

Research Proposal

by

Raghunathan Srinivasan

MS, Computer Science

ASU ID: <Deleted>

Department of Computer Science & Engineering

Arizona State University

Tempe, Arizona

Committee Approval

Chair/Member	Name	Signature
Chair	Dr. Partha Dasgupta	
Member	Dr. Charles Colbourn	
Member	Dr. Aviral Shrivastava	

Contents

S. No	Name	Page #
1.	Introduction	1
2.	Background	2
3.	Recent Trend in Malware	5
4.	Terminologies to be used	7
5.	Assumptions made in this research	8
6.	Threat Model	8
7.	Storage of virus definition files	10
8.	The Hidden Anti Virus Process	12
9.	Completion Criteria	14
10.	Timeline	14
11.	References	15

1. Introduction

Malware are a very big threat in today's computing world. The older viruses which attempted to delete files or crash a hard disk have been replaced today by stealth malware that aim to do something very different. Many of the recent malware attempt to steal information such as passwords, credit card numbers and social security numbers.

One of the first well known worms was the Internet worm in 1988 [1]. It broke into the system by means of flaws in the system software. The worm was not designed to do any damage, but was simply made to estimate the size of the internet. However each computer started getting infected multiple times and the consequence was that the machines became very slow due to running many processes to the point that they could not be used.

The first few viruses were simple machine language programs [2]. The viruses had the ability to infect other executable code. When a user would execute such code, the virus would take control of the system and infect more files. Such viruses were known as parasites as they would not totally destroy the running machine. These viruses were easy to detect and remove, as administrators only had to search for patterns of instructions to find instructions that matched an infection. These patterns were known as virus signatures. This method worked well till the virus writers started writing polymorphic viruses. Writing a complex routine that could change portions of it significantly and also retain virus like properties was a challenge, hence the writers chose for a compromise and wrote virus programs that were encrypted, with the encryption engine being different on every infection. The flaw with this method was that any encrypted code has to be decrypted in memory to execute.

The anti virus researchers used this and created CPU emulators that would run the code. This allowed detection of encrypted polymorphic viruses. After this came a generation of malware that would install a root kit, and save the virus programs as hidden and system files and remain elusive to most of the anti malware programs.

Today, malicious programs like the SpamThru Trojan, and the Beast Trojan, attempt to shut off the security processes such as Anti Virus, Firewall etc. running on the system they infect, hence making the host computer a sitting target for all sorts of exploits. This marked a significant change in the tug of war between the viruses and the anti virus software. Till now any infection could be contained and cleaned by the anti virus after the arrival of the update. Once the

malicious programs started targeting the anti virus products themselves, there would not be any further updates.

The methodologies used by anti virus companies have undergone changes, however the latest trend of killing the anti virus process threatens to make their presence inconsequential. There is an urgent need to address this issue. This research aims to create a process that can evade detection by such programs, but still stay visible to the System Administrator and the User.

The rest of this paper is organized as follows. Section 2 deals with background information on viruses, anti virus software and a few other malware. Section 3 explains the recent trend of attacks that attempt to shut security software running on the system. Section 4 briefly proposes a solution to the attacks. Section 5 introduces the list of terms that will be used in Section 6 and 7. Section 6 discusses methods to create a hidden process. Section 7 has the details for storage of definition files. Section 8 gives a timeline for the research.

2. Background

In this section an overview of the evolution and description of viruses, anti virus software and other malicious programs are explained.

2.1 Computer Virus

Computer viruses are programs that are deliberately designed to interfere with computer operation, record, corrupt, or delete data, or spread themselves to other computers and throughout the Internet [3].

Personal Computer Viruses first started appearing in the 1980's. In 1990 it was estimated that there were 500 virus programs. By 1996 the estimate rose to around 10000. It was estimated that by 2002 there were 60,000 viruses [4] Today the number of known computer virus is estimated to be around 103,000 [5]. The numbers suggest that the most predominant jump happened in the period from 1990 to 1996. There seems to be a relative slowdown in the period immediately after 1995. One of the reasons for this slowdown is attributed to the fact that most consumers around the globe shifted from using MSDOS to Windows which forced the virus writers to adapt to the new systems.

2.1.1 Types of Viruses

Viruses are categorized by their functionality and methodology.

