Global Information Assurance Certification Paper

MALWARE 101 – VIRUSES

GSEC Gold Certification

Author: Aman Hardikar .M, amanhardikar@yahoo.com

Adviser: John C A Bambenek

Accepted: April 12th 2008

© SANS Institute 2008, Author retains full rights.

TABLE OF CONTENTS [TOC]

FIGURES

TABLES

ABSTRACT

This paper provides new insights into establishing Incident Handling procedures for dealing with various types of malware. It also aims to give a detailed perspective into the various types of malware or malicious software and their propagation mechanisms. Malware needs to be handled in a certain way depending on its type and to do that, the different malware types and their handling procedures need to be understood. A clear handling procedure will help security personnel to quickly and efficiently handle the malware threat and reduce the impact/business disruption to the corporate users.

The paper is structured in the following order:

- Introduction to Viruses
- Subtypes and Working of the Viruses
- SANS Six Step Incident Handling Process

In this paper, the focus will be on one of the self replicating malware namely, Viruses. We will look at the various types that exist, how they work and the ways to handle them.

Keywords:
Virus, viruses, incident handling, virus types, identification mechanisms, malware, information security, malicious code

© SANS Institute 2008, Author retains full rights.

1. INTRODUCTION

1.1. MALWARE – OVERVIEW

According to NIST,

"**Malware** (NIST, 2005) *refers to a program that is inserted into a system, usually covertly, with the intent of compromising the confidentiality, integrity, or availability of the victim's data, applications, or operating system (OS) or of otherwise annoying or disrupting the victim.*"

Malware is the term that represents all software whose purpose is malicious in nature. There are many different types of malware. Some of the common ones are virus, worms, trojans, backdoors, rootkits, bots and spyware.

Virus: This is the most common type of malware that is found and is also used to represent multiple subcategories of the malware genre. It is a type of malware, which is parasitic in nature and replicates by copying itself to other programs. It does not have inherent automatic replication capabilities and in general cannot exist alone as it is parasitic.

Worm: This type of malware is the most common of all malicious code and causes maximum damage to corporate information. It self-replicates via networks and has the capability to sustain itself. It has inherent replication capabilities using

Malware	Host Required	Replication Mechanism
Virus	Yes	Self
Worm	No	Self
Logic Bomb	No	Manual
Backdoor	No	Manual
Trojan	Yes	Manual
Spyware	No	Manual
Rootkit	No	Manual
Bots	No	Manual

Tbl-01: Malware Properties

inbuilt email or scan engines to identify and spread to other hosts. It exploits vulnerabilities in systems and can also carry other malware as its payload.

A special type of worm called '*Rabbit*' (Aycock, 2006) is also known to exist, which rather than copying moves itself from one system to another.

Logic Bomb: A logic bomb is a type of malware that executes a set of instructions to compromise information systems based on the logic defined by its creator. Logic bombs are usually programs that use either time or an event as the trigger. When the condition(s) stipulated in the instruction set is met, the code present in its payload is executed. It is mostly used by disgruntled employees planning revenge on their employers or by Blackhats hackers for financial gains.

Backdoor: A Backdoor is an alternative entrance into a system. They are used to bypass the existing security mechanisms built into systems. They are commonly created by programmers to test specific code functionality in the least amount of time and are in most cases, accidently left behind. However, they may also be planted by attackers to enjoy continued privileged access into a system once initially compromised. Backdoors are generally standalone non-replicating type of malware.

Trojan / Trojan horse: A Trojan horse or a Trojan is any program that resembles a legitimate program, but has some malicious code inside. It is based on the concept of the Trojan horse in Homer's Iliad. It is a non-replicating code and is generally parasitic as it needs a legitimate program to hide itself.

Spyware: This is a type of malicious code that is used to spy on victim's activities on a system and also for stealing sensitive information of the client. These are among the most popular tools used for Identity thefts, which is a major risk for users who get online from unsecured or public systems.

Rootkit: Rootkits are (set of) programs used to alter the standard operating system functionality to hide any malicious activity done by it. They generally replace common operating utilities like kernel, netstat, ls, ps with their own set of programs so that any of the malicious activity is filtered before displaying results on screen.

Bot & Botnet: A bot is a program that does any action based on instructions received from its master or controller. A network of such bots is called a botnet. Since these are autonomous programs, they are used majorly in the 'dark community' to accomplish many malicious tasks as dictated by its controllers. IRC is one of the common channels that controllers use to communicate with entire botnets.

The following table summarizes the types of malware discussed.

Name	Property	Example(s)
Virus	Copies itself to other files; Needs a host file to propagate and execute.	CIH, Virut, Redlof, Autorun.abt, Peacomm, NewHeur_PE
Worm	Exploits the vulnerabilities that are present and can spread over the network.	Code red, Netsky, Stration, Sasser, Bagle, Skipi, no_virus
Logic Bomb	Triggers a specific code on meeting conditions as per the logic written by its author.	Michelangelo
Backdoor	Listens on certain ports so that the attacker can gain access through them later.	Xhaker, sub7, Beast, Ginwui, Rexob, Hupigon
Trojan	Deceptive program that spoofs a harmless or useful program; but, actually stores other malware.	Limbo/NetHell, Pidief, ZeuS/PRG , Banker.bdn, PGPCoder, Torpig, Gozi
Spyware	Software used to spy on victim's activities and also used to steal sensitive information.	WhenUSave, PuritySCAN Virtumonde, SecurityToolbar
Rootkit	Set of programs that alter the OS functionality to hide themselves.	LRK, AFX, SInAR, Rustock, Mebroot
Bot / Botnet	Program that do the work on behalf of its master. A master may control millions of such bots and can use them for malicious purposes.	Agobot, Slackbot, Mytob, Rbot, SdBot, poebot, IRCBot, VanBot, MPack, Storm

Tbl-02: Malware Types - Summary

1.2 Importance of this paper

1.2.1 Controlling the carriers: Virus and Worms are the only type of malware that have the self-replicating capabilities (Tbl-01) and are the major carriers of other malware. So, by controlling the carriers the threat of malware can be mitigated to certain extent. This paper highlights some of the ways to control these carriers.

1.2.2 Handling the malware threat: A robust incident handling plan and procedures can help in either preventing or mitigating the impact to businesses from various malware. This paper describes some of the processes that can be incorporated in such malware incident handling plans.

