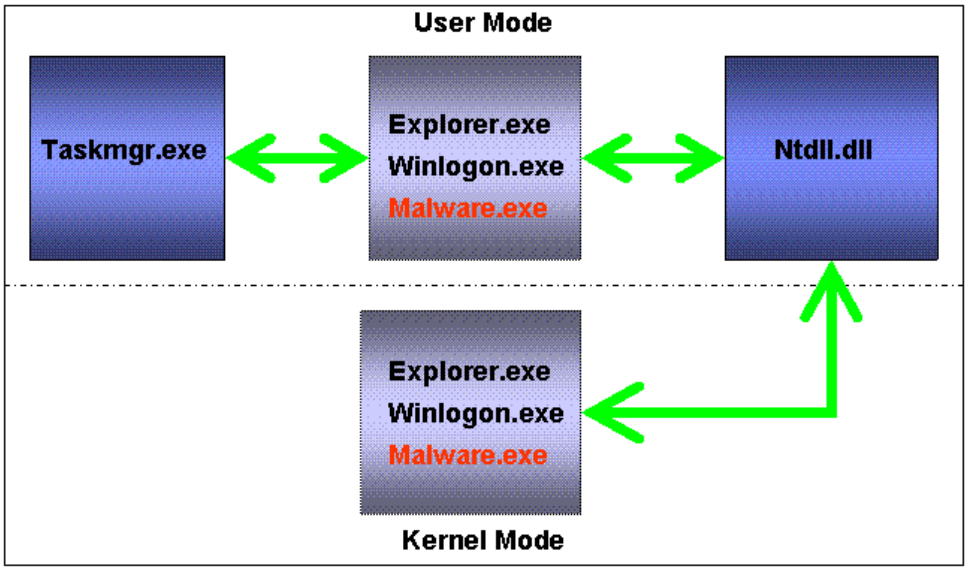
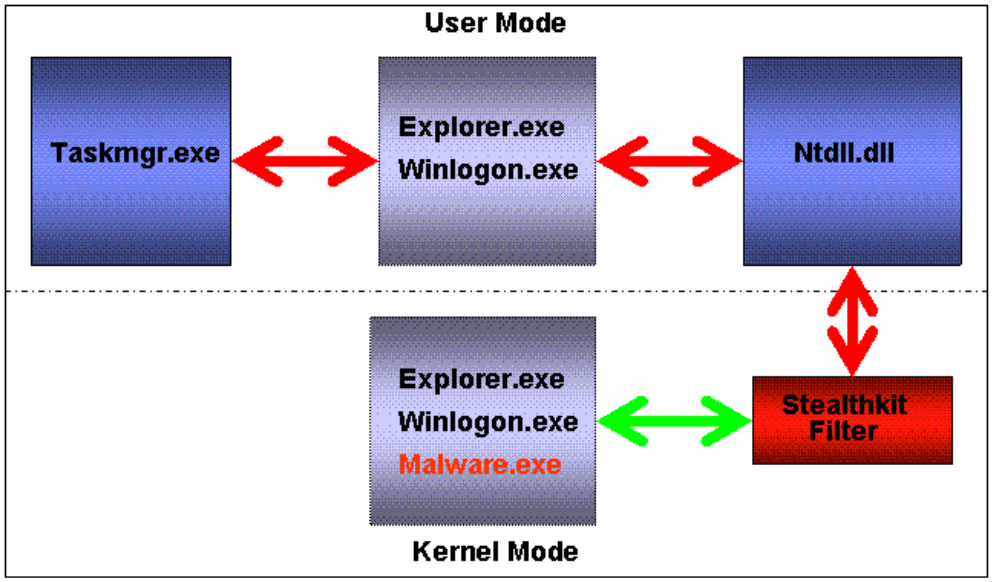


Figure 4: Malware infected system without Stealthkit/Stealth:TNG kernel mode filter

The above diagram shows the process list of an system infected with a standard worm/virus which is not, at this point, using a Stealthkit or Stealth:TNG functionality. So, once more the malware is visible.



[27] Although there are a few examples of User Mode stealthkit/stealth:TNG malware which continue to work even in safe mode.

Figure 5: Malware infected system without Stealthkit/Stealth:TNG user mode filter

Now we see that because a Stealthkit has been introduced to the system and configured to hide the file and process known as '`Malware.exe`' that, as this is a kernel mode Stealthkit, almost all of the data and traffic flow has turned from green to red.

As this Stealthkit is a 'Kernel Mode' one [it runs with the same rights as an Administrator, no matter which user is logged in] any requests made by *any* user will be filtered. This is why the process and file '`Malware.exe`' is now hidden in kernel mode and in user mode. In the above example Task Manager will not be able to see '`Malware.exe`' for any logged on user, even the administrator.

Furthermore, many of this class of Stealthkit may be able to load themselves even in 'Safe Mode'. This makes detection and remediation significantly more difficult than with a user mode Stealthkit.

This class of Stealthkit require administrator privileges to install and are they are not easy to write.

Let me now move on to a more advanced example using a fairly new technique, known as 'Direct Kernel Object Manipulation' or 'In-Memory Data Structure Manipulation'.

As with the previous examples we will start with a diagram showing a system infected with a classic worm or virus with no stealth capabilities.