1. Boot Virus

These types of viruses operate by infecting the Master Boot Record of a PC. The MBR is a program that resides on the first sector of a hard disk; it runs every time a PC starts up. This makes a Boot sector virus extremely dangerous. They are very difficult to remove and it requires a bootable anti virus disk to properly remove the virus.

2. Parasitic Virus

A program that attaches itself to executable files as part of the executable's code. It leaves the contents of the executable unchanged but attach in a way such that it runs every time the program runs.

3. Bomb

This is a virus that does not propagate, but simply resides on the machine. It gets triggered by some event such as date or an event.

4. Macro Virus

These are programs that take advantage of the macro utilities that are built into programs like Excel and Word. They ride on the data files and infect a large number of machines in the process.

5. Encrypted virus

This type hides its functionality by encrypting the payload. One of the first encrypting viruses for DOS was Cascade in 1987 [6]. The virus body consists of a static decryption routine that decrypts the encrypted payload.

6. Oligomorphic virus

The anti virus scanners reacted to the Encrypted viruses by detecting the static decryption routines. Virus authors responded by configuring their creations to choose between many different encryption/decryption routines, this behavior was known as oligomorphism. The first known virus of this kind was "Whale".

7. Polymorphic virus

Polymorphic viruses possess the capability to mutate their encryption engines, thereby making them random and very complex to detect. The first known polymorphic virus, 1260, was written in the U.S. by Mark Washburn in 1990 [7].

2.2 Other Malware

1. Trojan Horse

This is a program that enters a machine disguised or embedded inside legitimate software. The Trojan looks harmless or something interesting to a user, but is actually very harmful when executed. Each Trojan has its own characteristic that is totally dependent on what the designer intended it to do. The Trojan depends on successful implementation of social engineering concepts as it has to fool a user into installing the code. It does not depend on security flaws or loopholes present in the system. Once inside a system, it can exploit any resource or use the machine as a zombie, or use the infected system as a launch pad for further attacks

2. Worms

A worm is a self replicating program. Unlike a virus it does not attach itself to any existing program. It uses the network resources to infect other machines in the network. Worms always harm the network whereas viruses always infect or corrupt files on a targeted computer.

3. Rootkit

Rootkit is term derived from the UNIX term root. It was designed to give administrator privileges to the attacker. A well written rootkit can hook on the Operating System's Page fault handler and Virtual Memory controller to conceal its presence, and that of its files [8]. In recent years root kits have been used increasingly by malware, helping an intruder to maintain access to a system whilst avoiding detection. One of the most famous rootkit is the SONY DRM root kit. Sony intended to install a program on the consumer's PC that would not let the user rip any music files. This program hid files, registry keys and processes starting with the string \$sys\$. This led to many attackers use files starting with the above string.

2.3 Anti Virus

The appearance of computer virus in the 1980's, caused the emergence of the anti virus. Anti virus software has constantly evolved with the virus. The methodologies used by anti virus software are as below:

1. Signature detection or Pattern Matching

The initial programs would scan the entire executable file to find the presence of the virus code. Later the programmers found out that most of the virus infections involve virus placing its code on the entry point of the program. Hence the scanners would check

for the entry point of the program, this methodology became obsolete when Encrypted and Polymorphic viruses emerged.

2. X - Raying

As encrypted viruses arrived, anti virus software started using brute force decryption of the code. This was possible since they had to do known plaintext attack on the encrypted code. The virus writers used random encryption algorithms; hence the protocols were easy to crack. This type of scanning was known as X-raying [10].

3. Emulation

Another technique that emerged due to the appearance of polymorphic viruses was Emulation. This involved starting a CPU emulator and loading the executable file on it. The virus would not realize that it was being run on an emulation environment. This allowed the software to observe the behavior of the program in a closed environment where no damage would be done to the user machine.

4. Frequency Analysis

This was not a stand alone method. Frequency analysis involved scanning a code for the presence or absence of particular opcodes. Programmers found that most legitimate programs would use DOS interrupts like 21h in their code. This interrupt was hardly visible in malicious codes. Hence they would examine frequency of such opcodes, and programs having them would not be scanned

5. Heuristics

Virus writers slowly started using techniques such as entry point obfuscations, by which they would not keep the virus code at the entry point of the executable. The emergence of viruses that could change their properties required a change in virus detection technology. Programmers came up with decision support systems, where a scan would have weight system. Points were allotted to behavior such as memory access, file access, page faults. After scanning, the number of points accumulated determines if a file would be flagged for a virus from a known set of behavioral trend.