1.2.3 Understanding the technologies used: A strong understanding and a solid foundation is required to tackle sophisticated attacks against the corporate assets. This paper gives an overview of the different technologies used in the construction of these and other malware to better the readers understanding of the same.

"If you know the enemy and know yourself, your victory will not stand in doubt." -**Sun Tzu.**

2. SANS SIX STEP INCIDENT HANDLING PROCESS

Before we proceed to handling incidents of various malware, a basic understanding of the process is recommended. In this paper, SANS Six (6) Step Incident Handling process (SANS, 2006) has been selected.

Other Incident Handling processes that exist like NIST SP800-61 for example have the same stages albeit with different names as denoted in the table.

Phase	SANS	NIST	Phase
1	Preparation	Preparation	1
2	Identification	Detection and analysis	2
3	Containment	Containment, Eradication and Recovery	3
4	Eradication		
5	Recovery		
6	Lessons Learned	Post-Incident Activity	4

Tbl-03: SANS & NIST IH Process Comparison

Step	IH Activity
Preparation	The goal of this phase is to get our team ready to handle incidents. Warning banners, response strategies, notification to various parties, IH team building, checklist creation, jump bag[1] creation and emergency communication plans are some of the tasks that are done in this phase. This is the stage where we prepare to fight against all evil.
Identification	The goal of this phase is to identify whether an event is an incident or not by collecting and analyzing all the events happening in the system. Identification can be done at network perimeter level, host perimeter level or at the system level. A provable "chain of custody" must be established before any incident identified is handled.
Containment	The goal of this phase is to contain the incident and prevent its spread to other areas. The different sub-phases in this phase are short-term containment, system back-up and long-term containment. Short-term containment is done to reduce further impact (by disconnecting from network). Long-term containment is done to keep the system in production while a clean system is being rebuilt.

Eradication	The goal of this phase is to remove the infection from the system. This is done with the help of information gathered in the previous stages and by analyzing the cause of the incident (root cause analysis). Eradication is truly possible only if the root cause analysis is properly done.
Recovery	The goal of this phase is to restore services to normal. The system needs to be validated after the restoration process. The business unit should test the system and confirm its complete recovery. The system should be monitored for any undetected malware and also logs should be parsed with extra care to detect any unauthorized activity.
Lessons Learned	The goal of this phase is to document the entire incident handling process. This would help in quicker handling of such incidents next time around and also help in improving the defenses. Incident handling teams need to be trained on handling similar incidents in the future.

Tbl-04: SANS Six Step Incident Handling Process

The following diagram shows the different phases in the incident handling process and the activity done in each phase.

Source: SANS SEC 504 (Incident Handling), Book 1, Page 13

Fig-01: Incident Handling Steps

[1] Jump Bag is a kit with all relevant items used for Incident Handling like audio recorders, software, hardware, disks, hard drives, books and USBs.

3. VIRUS

3.1 INTRODUCTION

Viruses have restricted propagating mechanisms and are parasitic in nature (need other host programs to propagate). Most of them carry a payload that is the action(s) they perform after infection.

When an infected file (any file that has the virus attached) is executed, the virus also gets executed, thereby infecting that system. It does this by making copies of itself and attaching or injecting them into other files available (they are the Matrix/'Agent Smith' in the real world's cyber world).

The impact varies from low to high levels. Most viruses typically destroy specific file types, either by deleting the contents of the file or by encrypting the contents with a random key and corrupting the boot sectors / metadata areas / file system tables (FAT tables in Windows / inode metadata in Linux). Certain viruses merely executes certain instruction sets which in turn could enable or disable certain functionality; slow down all the processes by consuming CPU cycles and even memory. Another category of viruses could disable the existing defense mechanisms such as Antivirus software or firewall thereby permitting other malicious programs to infect the system.

Are they really "Vital Information Resource Under Siege"? Let's take a closer look and find out for ourselves, shall we?

3.2 SUBTYPES and WORKING

Classification of viruses can be done as follows:

1. Memory Based
   (How they live (stay) in memory)
2. Target Based
   (How they spread to others)
3. Obfuscation Technique Based
   (What they do to hide)
4. Payload Based
   (What they do after infection)