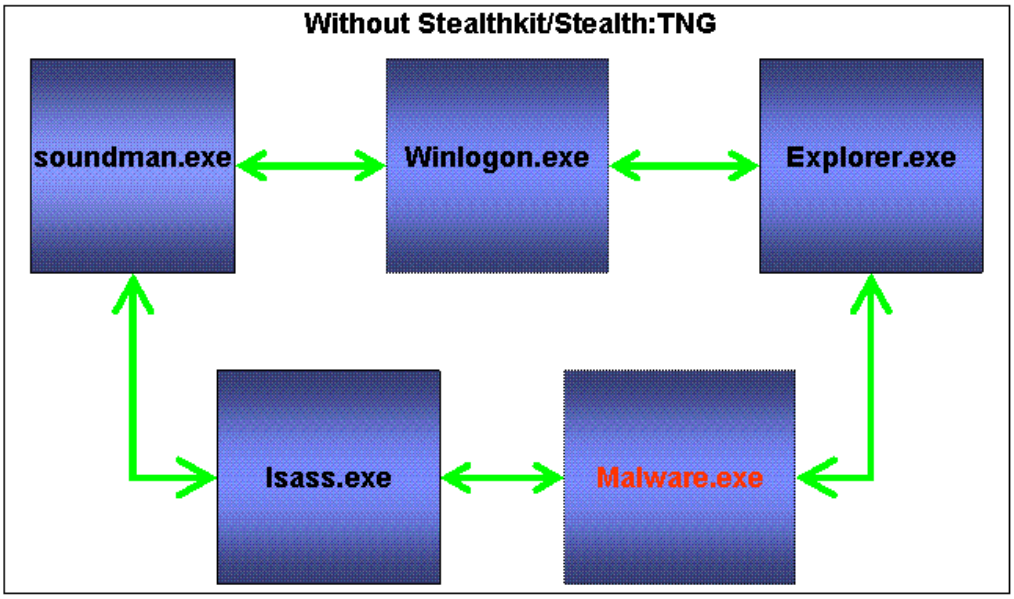


Figure 6: Malware infected system without Stealthkit/Stealth:TNG DKOM filter

As you can see in figure 6 all the data and traffic flow are green which indicates that no data or traffic is being filtered or hidden from the Operating System or programs running on it.

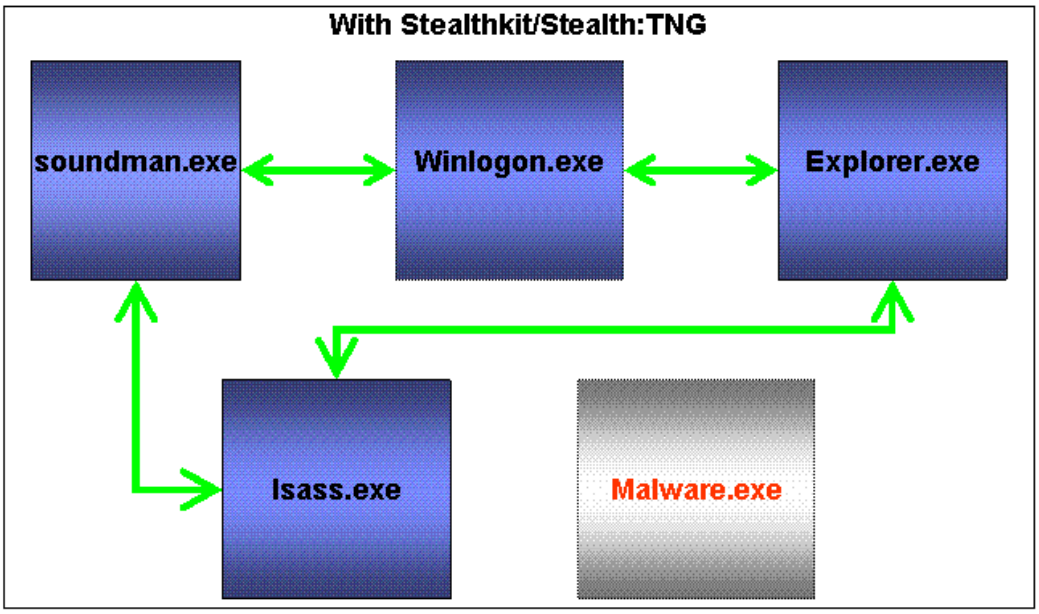


Figure 7: Malware infected system with Stealthkit/Stealth:TNG DKOM filter

However, in figure 7 we see the result of a Stealthkit using 'DKOM' to filter [hide] the '`Malware.exe`' process from the Operating System. The program will still run and also get its fair share of processor time, but as far as the Operating System is concerned it is invisible; effectively become a ghost process.

This class of Stealthkit require administrator privileges to install and they can cause system instability or crashes. This in itself could be used as a tip-off that something has infiltrated the system.

4.3 Stealth:TNG

These are malware which includes so-called 'Rootkit' techniques in the body of the malware itself. In many ways this is really the rebirth of stealth; this time for the Windows Operating System.

The techniques used and included in the body of the malware itself are mostly the same as those used in the previous two sections; Rootkits and Stealthkits. However, there is where the similarity tends to end as these are self-contained malware with no reliance on third-party assistance whatsoever, well apart from the Operating System turning a blind-eye to their presence.