3. Recent Trend in Malware

As Anti virus products made it tougher for the viruses to escape detection, the virus writers started a new trend. The malicious programs started shutting off the anti virus and other related security software in the system. The increasing number of such attacks in recent months shows how vulnerable the anti virus software is to such attacks. Three such Trojans are discussed in the following section:

1. SpamThru Trojan [11]

This Trojan demonstrates all sorts of loopholes that can be exploited in our current security architecture. Once installed on a machine by means of some social engineering. It patches the running anti virus by blocking updates. It does so by changing the anti virus update sites to point to the localhost address. After this it downloads and installs a pirated copy of Kaspersky AntiVirus for WinGate into a concealed directory on the infected system. Once this is done, it patches the license signature check file of the anti virus so that the user does not receive any notification of the expired license. It then runs the anti virus to find the presence of any other malware and removes them – this is done so that there is no competitor for system resources. It conceals its own files from the scanner. It then also installs a P2P engine to keep track of IP addresses of all infected machines. It runs a control server from any of the infected system, and transfers control if the control server is shut down.

2. Beast Trojan[12,13]

This is a backdoor Trojan horse. It works as a RAT (Remote Administration Tool). This was first found in 2002. Beast employs client/server architecture. This was one of the first Trojan to establish a reverse connection in client/server architecture. The server did not require the IP address of the client. The server would just connect to a DNS that would in turn connect to the client. The first version of the Trojan injected its DLL's into explorer.exe. The latest version of the Trojan injects its DLL's into winlogon.exe, which is a process in Windows 2000 and XP that is always active. Winlogon.exe also handles the logon process for windows. The Control+Alt+Delete sequence is also handled by winlogon.exe. Its ability to inject its DLL's in a critical system process demonstrates the loopholes present in the current system. Once it infects a system it shuts off the anti virus and Firewall and the attacker essentially obtains complete control of the system.

3. Win32.Glieder.AF [14]

This is a Trojan that downloads and executes files from a list of URL's. This Trojan is spammed to users by another Trojan - Win32.Bagle.BG. The Trojan arrives in a mail having a zip attachment.

When the executable gets launched, the Trojan copies itself to *%System%\winshost.exe*

It then adds winshost.exe to the startup process by adding the registry item:

```
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\winshost.exe =
```

```
""%System%\winshost.exe
```

After doing this Glieder.AF deletes registry items if found, belonging to security software such as firewall, anti virus, etc.

4. Terminologies to be used

Encryption - Encryption is the process of obscuring information to make it undecipherable without some special knowledge about the information.

Key – Key is a piece of information that is taken in by the encryption/decryption algorithm to produce ciphertext/plaintext from plaintext/ciphertext.

Symmetric key – When the same key is fed into the encrypting and decrypting engine. Symmetric keys have to be known both to the encrypting and decrypting location.

Public & Private keys – In public key cryptography, the private key is kept secret, while the public key may be widely distributed. The user generally uses private key to decrypt, while people wishing to send secure message to the user, encrypt the message using the public key of the user.

Steganography – This is the art of writing hidden messages such that no one apart from the recipient and sender know the existence of a message. Here the existence of a message is disguised, whereas in cryptography the contents of the message are obscured.

Hash – A hash of a long string of variable length is a fixed length string as output. Hashes of two different strings are different, and hash of the same string taken any number of times produces the same output.

Obfuscation – Obfuscation is the process of making source code unreadable. Obfuscation is done to prevent reverse engineering and to manage unauthorized access to source code. The risks from these accesses may be loss of intellectual property, ease of probing for application vulnerabilities.

Object code obfuscation – This is a method used to protect intermediate code such as those compiled from java and C #

Binary Obfuscation – This is the process of obscuring the raw bytes of an executable so that obtaining assembly level code from the executable is difficult.

Threads – Threads are a way for a program to split itself into two or more simultaneously (or pseudo-simultaneously) running tasks. Threads may be User level threads or Kernel level threads.