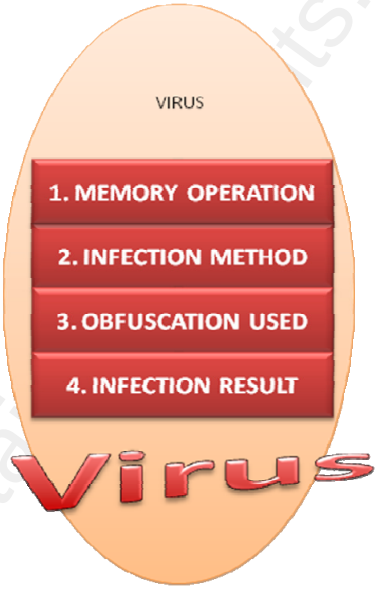


Fig-02: A Virus Model

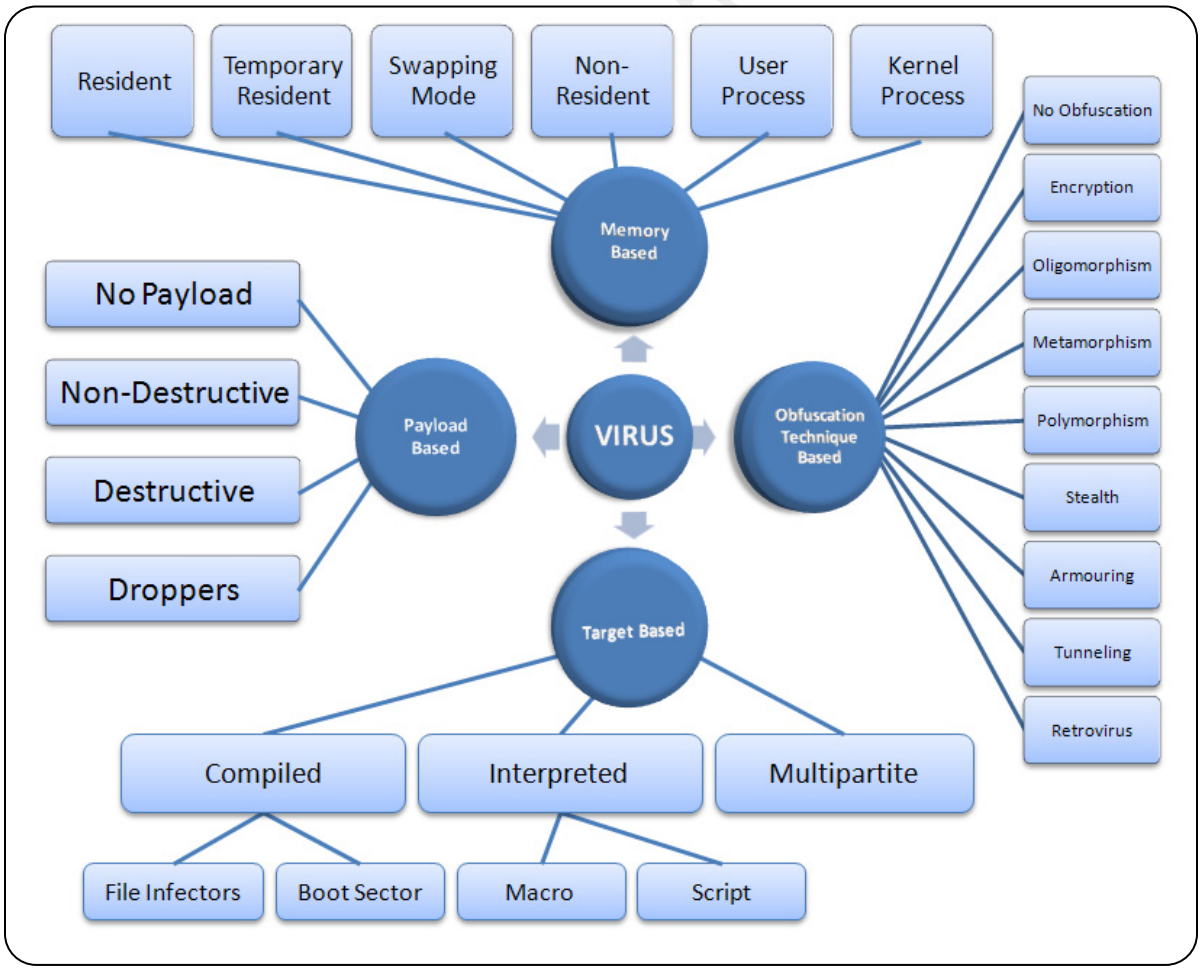


Fig-03: VIRUS Classification

3.2.1 – Memory Based classification

One method of classifying viruses is based on the way they operate in the memory. There are six subtypes according to this classification (Szor, 2005, chap. 5), namely,

1. Resident (In memory)
2. Temporary Resident (In memory temporarily)
3. Swapping Mode (Only a part loaded in memory temporarily)
4. Non-Resident (Not in memory)
5. User Process (As a user level process)
6. Kernel Process (As a process in the kernel)

3.2.1.1 Resident Virus: These types of viruses stay in memory and infect all the relevant files that exist in memory or are in view. The code that is present in the virus is loaded into memory and is copied to all the host files that are running in the memory. A TSR [Terminate and Stay Resident] program is a good example of staying in the memory allocated even after the termination of the main program.

3.2.1.2 Temporary Resident Virus: As the name implies, these viruses stay in memory temporarily and removes themselves out of memory when a certain event occurs. These programs are extremely difficult to detect as the virus activity is encapsulated by the events occurring in the system. Monxla, Antrax are some viruses of this type.

3.2.1.3 Swapping Memory Virus: These types of viruses load a part of their code into memory on occurrence of a certain event and then infect the files present in memory and unload the code from memory. These viruses may be spotted by the increase in disk activity due to loading and unloading of viral code and infection of other host files.

3.2.1.4 Non-Resident Virus: These types of viruses do not exist in physical memory. They have an offline mechanism to search for and infect files present in the hard disk. These viruses contain two (2) key sub-routines. One is the finder or search sub-routine that searches the hard disk for the relevant files to infect. Other is the copy sub-routine that copies the virus code into the files found. If writable network shares are present, these can spread to other systems using them. These are also called *'Direct-action viruses'* (Szor, 2005, chap. 5). VCL, Virdem, Vienna are examples of this type.

3.2.1.5 User Process: These viruses run as a user process and infect the files that are accessible. The virus can exist as its own process. Most of the time, they exist as a sub-process loading before or after the main process. In some of the cases, the virus exist as a DLL and uses DLL Injection method (through registry keys) to load the DLL into the process. Autorun.abt is an example of this type.

3.2.1.6 Kernel Process: These types of viruses generally hook themselves into the kernel through a system driver like program. They have the highest privileges after infection as they are present in the kernel space. These generally infect/modify the IDT [Interrupt Descriptor Table] to get themselves executed every time a particular interrupt is generated. As these viruses require changes to the main file system, they need administrator/super user privileges to run. CIH, Infis are examples of this type of virus.

3.2.2 – Target based classification

Another classification that can be done is based on the target the virus attacks. There are three (3) main types in this classification, namely, compiled, interpreted and multipartite.

3.2.2.1 Compiled Viruses: These are a type of viruses that are compiled into machine executable instructions, so, that they are executed by the Operating System directly.

These are again sub-divided into two (2) sub-categories, namely, File Infectors and Boot Sector.