In this group I would include malware such as:

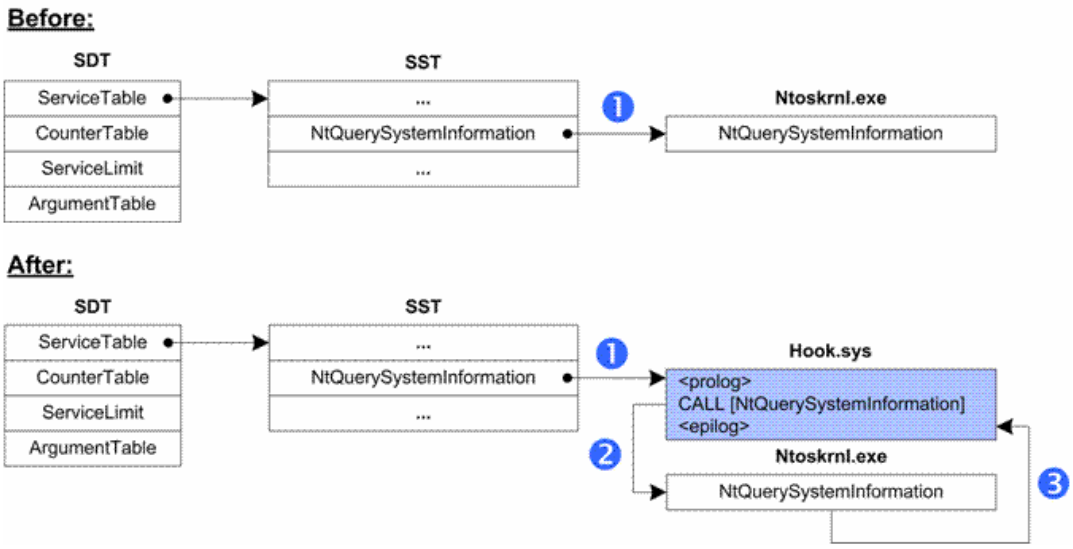
- Cabanas
- Maslan
- Apropros
- Bagle
- Mydoom

Interestingly, as mentioned previously in this paper, Cabanas was the first true Windows NT Virus and what's more is was semi-stealth. It hid the change in file size caused by it infecting a host file, from the Operating System. Not much use for modern malware, but Cabanas was a virus, not a worm, bot or Trojan; the rules of attack and engagement have changed a lot since the days of Cabanas.

Technique Used	Tools/Malware Using This Technique
Hiding in memory by directing execution to a replaced handler	Shadow Walker (2005) by Butler&Sparks, Apropos
Virtual Machine stealth	PMBS (1993) SubVirt by Microsoft Research (2006)

Table 3: Techniques used by Stealth:TNG

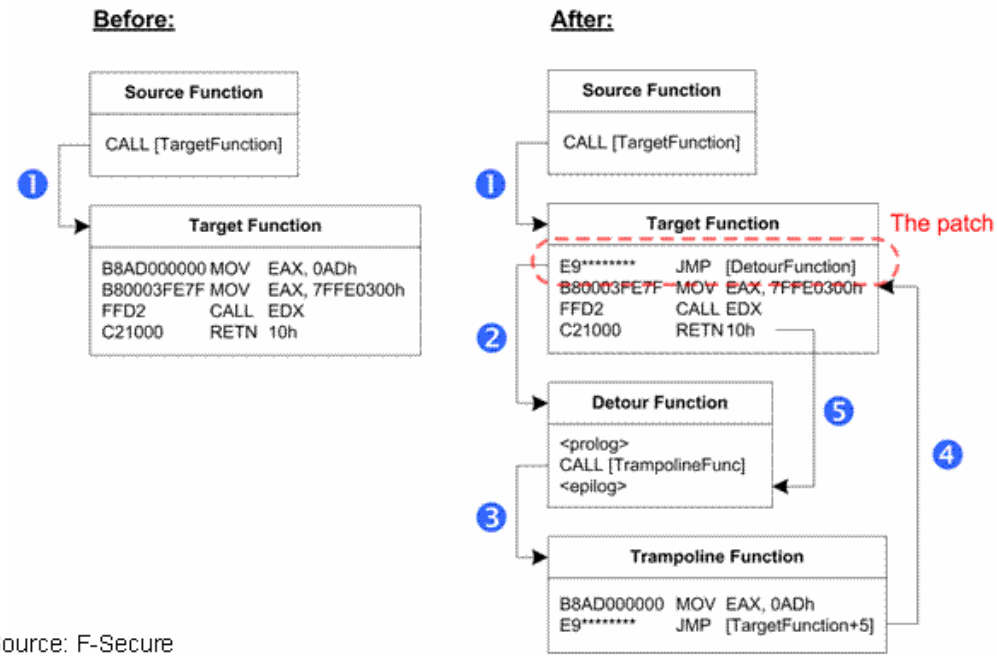
The illustration below shows the different states of a system; clean and infected by malware using a hooking technique known as SST [System Service Table] hooking to hide selected processes from the Operating System.



Source: F-Secure

Figure 8: Malware infected system with Stealthkit/Stealth:TNG using handler table hooking

The next illustration shows the different states of a system; clean and infected by malware using a inline hooking technique to hide selected processes from the Operating System.



Source: F-Secure

Figure 9: Malware infected system with Stealthkit/Stealth:TNG using inline hooking

These are just two of the more advanced methods that can be used. As this is not a technical paper, if you want to know more then I would suggest that you read 'Hide N Seek Revisited – Full Stealth is Back' which was presented at the Virus Bulletin 2005 conference by Kimmo Kaslin, et al from F-Secure.

Back in March of 2006, Microsoft caused a bit of a storm when it published some research on a rootkit

which used 'virtual machine' functionality to hide. They called this rootkit 'SubVirt'. The research was a joint project between Microsoft and the University of Michigan.

I, like many others, are asking why are Microsoft trying to build a rootkit? This seems to me to be somewhat dangerous as it could possibly give the bad girls and guys even more ideas than they already have. Not very smart!