If User level threads are utilized then the operating system cannot distinguish between the code executed by each of the threads.

Pixel – A Pixel is a single point in an image. A pixel may be represented by anywhere between 8 and 24 bits.

5. Assumptions made in this research

This research aims at protecting an anti virus from getting disabled or shut down by an attacking virus. There are a few assumptions that will be made while tackling the problem of malware shutting or incapacitating the anti virus. The assumptions made in this research are

1. The Anti Virus will get installed on a clean machine
2. The Virus will not attempt to shut down all processes
3. The Virus will not attempt to delete all files or all files of a particular format
4. The Virus will allow applications to upgrade to newer versions

The assumptions have been made for the following reasons. If there is already the presence of a virus or a root kit on the machine, then the virus may be able to hinder the installation process, or make a patch on the software being installed. This will render the software ineffective.

If the virus shuts down all processes or attempts to delete all files, then the user will certainly get some indication that there is some virus residing on the machine, it is also not possible to protect against such viruses that will kill all running processes. The last assumption is made so that the anti virus can request for upgrades and receive new definitions.

6. Threat Model

A virus attack may have one or many of a variety of motives behind it. The most common reason is to take control of the machine and turn it into a Zombie. The compromised machine becomes a part of a botnet. The machine then performs a variety of tasks that it is commanded to do from a remote terminal. The machine maybe used as a launch pad for sending out spam, spread spyware or even steal private information such as passwords on banking sites. The number of viruses that are used to churn out quick money is increasing very rapidly [15]. The capacity and strength of an attacking virus is expected to increase as the stakes and money involved with organized cyber crime increases.

A virus potentially has the capacity to do anything it wishes on a system. This includes disabling security software such as Firewall, Anti virus, killing or mimicking any process, change operating system files, delete any file on the machine and above all format the entire hard drive. The virus can also install a root kit which will make it close to impossible to detect. It is nearly impossible to protect against viruses that install root kits. The attacks can be classified into two broad categories, one that attempt to delete all files or do a format of the system, and the second which will attempt to install itself and reside on the host machine. It is quite difficult to deal with the attacks of the first kind. The virus proceeds to remove everything, in such an eventuality the machine will shut down and a re install of the entire operating system will be required. The developer of such an attack does not get any profit or return from this type of attack, this kind of attack is aimed just to have fun.

The second type of attack is more complex, and ironically, can be dealt with compared to the first type of attack. This type of attack includes patching operating system files, installing keystroke logger, spyware, installing a root kit to convert the machine into a zombie, etc. These viruses will somehow attempt to stay concealed within the machine or run in the background. Another motive for these attacks might be to steal information such as credit card information, passwords, social security numbers, private key, etc. Generally this type of attack will not involve deleting system files, it will only attempt to shut down or patch a few programs in an attempt to conceal itself. Hence a virus will more often than not try to disable the anti virus, or not allow update of virus definitions to happen so that it can stay hidden, or patch the definitions to make the anti virus think that it is a legitimate program. The anti virus programs need to overcome these attacks. This research aims at countering this exact problem.

To solve this problem, security software can adopt a few tricks from the virus world. Polymorphic viruses are very difficult to detect as they keep on changing form, if the anti virus software keeps changing its form then it would be difficult for malicious programs to shut it down. If the anti virus software can change its process name, file name, directory name, executable, registry entries, code and checksum from time to time, it will be very difficult for virus writers to write a program to shut off the anti virus. Hence there is a need to create a process that can hide its presence from other programs. At the same time the user at the machine should be able to identify a process as the anti virus software. This is necessary because in the eventuality that a virus patches onto to the hidden anti virus, the user can uninstall the software. Creating such a process will comprise of two steps, the first would be to scatter and hide the virus definition files and the second would be to hide the process itself.

7. Storage of virus definitions

The virus definition files should be embedded in image files by means of Steganography. One or two bits of data can be stored in every pixel in the image file. Apart from the definitions, the image files should also contain a section for storing the checksum for all the definitions present in the file. The checksums must be encrypted with the private key of the software provider. This is done so that anyone can decrypt the value using the software provider's public key to read the checksum, but no one can encrypt and place a false checksum. This takes out an attack where a malicious code might remove itself or change some definition in an attempt to confuse the software. There is no way that malicious code can place a correct checksum without knowing the private key of the software provider.