3.2.2.1.1 Compiled – File Infector Virus: These viruses infect the relevant files present in the system by attaching themselves to the file. These are dependent on the particular file type and platform as they are designed keeping in view the way these files execute. To infect a particular file, the virus program should be able to parse it, copy itself into the program and modify the header to get executed, whenever the program is executed. For this to happen, it needs to understand how the various executables are executed in the operating system. The copying of the virus can be done in different ways, either add itself at the beginning or the end; completely overwrite the file or inject itself wherever there is a gap. Accordingly, there are nine (9) subtypes in this category. They are

1. Appending Virus
2. Prepending Virus
3. Overwriting Virus
4. Cavity Virus
5. Compressing Virus
6. Amoeba Virus
7. EPO Virus
8. Companion Virus
9. Code Virus

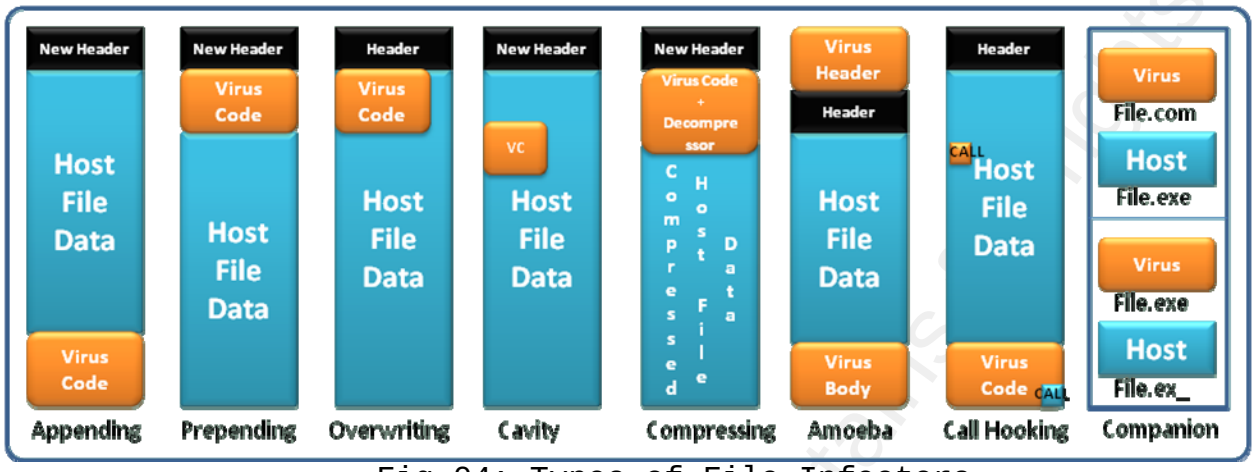
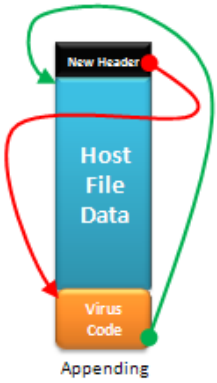


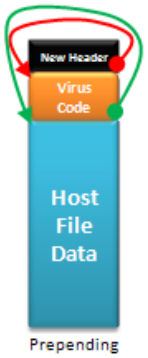
Fig-04: Types of File Infectors

In the following sections, we will discuss briefly about these types along with a diagram showing calls to the virus code execution in red and the calls to program code execution in green.

3.2.2.1.1.1 Appending Virus – This is a type of virus that attaches itself to the end of the host file and modifies the header of the host file so that the control shifts to it on execution. In an appending virus infection, the virus code is appended to the host program and the main entry point of the host program present in the program header is changed to point to the beginning of the virus code. So, when the program executes, the virus is executed first. Then at the end of the virus code, a jump or call routine takes the control back to the start of the host program. Also the new size of the infected host file is updated in the header accordingly. Vienna is an example for this type.



3.2.2.1.1.2 Prepending Virus – This is a type of virus that attaches itself to the start of the executable before the host file content. In a prepending virus



infection, the virus code is inserted in the starting of the program immediately after the header. So, when the program executes, the virus is executed first. Then the control reaches the end of the virus code and passes down into the host program code to execute the host program. The new size of the host file size is updated in the header accordingly. Polimer.512.A, Bliss are examples for this type of viruses.

A special case of prepending virus is the '*classic parasitic* virus' (Szor, 2005, chap. 4), which removes the top of the host program and places its code in the vacancy created. The removed host program code is either appended to the host file or stored in another hidden file. W32/Klez, Qpa are example for this type of viruses.

3.2.2.1.1.3 Overwriting Virus – This is a type of virus that completely overwrites the entire host file that it attacks. The host file is lost and completely modified by the virus to add its code. In an overwriting virus infection, the virus code is overwritten over a portion or entire host program code. If the host file is larger than the virus program, the virus can either remove the whole host program code and replace it with a copy of its own code or overwrite the program code with its own code starting at the initial program code entry. So, when the host program is executed, the control is passed to the starting of the program code (that is overwritten by the virus code) and the virus gets executed. After the execution, the control passes to the remains of the host program code that will not make any sense as the initial part of the code is missing and the host program crashes. If the virus replaces the whole of the host program data segment and is larger than host program, the header needs

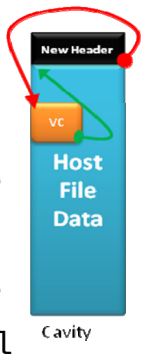
Overwriting

to be modified to reflect the new size. If the virus replaces portion of the host program data segment, there will not be any change in the program size. These are the smallest (just a few bytes) and are mostly destructive. Trivial.22 is an example for this type of virus.

A special case of overwriting virus is the '*random overwriting virus*' (Szor, 2005, chap. 4), which overwrites at a random position in the host program data segment instead of the top part. The virus might or might not get control in this case. Omud virus is an example for this type.

A very complex overwriting virus uses the *embedded decrypter* technique (Szor, 2005, chap. 4). Instead of overwriting with the plain code, these viruses overwrite with their encrypted code. The decrypter dynamically decrypts the encrypted virus on execution of the program. Some viruses also use a *fractured decrypter* that is spread across the host file data.

3.2.2.1.1.4 Cavity Virus – This is a type of virus that injects itself into the gaps/cavities that are found across some of the executables. It is also called 'Space-filler Virus' (*Virus Tutorial*, 2006) or 'Code Interlacing'. In a cavity virus infection, the virus copies itself to one of the cavities present in the executable. It modifies the header, so that the control jumps to its location and once the execution of virus code is over, the control is passed back to the starting of the host program code. Because of this technique, there will be no change in the file size. Lehigh, Darth_Vader, W2K/Installer are examples for this type of virus.

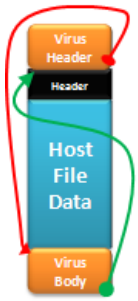


A special case of cavity virus is the *'fractionated cavity* virus' (Szor, 2005, chap. 4), which uses multiple gaps found in the executable. This virus has a head portion that contains information about all parts and their locations in the file. CIH virus is an example for this type.