However, one saving grace is that, in this case, the technique they are using isn't really new, as back in the days of DOS stealth viruses this technique was pioneered by a boot virus known as PMBS[28], this was back in 1993, some 13 years later, Microsoft re-discovered the technique.

PMBS when installing itself from and infected disk, copies itself into extended memory and then switches the computer processor into protect mode and runs a virtual V86 machine. DOS and all applications running on it will be run inside the virtual PC owned by PMBS.

This is a common theme as many of the stealth technique that were pioneered in those early DOS stealth days are also being re-discovered, dusted off, re-worked for Windows and deployed. Many of us that have been in the industry since the start of the malware problem have been expecting this and are somewhat surprised just how long it has taken for stealth malware techniques to be migrated to Windows.

No matter, the Stealth arms race has started once more, and it seems that some vendors have been sitting on their laurels rather than ensuring that their customers are protected from this re-emergent threat.

5 Solutions

We have seen how Rootkits, Stealthkits and Stealth:TNG work, what damage they can do and what the cost can be, both financially and in some ways more importantly, damage to you/your company's reputation, brand image or credibility. Let us now look at different ways to combat them using methods which range from simple security methodologies through to technical solutions.

Please note that many of the solutions are the same for many classes of malware and related security threats. I have covered many of these in my earlier papers on 'Bots and Botnets' and 'Spyware: Risks, Issue and Prevention'. I would suggest that you read these as I will only cover newer or specific risks that apply to rootkits, stealthkits and Stealth:TNG in this section[29].

5.1 Generic

What do I mean by generic? Simply this; techniques and/or methodologies which are not specific to rootkits, stealthkits and Stealth:TNG. These may also be applied to other malware and other security issues.

5.1.1 Run as Restricted User

In the *NIX World running as a privileged user is generally considered a 'bad-thing', and quite rightly so, unless you are carrying out administration or other maintenance on a system.

However, in the Windows World it is less common for a user to have a restricted 'user' level account; more often than not they have 'Administrator' [in the *NIX World it would be 'root'] access. Why? Well partly it is down to the way that the Operating System works and also many companies allow their staff to install 'allowed/approved' software on their work computers to keep costs down.

This state of affairs, with regard to Windows users running with privileged rights makes it so much easier for malware to install itself than if they were running with a restricted 'user' account.

If malware requires privileged rights on a system where the current logged-in user has restricted rights, the malware author is going to have to somehow escalate their offspring's rights, usually via using

[28] More details on PMBS can be found here:
http://www.viruslist.com/en/viruses/encyclopedia?virusid=17352
[29] All my published papers and articles can be found here: http://arachnid.homeip.net/papers/

know holes [vulnerabilities] in the operating system, or by trying to brute-force an existing privileged account; such as via a dictionary attack.

So, to make a malware authors job harder, and your administrator job easier, in regard to restricting the penetration of malware on Windows, lock your users down, restrict their rights. You will also be restricting the malware's rights and probably stopping your staff from installing other 'unlicensed' or 'unwanted' software on your machines as a knock-on effect.

This will also make remediation significantly easier, whether it is malware or spyware related.

5.1.2 Education

I commented about the 'human problem' [aka Wetware] in a Virus Bulletin article back in 2002 which stated that "*the overall view of most end-users that security is an IT issue, and therefore not their problem. They seem to think that the technology will save them, what they really need to understand is that they are part of the problem, and are currently exacerbating it.*"

Education is important, but for most staff a simple security policy and acceptable use policy will be more effective than trying to educate them about all the types of risks out there on the internet. Instead focus on your support and technical staff, as they will probably be more interested and likely to retain the knowledge for a longer period. They may even end up by educating the end-users they visit and rub off some of their knowledge onto them; a bit like 'pollination' but without the mess.

5.2 Tools and Technologies

So, we have now covered a number of 'generics', let us move swiftly on to some of the tools and technologies that can be leveraged in the Rootkit/Stealthkit and Stealth:TNG wars.

5.2.1 Clean Boot

In the days of the DOS stealth viruses there was a saying; '*he who runs first wins*'. This means that whichever managed to execute first; malware or anti-virus, would have control of the system. In the case of the malware this may have meant that it could hide from the anti-virus tool via stealth techniques; effectively making it invisible. If on the other hand the anti-virus got executed first then the game was usually over for the stealth malware; it could be detected and removed.

The usual way to ensure that the anti-virus ran first was via 'clean bootable media'; such as a write protected bootable floppy disk. This usually ensured that no malcode could be running. This allowed a full scan to be carried out of a suspected hard disk without the risk of either:

- Malware being able to hide using stealth techniques.
- Malware being able to infect the 'clean' boot media.
- Malware being able to take advantage of the anti-virus scanner to infect all the files opened by the anti-virus. This was known as a fast infector.

In the days of DOS stealth malware it was rather easy to create and run such a 'clean boot' scan; in fact some anti-virus vendors actually supplied these so-called 'Magic Bullet[30]' disks to their customers as part of their license. This was not just a clean bootable floppy disk [3.5"] but it also contained a modified version of the anti-virus itself.

What has this to do with Rootkits, Stealthkits and Stealth:TNG?

Well, we are basically back to the situation that we were with the DOS Stealth viruses in that it isn't easy or simple to detect such things and this could have serious consequences to not just the security and stability of the infected system, but also to the level of confidence that the system is/isn't infected.