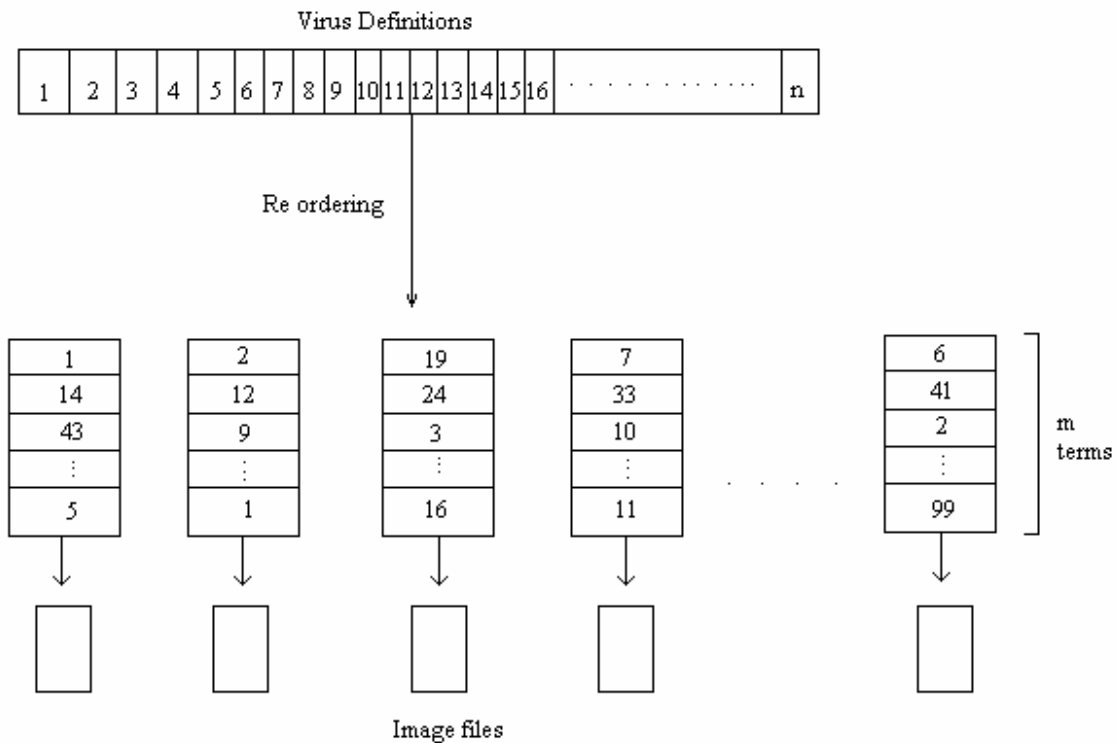


Figure 1. Placing Definitions in Image files

The individual virus definitions themselves should not be stored at only one place. They must be stored at multiple places, and grouped randomly. To avoid complexity at the user end software, this can be done at the control server running at the software vendor's site. This is summed up in Figure 1 and explained in detail below.

Let $D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8, D_9, D_{10}, D_{11}, D_{12}, \dots, D_n$ be the virus definitions.

The definitions should be stored bit by bit in every pixel in the image file.

Suppose the control server decides to store m definitions in every file. The server should then decide on how many copies it wants. Let the number of copies be c .

Therefore $c*n$ will be the total number of definitions to be stored.

Since m definitions can be stored in one file, $c*n$ definitions will take $c*n/m$ files.

The control server has to generate $c*n/m$ files, and in each of the file the definitions have to be placed in a random order. The same definition should not occur twice in one file, otherwise an attacker corrupting contents of one image file, might end up corrupting all copies of the definition.