3.2.2.1.1.5 Compressing Virus – This is a type of virus that compresses the host program and attaches itself. It copies itself to the start of the data segment and includes a decompressing algorithm that is used to decompress the host program and execute it. In this type of infection, the virus compresses the host program using any of the common compressing programs like UPX, ASPACK. Then it adds itself immediately after the header. So, the control passes to it on program execution. Once the execution of virus is complete, it uses the decompression routine present in it to decompress the host program and execute it. This is an attempt to keep the file size as close as possible to the original file size. HybrisF, Aldebera and Redemption virus are some examples of compressing viruses.

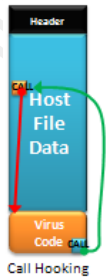
Compressing

3.2.2.1.1.6 Amoeba Virus - Amoeba (Szor, 2005, chap. 4) is a type of virus that copies the entire host program code into its body. In this type of infection, the virus header is located at the top and then the host program is reconstructed and placed after the virus header followed by the virus body. The control from the virus header is transferred to the virus body and then given to the jailed program code. Sand virus is an example of this type of virus.



Amoeba

3.2.2.1.1.7 Entry Point Obfuscation (EPO) Virus – This type of virus changes a random location in the host file data instead of the changing the headers or the initial host file data, so that the entry point of the virus is hidden in the host file safely.



Call Hooking

One such type uses a function call routine to get itself executed. To do this, the virus first scans all the program code for any function or sub-routine calls. It then changes one of the call routine to get control and after execution passes the control to the actual sub-routine. Rainsong and Zhengxi are examples of this type of viruses. Another type of virus inserts itself into the host program code. The control is transferred to the virus via a routine that is embedded in the host program code. After the execution is complete, the control is transferred back to the host program.

Code Inserter

The virus can use multiple obfuscation techniques and fragmented call routines to make detection very difficult. Zmist is an example for this type of virus.

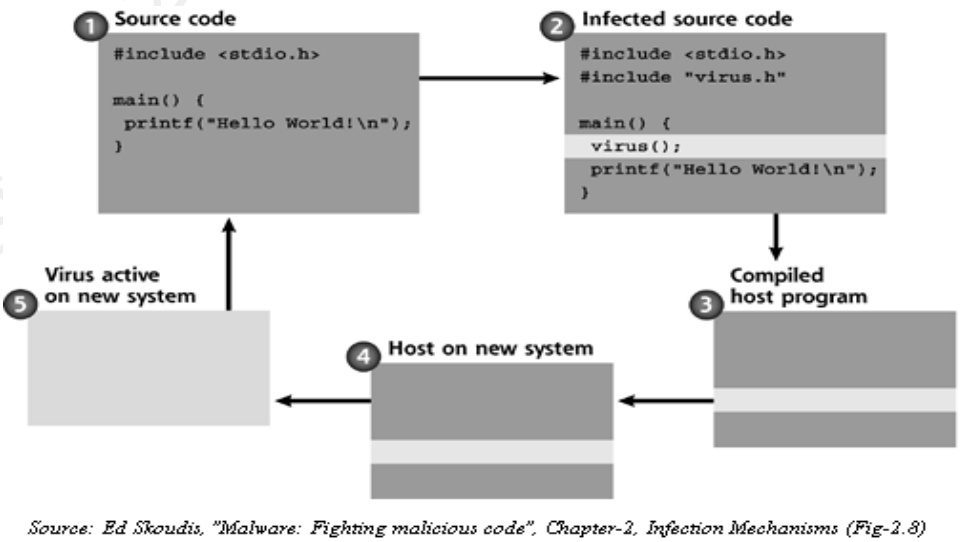
3.2.2.1.1.8 Companion Virus: A companion virus (also called spawning viruses) is a virus that exploits the way the operating system gives preference in execution of different file types. For example, in the Microsoft Windows operating system, COM files get first preference over EXE files (COM, EXE and BAT is the order of precedence). So, if a COM and an EXE file exist with the same name, the COM file is taken into consideration when the user specifies without the extension. Another way of infecting is renaming the original host file to a

Companion

close name (changing only the last character of the extension) and renaming itself to the host filename. So, the virus executes first then passes control to the actual host program. This type of virus is of a special kind as it never infects a host file and exists as a standalone file, which contravenes typical virus behavior. Globes, Trilisa, Win2k.Stream viruses are examples of this type.

3.2.2.1.1.9 Code Virus: This type of virus first creates a hard to understand version of its source code that can avoid detection by simple verification and insert into any source files that are found on the system. The main advantage of this type of infection is the homogeneousness of the executable after compilation. Also these viruses can go undetected by most of the detection techniques. For example traditional infection detection mechanisms like hashing[2], entry point verification fail to detect these types of infections.

The infection is carried in five steps (Skoudis & Zelster, 2003, chap. 2) as shown in the figure. SrcVir, Subit and Urphin are examples for these types of viruses.



Source: Ed Skoudis, "Malware: Fighting malicious code", Chapter-2, Infection Mechanisms (Fig-2.8)

Fig-05: Infection by a Code Virus

[2] - Hashing is the process of generating a small fixed length output from a file that gives the integrity status of the file.

3.2.2.1.2 Compiled – Boot Sector Virus: These viruses infect the boot sectors present in the hard disk. There are basically two types of boot sectors. The Master Boot Record (MBR) is the main boot sector of the hard disk; and every partition in the hard disk also has a boot sector called the Partition Boot Record (PBR).

Fig-06: Hard disk layout

The boot sector viruses infect and stay in the boot sectors. They replace the code present in these boot sectors with their own. Some of the specimens copy the boot sector code to a different location, so that the code is executed after the virus code in the boot sector is executed. Whenever the computer is booted the virus loads itself from the boot sector and infects any other boot sectors (of floppy disks or other devices) and helps in the replication of the virus to other systems.

3.2.2.2 Interpreted Viruses: The viruses that exist in the form of some code that is interpreted by an application are called 'Interpreted viruses'. There are two types of interpreted viruses, namely, macro viruses and script viruses.

3.2.2.2.1 Interpreted – Macro Virus: These viruses use macros to infect and spread to other systems. A 'macro' is a snippet of code present in the document that is executed by the application to make the document more interactive for example, enabling part of document depending on the input. Some of the applications warn about the presence of macros; but, the user is given the choice whether to execute the macro or not. If the user can be tricked into running the macro(s), the virus can push its macro

to the application global macro pool. And, whenever a file is saved this macro is placed in the document. This way they spread from one system to other.