So, what can be done to redress this situation? Are we looking at the return of the clean-boot disk to detect this threat, and if so a floppy disk just isn't going to cut it anymore, we would need bootable CDs or USB thumb drives or similar to cram our modern anti-malware tools onto. Luckily a number of

[30] Dr. Solomon's actually produced such a disk and called it 'Magic Bullet'. It was launched in July of 1996

anti-virus vendors either already have such tools, or are working on them.

There are other makeshift options available now; these include bootable CDs or DVDs using tools such as WinPE, Bart PE or various Linux Live Boot systems with NTFS support.

So, let me quickly cover these solutions as these are the most useful of the current clean-boot methods available.

WinPE
Here is a short description of WinPE from Wikipeadia

"*Windows Preinstallation Environment (WinPE) is a light-weight version of Windows XP/Windows Server 2003 that is used for the deployment of workstations and servers by large corporations. It is also used by OEM's to preinstall Windows client operating systems to PC's during manufacturing. It can also be used as an alternative to MS-DOS as an OS by booting from a CD or USB flash drive instead of booting from a floppy or hard disk.*"

The problem with WinPE is that is it only normally available to OEMs, not normal customers. This makes it less than useful as a possible remediation tool when battling Rootkits, Stealthkits and Stealth:TNG malware.

BartPE
Here is a short description of BartPE from Wikipedia[31], the software can be found on the BartPE website[32]:

"*BartPE is a Live CD version of the Microsoft Windows XP or Windows Server 2003 operating systems.*

BartPE allows a user to boot Windows XP/Windows Server 2003 from a CD-ROM regardless of the condition of the installed operating systems on the internal hard drive. This means that the user can, for instance, recover data from a failed operating system installation, or reset a lost administrator password.

A user can create his own installation of BartPE using the installation disk of the operating system in question and the program PE Builder. PE Builder is available on the BartPE homepage".

The beauty of BartPE is that you can create your own custom ISO with recovery/security tools that you wish to use. Even better is that a number of anti-malware products are supported as standard with plugins, these include: McAfee Stinger, McAfee Command Line Scanner and Lavasoft's Ad-Aware. Other plugins which have been created for BartPE include: Kaspersky[33].

Live Linux
What is Live Linux?

Live Linux is a version of the Linux operating system designed to run directly from a CD, DVD and/or USB flash drive rather than a hard disk. It does not, by default, modify the system [which is acting as a temporary host] hard drive(s), partitions or any files. Personal data and configuration details can be saved to USB flash drives. This effectively makes it possible to use Live Linux on any Intel based computer which has a CD/DVD reader and optionally a USB port.

The latest version of Knoppix can not only read NTFS but now has write access too. This makes using Knoppix for cleaning-up [dis-infecting] far more useful than it was in the past.

LinuxDefender Live![34] from Bitdefender is a modified version of Knoppix complete with Bitdefender

[31] Source: http://en.wikipedia.org/wiki/Bartpe
[32] The web site for BartPE can be found here: http://www.nu2.nu/pebuilder/
[33] Included as part of version 6 of Kaspersky Anti-Virus.
[34] Details and download links can be found here:
http://www.bitdefender.com/bd/site/presscenter.php?menu_id=25&n_id=58

Anti-Virus in a single ISO image ready to be burned to CD. This also offers full NTFS read and write capabilities.

Like BartPE, these live CD bootable Operating Systems are very useful in ensuring that you have booted clean and any malware on the target system will not be active. Because any malware on the target media is inactive it can be detected and removed with little risk. The caveat here is that this is only true if it [the malware] does not do something nasty to the Operating System, partitions, boot sectors or file-structure of the target drive, such as scramble or encrypt them, or carry out other data diddling or booby-trapping of the system if the malware is removed which can't easily be reversed by anti-malware tools.

There are now a number of similar Live Linux versions which can be run from USB flash drives instead of [or as well as] CDs[35]. This option makes the creation of custom bootable USB toolsets easier and more flexible than recreating ISO images to burn to CDs.

However, whichever method you use the bootable media needs to be prepared on a known-clean system and ideally written to media that is [or can be] write-protected, so that it has little or no chance of becoming infected.

5.2.2 Remotely Accessing Drives

The use of so-called Computer Forensic Tools and Utilities can be a useful addition to allowing safe access to a suspect hard drive or partition with little risk of cross-infection. Examples of such tools are Encase[36][Windows] and F.I.R.E [Linux].

The other option is removing a drive from one system and installing it as a slave [non-bootable] drive in a known-clean working system.

However, be warned the risks are somewhat higher using these two methods as simply accessing the drive or files on the infected drive may actually infect the new [clean] host. Examples of this type of infection technique would include malware that uses known exploits, such as the WMF exploit, where simply rendering a bogus file can execute malcode.

If they are used correctly then the risks are minimal.

5.2.3 Anti-Virus

The use of anti-virus technologies as a detection method for infected systems is self-evident, as many of the so-called Rootkits are increasingly being detected by anti-virus products.

Because of this we are seeing the inclusion of techniques in many of the modern malware to allow them to disable as many security and anti-virus products as possible. In some cases this functionality may well be the first to be deployed, as a dropper being spammed out. Once run the dropper lowers or neutralises any local defences and then opens up the backdoor, or just downloads more components as required to complete the infiltration, or in the case of Rootkit, Stealthkit and Stealth:TNG to hide the infection from the Operating System and all the security tools which run on it.

The thing to remember with anti-virus tools is that they can only [normally] detect malware they know about. New malware variants may well be detected by heuristics; however they are still far from perfect.

Many anti-virus tools now detect Rootkits, Stealthkits and Stealth:TNG malware, however the techniques and methodologies as well as the level of detection is rather variable at this time. By the time that this paper is presented, hopefully all major anti-virus products will have 'good' detection techniques and methodologies in place.