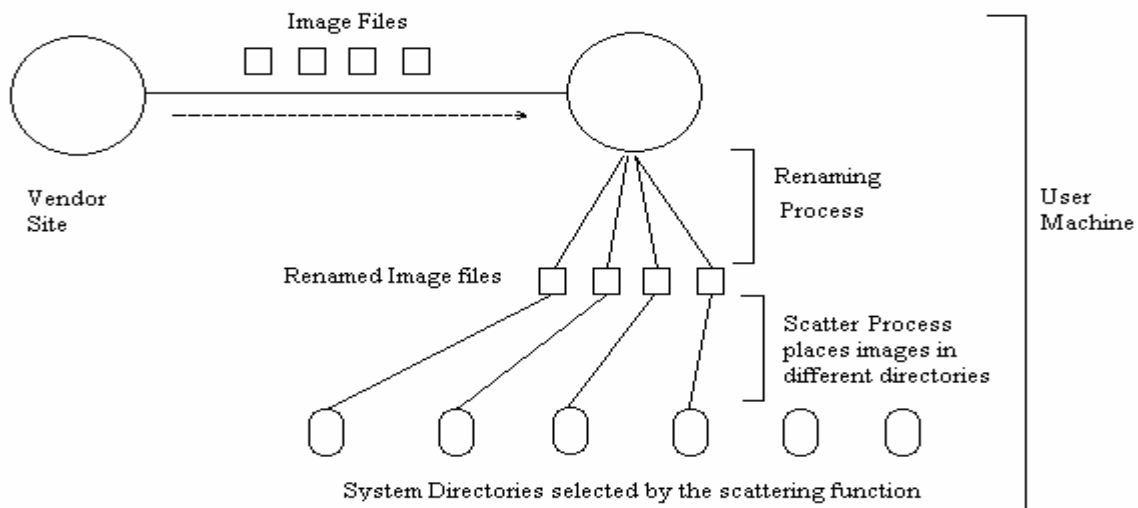


Figure 2: Receiving and storing image files

Figure 2 Illustrates the steps involved in storing a image file containing the virus definitions. The image files need to be renamed randomly at every user machine. This needs to be done so that an attacker doing reverse engineering at his machine cannot use that knowledge to attack the definition files on other user machines. To do this, a renaming function – R can be created, which will be placed inside the binary by the installation suite. The function R can be a pseudo random number generator, whose seed can be placed in the executable, the seed must be different on every executable. R must generate a name for each of the image files. The names provided by the software vendor can be treated as base names, and the list of base names can be present inside the executable to easily locate which files to load definitions from.

The image files should not be present in one directory. They need to be relatively scattered around system directories. To do this there should be a scatter function - S, present in the executable that takes in some parameters of the machine and generates a list of directories where the image files can be stored. The image files should be placed inside these directories.

8. The Hidden Process

A process can be hidden in more than one way. One of the simplest approaches would be to write a root kit that hooks itself onto the operating system memory management component and the page fault handlers. This would keep the anti virus hidden from most scans, and the anti virus will be able to monitor the system. The problem with this approach would be that if in any eventuality a virus patches on to the anti virus software then the virus can never be removed.

Another approach to this would be to use random naming sequence such that the process name, directory names all turn out to be some random combination of numbers, symbols and letters. The approach might fail if the virus program is designed to spot such random constructions.

An approach that might just work would be to patch onto an already existing process and start up as that process every time. This way a virus would not know which of the running process is an anti virus. The only way it can detect and shut off the anti virus would be to either shut off all processes, or do perform reverse engineering on all the running processes to extract executed opcodes. However, this approach has its own drawback that if the user wants to run the actual process, then the anti virus software would need to migrate to another process.

This research will aim to create a process that will masquerade as some other process on the outside, just like some viruses do. For this purpose the software must install itself as relatively well known software which may not be present on every machine, like VLC Player or MPlayer. This will involve copying the entire directory structure and file names of that software and placing the anti virus program's files in them.

The Installation suite should have a list of programs that are not installed very commonly by users, this is done to avoid as many name clashes as possible. Upon Installation request, the suite should find out which of those programs have not been installed on the machine. Once the suite finds this, it has to pick a process at random and install the anti virus software as that process. In the rest of the paper this chosen process will be referred to as P'. The suite then has to create

directory structure and registry entries such that they match that of P'. By doing this the attack where many Trojans simply search and delete registry items will get nullified.

The exe file must have places for inserting random code to create a different checksum every time. This should be done so that any program obtaining the checksum of the anti virus program gets some random value at every machine.