3.2.2.2.2 Interpreted – Script Virus: These viruses use scripts to infect and spread to other systems. A 'script' is a code that exists independently and is executed by the operating system or an operating system service to do some action. There are many languages to write these scripts. The operating system needs a parser to parse through the script and do the action requested. These are mainly used for automation of routine tasks and maintenance tasks. Some of them are used for creating interactive and appealing applications, mainly, the web based.

These scripting languages are used by viruses to infect other scripts and files. They are also used to plant other malware. For example, redlof virus, a VBScript virus infects all web related files (html, asp, jsp, php, vbs) present by appending the encoded script to the files. It also infects system files and executes whenever the infected files are executed.

Some of the scripting languages and file formats vulnerable to script infection are given in Table-05.

Language Name	Extension	Inbuilt / Parser
Unix Shell Script	Sh; bash	Inbuilt (Unix/Linux)
Windows Script	wsf	Inbuilt (Windows)
Perl	pl	Perl
BAT	bat	Inbuilt (DOS, Windows)
Javascript/JScript	js	Inbuilt (Browsers)
VB Script	vbs	Inbuilt (Browsers)
HTML	htm; html	Inbuilt (Browsers)
Executable HTML	mhtml	Inbuilt (Browsers)
Portable Document	pdf	PDF Reader

Flash Action Script	as	Flash Reader / plugin
PHP Hypertext Processor	php	Inbuilt (Browsers)
Active Server Pages	asp	Inbuilt (Browsers)
Java Server Pages	jsp	Inbuilt (Browsers)

Tbl-05: Script files and their extensions

3.2.2.3 Multipartite Viruses: These are viruses that use more than one mechanism to infect the host. They generally infect boot sectors or application documents and use one of the file infection mechanisms to infect files on the host system. These viruses have the capability to infect multiple file types, documents and boot sectors. They also use stealth techniques to avoid detection. As a result these are very efficient and hard to detect. Flip, Invader, Ghostball, Memorial, Junkie, Navrhar are all examples of multipartite viruses.

3.2.3 – Obfuscation Technique based Classification

Obfuscation techniques are those techniques that are used by virus writers to avoid detection and analysis of their specimens (programs). Viruses can be divided into nine (9) subtypes,

1. No Obfuscation
2. Encryption
3. Oligomorphism
4. Polymorphism
5. Metamorphism
6. Stealth
7. Armoring
8. Tunneling
9. Retro

3.2.3.1 – No Obfuscation: Some of the viruses don't use any type of obfuscation technology. It is easier to build a virus of this type. But, detection and analysis of such a virus is trivial as the virus code is readily available once the virus executable is found.

3.2.3.2 – Encryption: This type of viruses use cryptography to hide their functionality. They place a decrypter along with the encrypted body that decrypts the virus on-the-fly.

This decryption function can be a simple XOR function. The decryption of the virus body can happen in forward direction, backed direction or in random order. The decryption key can exist in multiple ways. The simplest one is in the virus body along with the decryption algorithm. In few cases, it is recovered with a simple brute force by the virus itself. Some

viruses can also use crypto API function present in the operating system. Some viruses also generate the keys using various methods like shifting, sliding or fixed random.

3.2.3.3 – Oligomorphism: These viruses are also called '*Semi-polymorphic*' (Aycock, 2006, p.38). These viruses use multiple decryption routines to avoid giving a signature for the antivirus software. The decryption routine is chosen randomly on infection. But, if the antivirus software have signatures for all of the decryption routine, detection is possible.

3.2.3.4 – Polymorphism: These viruses change the look of the virus code every time it infects a new file. This is achieved by changing the decryption routine. These viruses have a very large pool of decryption routines and are much harder to detect using signatures. This high number of decryption routines is possible by the use of a '*mutation engine*'[3], which does all the logic in creating a new decryption routine. The decryption of the virus body can be done using various mathematical functions that forms the base for generation of multiple decryption routines.

3.2.3.5 – Metamorphism: These viruses change the virus body instead of appearance. This is possible by using equivalent and unneeded functions (or code) or by changing the sequence of statements in the code slightly (as long as the logic remains relevant). This way every specimen looks different and generation of a signature is harder. These techniques are mostly used by macro and script viruses. W32.Evol belongs to this type.

[3] Contains sets of equivalent code snippets and takes a code as input and gives code constructed by using other equivalent code snippets

3.2.3.6 – Stealth: A stealth virus is a type of virus that tries to remain undiscovered by hiding the infection events from everyone, instead of trying to obfuscate its code. It achieves this by restoring certain original properties of the host file for example, timestamps. It also intercepts system calls to hide any other resulting changes like the increase in the size of the host file. Other techniques used are creating alternate data streams (NTFS) for infected files with virus in the alternate data streams.

A special type of Stealth virus is '*Reverse Stealth Virus*' (Aycock, 2006, p.37) that makes all the files look infected and are corrupted because of the disinfection process deployed by the antivirus software.

3.2.3.7 – Armoring: An armoring virus is a virus that makes analysis very difficult. These viruses use various anti-debugging, anti-heuristics, anti-goat and anti-VM (virtual machine detection) techniques.

Anti-debugging techniques can be deployed by hooking to various interrupts, using interrupts to generate new decryption keys, through the use of runtime code checksums, checking debugging API routines loaded, checking various registry keys (according to a particular debugger software), using registers and stacks.

Anti-heuristics techniques can be deployed by using file packers, copying itself to multiple sections in the host file and using various EPO (Entry Point Obfuscation) techniques (Szor, 2005). The advantage with packers is the resultant PE (executable) file will not have any of the ASCII strings of the

original executable, hiding the functionality of the virus. Another advantage is generation of a new virus code using a different packer.

Anti-goat techniques can be deployed by identifying goat files. Goat files are those files that are created to get infected by the virus. Generally these files are smaller in size and contain no logic (large number of NOPs (No OPeration) or neutralizing code).

Anti-VM techniques can be deployed by detecting whether they are running in a virtual machine or not. This can be achieved either by looking at VME artifacts in processes, file system, registry and memory or by looking for VME-specific virtual hardware, processor instructions and capabilities.