5.2.4 Anti-Rootkit Tools

As shown elsewhere in this paper a growing number of malware and spyware are starting to include

[35] These include, Damn Small Linux and Puppy Linux.
[36] More details on Encase can be found here: http://www.guidancesoftware.com/products/ef_index.asp

'rootkit' techniques[37] to allow them to hide from the OS and many security tools as they bind in directly to the kernel.

A number of bots have used a recompiled version of the FU rootkit driver to remove their process entry from the Windows Task Manager; others have used the JiurlPortHide driver for hiding network connections. It does look like we will be seeing increasingly sophisticated and 'invisible' malware as 'rootkit' technologies and techniques get added to the code-base of the major malware and spyware families.

There are a number of tools available that claim to be able to detect and remove rootkits, these are listed below, along with the OS that they are suitable for:

- ChkRootkit [*NIX - http://chkrootkit.org/]
- Rootkit Hunter [*NIX - http://www.rootkit.nl/projects/rootkit_hunter.html]
- RootkitRevealer [Wintel - http://www.sysinternals.com/ntw2k/freeware/rootkitreveal.shtml]
- UnHackme [Wintel - http://greatis.com/unhackme/]
- Blacklight [Wintel - http://www.f-secure.com/blacklight/]

Probably the two best known solutions to this threat for the Windows platform are RootkitRevealer and Blacklight. Below you will find a screenshot of each of these products showing detection of rootkits and/or rootkit protected files and processes.

The first screenshot is of RootkitRevealer from Sysinternals. Here you can see it detecting HackerDefender.

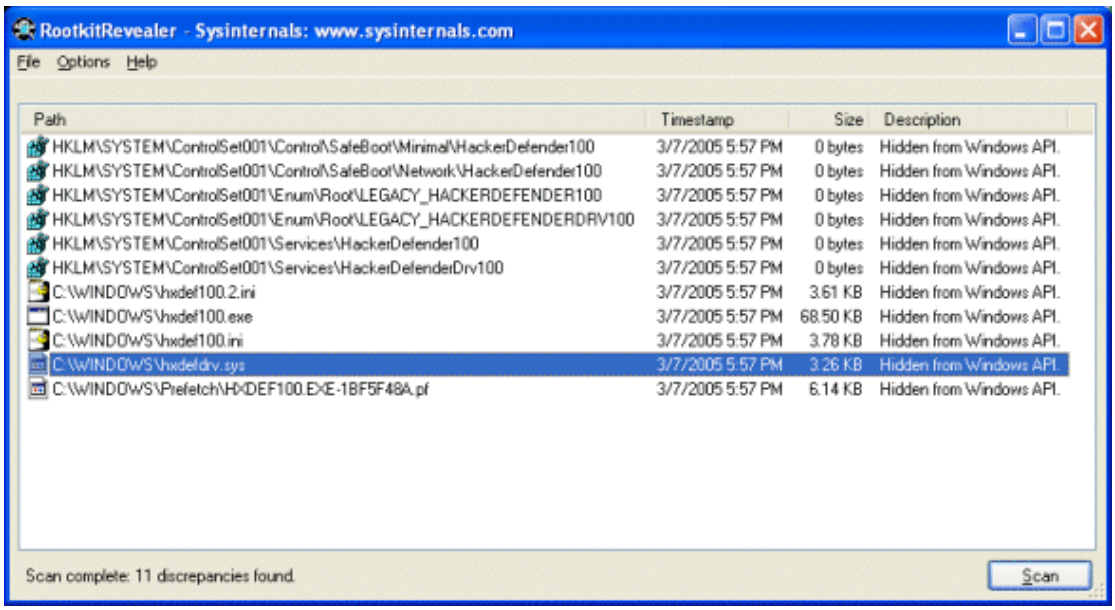


Figure 10: Screenshot of RootkitReavealer showing detection of HackerDefender

The next screenshot is of Blacklight from F-Secure. Here you can see it detecting FU.

[37] More details can be found here: http://www.f-secure.com/weblog/archives/archive-052005.html#00000559