The executable's binary code must be obfuscated to prevent static analysis of the binary. Trace analysis and memory access analysis of the process can be prevented to a certain extent by adding multiple user level threads. When started, the process should start multiple user level threads. Of these threads only one thread should do the actual monitoring of the system, the other threads must perform random memory access and random computations and execute random instructions. Since the threads would be executed at the user level, any program obtaining the Kernel snapshot of the process will obtain a random mix of executed code, and will find analysis of the trace difficult.

When the anti virus loads up as P', it has to locate the image files using S and R. Upon reading the virus definitions, it has to hash the items read, and should match the hash with the checksum stored in the files. The checksum will be obtained by decrypting with the public key of the software vendor. The public key of the vendor can be stored inside the anti virus executable. If the process finds that the image files have been tampered, it has to contact the vendor by means of a secure connection and obtain fresh image files. Updates should also occur in a similar way. The vendor should send out complete image files, with the definitions and encrypted checksums.

The control server should once in a few weeks issue out command to the software running at the client machine to flush all image files it is using and obtain fresh files from the server. When a new image is sent to the client machine, it has to be renamed according to the function placed in it by the installation suite and saved in some directory according to the scatter function. The control server should also direct the software running at the user machine to change its credentials from time to time. This will involve repeating the process that was done during installation. The process at the user machine has to receive a bunch of program names, it should select one from them and inform the server of its selection, and then the server can send the directory structure to the program, which creates fake directories, fake registry entries in the user machine and deletes the old directories. This will also involve changing the scatter function and the renaming function.

The anti virus program can also choose to store the hash values of all residing executable files on the machine. If this is done then on subsequent scans the anti virus software can check whether or not any executables have changed. If there have been any changes, then those executables can be flagged for user review. For the user scanning this list may be a difficult and arduous task, but some attacks like patching on to the explorer or to certain other system files can easily be uncovered by this. If the user knows that no updates were performed, then most likely the changes are part of a virus attack.

9. Completion criteria

The completion criteria for the project are:

1. The process can install itself as different programs on different machines.
2. The names of image files are random on every machine and the images are scattered randomly in every machine.
3. The process can survive attacks that attempt to switch off security software.
4. Process can receive commands from a control server to change directories, image files, renaming and scattering functions.

10. Timeline

Completion Date	Hidden Process	Definition Files
December 2006	1. Identify which programs are installed on a machine	1. Store plaintext data in Image files
January 2007	1. Store public key, generate R & S functions	1. Create and store encrypted Store encrypted checksums 2. Rename Image files
February 2007	1. Create fake directories and registry entries	1. Scatter image files using S
March 2007	1. Process to run, verify checksums, notify errors and perform updates 2. Perform migration of process, image files to different directories	
May 2007	Completion of Thesis	

11. References

1. Eugene H. Spafford, “The Internet Worm Program: An Analysis”, Purdue Technical Report CSD-TR-823.
2. Carey Nachenberg, “Computer Virus — Coevolution”
3. http://www.microsoft.com/athome/security/viruses/intro_viruses_what.msp
4. Munro, J. “Antivirus Research and Detection Techniques”, (July 1, 2002)
[<http://www.extremetech.com/article2/0%2C1558%2C325439%2C00.asp>]
5. <http://www.cknow.com/vtutor/NumberofViruses.html>
6. Daniel J. Sanok Jr., “An Analysis of How Antivirus Methodologies Are Utilized in Protecting Computers from Malicious Code”, Information Security Curriculum Development (InfoSecCD) Conference, September '05, ACM
7. <http://www.viruslist.com/en/viruses/encyclopedia?chapter=153311150>
8. <http://www.certifiedsecuritypro.com/content/view/137/90/>
9. Sherri Sparks, Jamie Butler, “Shadow Walker - Raising The Bar For Windows Rootkit Detection”, Black Hat, USA 2005
10. Igor Muttik, Network associates Inc, “Stripping down an AV engine”, Virus bulletin Conference, September 2000
11. <http://www.eweek.com/article2/0,1895,2034680,00.asp>
12. <http://www.answers.com/topic/beast-trojan>
13. <http://lists.virus.org/dshield-0310/msg00337.html>
14. <http://www3.ca.com/securityadvisor/virusinfo/virus.aspx?id=42627>
15. Virus bulletin conference 2006 - Report