3.2.3.8 – Tunneling: This technique is mainly used to evade behavior blocking antivirus software. These capture Operating System interrupts. So, whenever these interrupts are made, the virus executes first and after that the control is passed to the original destination. This way they are at a much deeper level in the operating system than the antivirus software and may avoid detection by it.

3.2.3.9 – Retro Virus: A retrovirus (Szor, 2005, chap. 6) is a computer virus that specifically tries to bypass or hinder the operation of an antivirus, personal firewall, or other security programs. These are also called '*Anti-antivirus viruses*' because of these properties. They generally have a database of identification mechanisms for different security controls like process names, registry keys. Once identified, the security controls can be terminated or corrupted. Once the security is

taken down other viruses can enter the system. Some specimens block users from updating their antivirus software or opening of system utilities or antivirus vendor websites.

3.2.4 – Payload based classification

Another method of classifying viruses is based on the result of the infection. There are four subtypes according to this classification, namely,

1. No Payload
2. Non-Destructive Payload
3. Destructive Payload
4. Droppers

3.2.4.1 – No Payload: Some of the viruses present don't do anything than just infecting the files. But, still there can be damage due to non-productivity and loss of reputation. Also, the cleaning process requires money and time that adds to the damage caused.

3.2.4.2 – Non-Destructive Payload: These viruses generally carry a message or a graphic. Some of them just tease the user by controlling hardware like cdrom, speakers. They can be designed to disable certain features like caps lock, special keys. This can be accomplished by changing the states of the keys in the operating system. These can be very annoying at times and most of the time reduces the productivity of the user. For these viruses, damage is only caused by the non-productivity of the user.

3.2.4.3 – Destructive: Destruction is one of the main motives of attackers. Viruses with this kind of payloads are decreasing as there is no financial gain except in few situations that involve rival groups or businesses. In areas where there is a financial gain, more advancement in the virus creation is happening. The

destruction varies according to the virus. Some viruses carry payload that create major catastrophes like destroying partitions by modifying or corrupting metadata. Some have payloads that result in lesser damage like corrupting files in hard disks.

3.2.4.4 – Droppers: Some of the viruses help the attackers in gathering the resources required for conducting malicious activities like identity theft, DDOS, software license theft and phishing. Most of the viruses today belong to this category as there is a huge financial gain. These viruses drop various bots and key loggers that are used to carry these malicious activities. Bots are used to add the victim host machines to a botnet that perform various activities. Few viruses steal software license information from victim's registry, which are later posted to various illegal warez sites.

3.2.5 – The Congregation

Now let's look at how we can understand the design of a virus with the techniques discussed.

3.2.5.1 – A Simple Virus

A simple virus can be designed using just few modules. It uses non-resident (direct action) method to stay in memory. It searches for the relevant files on the disk and infects them. Infection of host files is done using the appending technique. Virus code gets appended to the host file and the header of the host file is modified to pass control to virus on host file execution. The earlier viruses used such techniques. Also simple viruses do not use any obfuscation technique or neither do they carry a payload.

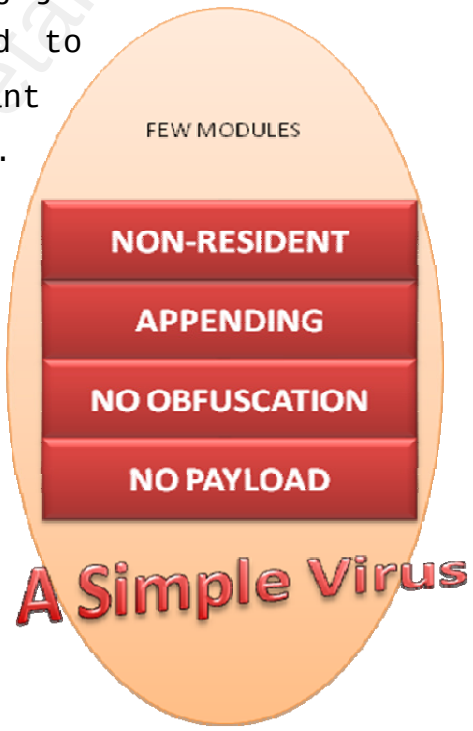


Fig-07: A Simple Virus

1. Where do they live –> Non-resident in memory
2. How do they spread –> Search and append to host file
3. What they do to hide -> Nothing
4. What they do post infection –> Nothing

Using the above parameters, a very small virus can be designed with an overwriting module that overwrites at random / pre-determined sectors on the hard disk.

3.2.5.2 – A Complex Virus

A complex virus contains multiple modules and uses multiple complex mechanisms. It runs as a kernel process that makes it hard to detect. And it is generally multipartite (infects the host in multiple ways). It tries to be stealth, prevent the formation of a signature and also make reverse engineering difficult. For achieving these objectives, it uses multiple obfuscation techniques multiple times. As designing this virus is hard and takes lot of effort, its main purpose is usually for stealing sensitive data for financial gains.

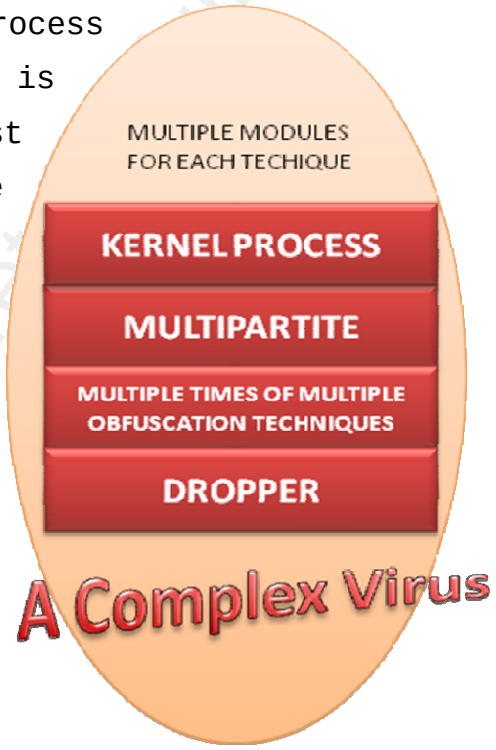


Fig-08: A Complex Virus

1. Where do they live –> In the kernel
2. How do they spread –> Multiple ways of infection
3. What they do to hide –> Multiple hiding techniques
4. What they do post infection –> Steal data from victim

3.3 INCIDENT HANDLING

3.3.1 Preparation

This is the stage where policies, procedures, technology and people are used together for preparation of ways to prevent any incidents arising due to various malware.