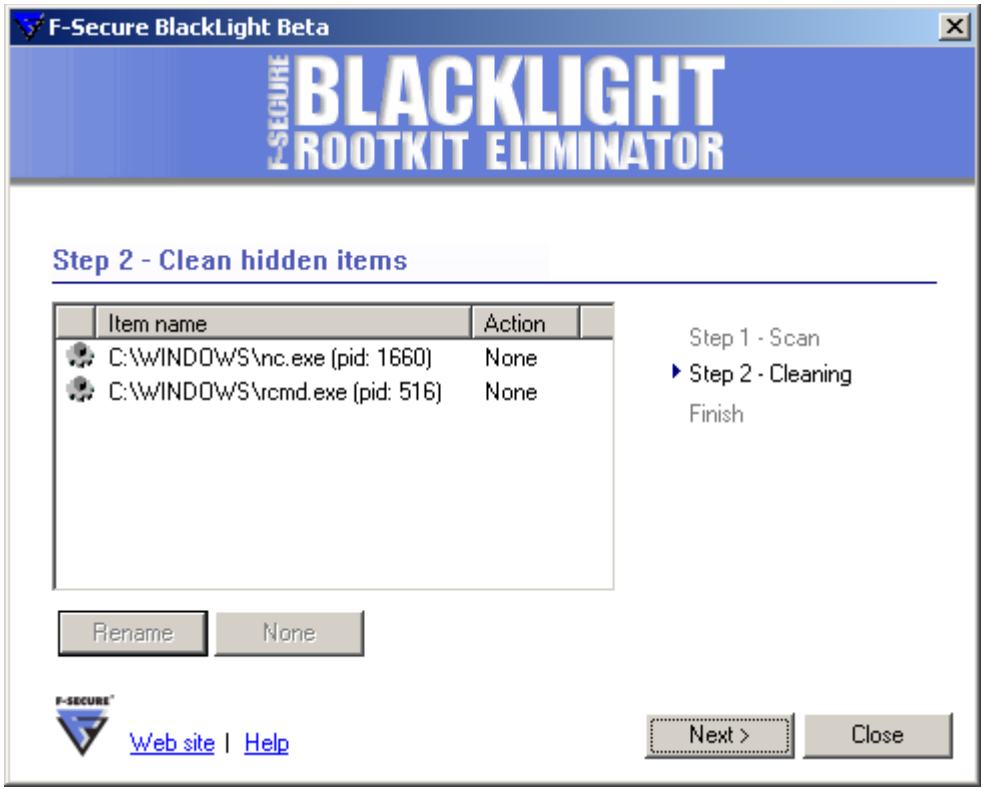


Figure 11: Screenshot of Blacklight showing detection of files/processes hidden by FU.

So, what are these 'anti-rootkit' tools doing? How do they detect [see] something that is effectively invisible?

There are a number of ways that are used, the most commonly used one, is some form of differential analysis of the running operating system. This could be just requesting lists of processes and related data, or a complete system scan, using multiple methods. One will produce a so-called 'tainted' view, and another will produce a so-called 'trusted' view, the two are compared and hopefully the hidden processes and files will appear in the 'trusted' view and be missing from the 'tainted' one. However, if both views are the same, it is assumed that the system in clean; no Rootkit, Stealthkit or Stealth:TNG malware are on the system. This process is most commonly carried out on the 'active' Operating System and therefore it has a greater chance of being subverted by any active Rootkit/Stealthkit/Stealth:TNG malware.

As mentioned by the authors of 'Hide N Seek Revisited – Full Stealth is Back' when discussing this method of acquiring the two distinct views, they state: "*All sources can be tampered with without any noticeable side effects*".

This makes it clear to me that although this is a useful technique, when used on a potentially compromised [tainted] system, it can be subverted to effectively give back misleading data; such as that the system is clean, when in fact it is still infected and under control of the malware.

The authors of the paper go on to state: "*Obviously a simple 'tainted view/clean view' comparison is not enough. Either we will have to find alternatives for the comparison approach or we will have to prevent rootkits from recognizing the scanner*".

Indeed.

A potentially better way to use this 'tainted/trusted' method is by assuming that a scan of the active [on-line] Operating System, drives and registry hives is the 'tainted' view and that this is compared to a similar scan performed via a 'clean-boot'; this means that the Operating System [which may be compromised] is effectively 'off-line' and therefore the chance of any malware 'skewing' the result is minimised. This is the method that Microsoft published a research paper on; they called the project

Strider Ghostbuster[38].

Another common trick is to monitor registry keys which are commonly used as launch-points for malware; to ensure that they get loaded when the system starts. However, this assumes that the malware is not already in control of the system and hiding the modifications of the very same registry keys that the tool is monitoring.

The bad girls and guys are actively trying to detect or otherwise defeat or bypass the anti-rootkit scanner. The techniques so far tried include:

- Obfuscation techniques.
- Detecting the scanner and un-hiding themselves. (Defeats the tainted/trusted view technique)
- Adding 'Rootkit/Stealthkit/Stealth:TNG' process and/or file name(s) to trusted list.
- Heuristical detection of the scanner.
- Signature detection of the scanner.
- Packer support, to allow unpacking of packed scanners.

There are numerous other techniques that could be used with great effect; however I will not include them here as I don't want to be accused of giving the bad guys and girls new ideas.

As far as I can see at this time, the *only* reliable method of detecting this threat, as it was in the days of DOS malware using stealth, is to 'clean-boot'; this is the only way that we can be fairly certain that the malware isn't in control of the system and feeding us false data.

5.2.5 Other

This section will quickly mention some of the other tools, techniques and methodology which may be useful in combating Rootkit/Stealthkit/Stealth:TNG threats. More data on these areas can be found in several of my other papers.

- Honeypots and Honeynets; used to act as early warning system of new network worms and droppers.
- IDS and IPS; can be used to detect traffic from compromised systems phone-ing home and so on.
- Perimeter firewalls; can be used to restrict which ports and protocols can traverse your network and your internet connection.
- Partitioning your network with router ACLs and internal firewalls; can be used to protect more sensitive or valuable parts of your network.
- Patch management; can help to minimise penetration of malware which relies on known vulnerabilities.
- Strong passwords; many bots and other malware carry a list of common and/or weak passwords in their body.

6 Conclusions

Rootkits were just a *NIX problem for many years, they have now made the move to Windows and caused a considerably amount of concern. The problem came to the fore in 2005 when the Sony 'rootkit' story broke and the malware authors first took advantage of it. Since then, the malware authors have been very busy, not only taking advantage of existing Windows rootkits and stealthkits such as HackerDefender, FU and AFX, but also alerting them to the value of stealth and the existence of many stealth techniques used in the early days of malware writing, when DOS was the dominant IBM PC [and clones] Operating System.

I believe that modern rootkits should be called stealthkits, where the functionality is supplied by a separate tool. Where this functionality is included in the body of the malware itself, it should be called stealth or Stealth:TNG.

We seem to be currently witnessing the rebirth of stealth. This is turn has caused many security vendors to have to dust off anti-stealth techniques and technologies and adapt them to the modern

[38] Details on Strider Ghostbuster can be found here: http://research.microsoft.com/rootkit/

Windows platforms or in some cases to start again from scratch as they [the vendor] were not around when stealth was last a threat.

I expect that Windows rootkits and stealthkits will only be a stepping stone, a stop-gap, for most malware authors before they learn stealth techniques which they can, and will, include in their offspring, in fact they already are. Pandora's Box is open once more and not only are the malware authors using what they find inside, but so are the adware and spyware creators.

The most worrying part of this will be the use of stealth by cyber-criminals to allow them to hide the presence of their malware for longer, which will allow them to misuse a compromised system more fully, not only for use as part of a botnet, but also for theft of intellectual, personal and financial data, but also as a stepping stone to increase the penetration of corporate and academic computer networks.

This in itself is the one most companies and academia will be concerned with as they may well lose extremely valuable data and research which will then be sold to the highest bidder.

Many of us that have been in the industry since the start of the malware problem have been expecting the re-emergence of stealth, and are somewhat surprised just how long it has taken for stealth malware techniques to be migrated to Windows. More worryingly I am somewhat stunned that some vendors have seem to have been caught napping by the re-emergence of stealth techniques and tools, it really isn't good enough; it isn't as if we [and they] haven't been in this situation before.

For the time being, I believe that the most acceptable solution to the problem will be to 'clean-boot' from CD/DVD or USB solutions such as WinPE, BartPE or a Live Linux solution as otherwise you may be lead to believe that you have a clean system, even though the bad guys and girls are in full control of it.

As for the use of stealthkits and Stealth:TNG in commercial software, I believe that this is a 'bad' idea as protection of key files/processes can be achieved in other ways, such as via encryption, digital signatures and other more acceptable and more ethically sound techniques.

7 Thanks and Feedback

I would like to thank F-Secure for their assistance with data used in this paper, and Mika Ståhlberg in particular for his excellent presentation at EICAR 2006 and kindly sending me his slides and allowing me to use some of the data from them. Thanks also go to members of the IBM MSSD and IBM Virus Emergency Response Teams.

All constructive feedback on this paper will be warmly received.

Further Reading/Resources

- FAT32 - a new problem for anti-virus or viruses? - Proceedings of the 9th International Virus Bulletin Conference 1997 – Martin Overton - http://arachnid.homeip.net/papers/VB97-FAT32.pdf
- Worm Charming: Taking SMB Lure to the Next Level - Proceedings of the 13th International Virus Bulletin Conference 2003 – Martin Overton - http://arachnid.homeip.net/papers/VB2003-Worm_Charming.pdf
- Spyware: Risks, Issues and Prevention – EICAR 2005 – Martin Overton - http://arachnid.homeip.net/papers/EICAR2006-Spyware-v1.0.2.pdf
- Bots and Botnets - Risks, Issues and Prevention - Proceedings of the 15th International Virus Bulletin Conference 2005 – Martin Overton - http://arachnid.homeip.net/papers/VB2005-Bots_and_Botnets-1.0.2.pdf
- Detecting Stealth Software with Strider GhostBuster -Yi-Min Wang; Doug Beck; Binh Vo; Roussi Roussev; Chad Verbowski - February 2005 - http://research.microsoft.com/research/pubs/view.aspx?type=Technical%20Report&id=875
- Hide 'N Seek Revisited Full Stealth Is Back. – Proceedings of the 15th International Virus Bulletin Conference 2005 - Kasslin et al - http://www.f-secure.com/weblog/archives/KimmoKasslin_VB2005_proceedings.pdf
- Rootkits, Part 1 of 3: The Growing Threat. McAfee - http://download.nai.com/products/mcafee-avert/WhitePapers/AKapoor_Rootkits1.pdf
- RootkitRevealer - http://www.sysinternals.com/utilities/rootkitrevealer.html
- The Art of Computer Virus Research and Defense – Peter Szor – Symantec press – ISBN 0-321-30454-3
- Dr. Solomon's Virus Encyclopedia
- 2006 Australian Computer Crime and Security Survey - http://www.auscert.org.au/images/ACCSS2006.pdf

8 Appendix A

A selection of descriptions on Rootkit\Stealthkit\Stealth:TNG malware:

- http://www.f-secure.com/v-descs/maslan.shtml
- http://www.f-secure.com/v-descs/myfip_h.shtml
- http://www.f-secure.com/v-descs/padodorw.shtml
- http://www.f-secure.com/v-descs/lecna_b.shtml
- http://www.f-secure.com/v-descs/haxdoor.shtml
- http://www.f-secure.com/v-descs/elitebar.shtml
- http://www.f-secure.com/v-descs/hacdef.shtml
- http://www.f-secure.com/sw-desc/apropos.shtml
- http://www.f-secure.com/v-descs/xcp_drm.shtml
- http://www.f-secure.com/v-descs/feebs.shtml
- http://www.f-secure.com/v-descs/fu.shtml
- http://www.f-secure.com/sw-desc/navipromo.shtml
- http://www.f-secure.com/v-descs/agent_p.shtml
- http://www.f-secure.com/v-descs/gurong_a.shtml
- http://www.f-secure.com/v-descs/hearse_a.shtml
- http://www.f-secure.com/v-descs/mailbot_az.shtml
- http://www.f-secure.com/v-descs/small_la.shtml