3.3.1.1 Policies and Procedures

A policy document is typically a document that outlines specific requirements or rules that must be met. A procedure document is the document that guides the user with the technical process (step by step) on how to achieve the requirements outlined in the policy document. Some of the policies, procedures & activities that often help in preventing the entry of malware and in halting it's spread are the Security Policy, Antivirus Policy, Acceptable Usage Policy, Internet Policy, Email Policy, Desktop Policy, Incident reporting and tracking mechanisms, Incident Handling procedure and periodic audits.

3.3.1.1.1 Security Policy:

A Security policy is a **high level document** from the top management showing the organization's approach towards information security.

According to the ISO 27001 Information Security standard,

"It provides management direction and support for information security in accordance with business requirements and relevant laws and regulations."

The security policy should define how the organization deals with malicious code and should also refer to all relevant sub-policies dealing with the control of malicious code.

3.3.1.1.2 Antivirus Policy:

The Antivirus policy should define what **do's and don'ts** are expected from the users regarding the antivirus (AV) software they are using, including how the AV software needs to be maintained; for normal user machines and also lab machines. A procedure manual should accompany this policy that should guide the users on how to check for the version and new virus definitions and how to keep the software updated. It should also guide users on how to identify if the antivirus is working properly or not.

A guideline on antivirus process is available at the SANS Security Policy page[4], which highlights some of the common tasks that can be performed which goes a long way in making an antivirus more effective and efficient.

3.3.1.1.3 Acceptable Use Policy:

Acceptable use policy[5] should declare to the audience what are considered **acceptable and unacceptable behaviors/actions**, regarding the use of the various corporate resources. It helps in preventing the entry and spread of malware by making the user aware of actions that may intentionally or unintentionally prove risky to the corporate resources.

[4] The guidelines documents is available at www.sans.org/resources/policies/Anti-virus_Guidelines.doc

[5] Some of the policy templates can be downloaded from SANS Security Policy project available at http://www.sans.org/resources/policies/

3.3.1.1.4 Internet Usage Policy:

An Internet Usage Policy is a policy that defines how the user is **expected to use the internet** access that his or her organization has provided. It should also define what is prohibited and associated disciplinary actions for committing the violation. This helps prevent the users from browsing unauthorized site and downloading software from the Internet, which are common entry points of malware into the corporate intranet.

3.3.1.1.5 Email Policy:

The email policy should define how the **corporate email is used**. It should discourage users from using the corporate email for personal use, including publishing and registering in internet groups and forums. This will reduce the amount of spam received by the organizations mail servers and also help reduce the probability of users receiving malicious content via their email.

3.3.1.1.6 Laptop Policy:

The laptop policy should define how the user is expected to **use the allocated laptop** for what precautions the user should take while using the laptop. It should also define what steps the user needs to take to ensure not only the physical security of the laptop itself but also of the information contained within.

3.3.1.1.7 Backup Policy:

The Backup policy should define **what, when and how** information is to be backed up. It should clearly define up to the extent possible what the information is, when and at what intervals it needs to be backed up and how or using what steps. A good backup is sometimes the only way to recover from serious destruction caused by malware infections.

3.3.1.1.8 Incident Reporting and Tracking Mechanisms:

The success behind any incident handling plan is to have a proper incident reporting and tracking mechanism that is **easy to use and effective**. Users generally expect the reporting mechanism to be easily understood and capture the incident with as little information as possible (an option to include detailed information, if present / needed is recommended). Users should also give formal priority levels that can be validated and changed, if necessary by the helpdesk or the central security team.

Names, phone numbers and email numbers to contact in case of a suspected malicious activity should be advertised through all the communication mediums like the corporate intranet site, newsletters and posters around user workstations.

3.3.1.1.9 Incident Handling Procedure and Forms:

The organization must have a proper Incident Handling plans and procedures in place. It should have an Incident Handling form that can **capture detailed information** from all the stages

of incident handing. Sample forms can be downloaded and used from SANS Incident Handling page[6].

3.3.1.1.10 Periodic Audits:

Periodic audits of information systems helps uncover any malicious activity that is present. It can uncover those activities that the user of the systems may not be aware of as the audit teams usually comprise of trained personnel who know what to look for.

3.3.1.1.11 Project based software and processes profiles:

It is recommended to have a **profile of all the software and the processes** that need to be running on the system based on the project or department. This helps in quick identification of any unknown software or processes that might have come into existence due to infection from the malware.

3.3.1.1.12 Knowledge Base:

A good knowledge base with **detailed documentation** and **easy retrieval** can save lot of time when an incident occurs. When an incident happens, all the documentation regarding the handling of the incident should be added to the knowledge base. So, if the same incident happens again, the process can be simply reinitialized. This saves a lot of time that would be consumed in a repeated analysis of the incident. A Root Cause Analysis (RCA) template that can capture most of the details of the incident should be prepared and used.

[6] SANS Incident Handling project page is available at http://www.sans.org/score/incidentforms/index.php. It contains templates for various incidents and can be adopted (with appropriate permission, where required) & customized according to the corporate needs.

3.3.1.2 Technology:

The various technical infrastructure & software that prevent malware include Online Antivirus Scanners, URL and email filters, Virus Submissions URLs, Test Machines (Real machines and Virtual machines), Operating System Utilities and Reverse Engineering Tools.

3.3.1.2.1 URL and email filters:

Almost all organizations today (barring a few military establishments in certain countries) are connected to the Internet for a variety of purposes including email. Connections to the internet and email are the most common paths for malware entering a company's intranet. Such entry must be denied at the network perimeter itself, so that any malicious traffic can be stopped before they enter the corporate LAN. URL filters can help in preventing users from downloading files from the internet that might contain malicious hidden programs. Also, email filters should be deployed to filter any email carrying malicious attachments. Any emails with attachments of the vulnerable file types as given in Tbl-06 should be filtered.

Vulnerable File Types
WIN32
- EXE
- COM
- SCR
- VXD
- DLL
- BAT
- PIF
- ZIP
- OCX
- CPL
LINUX
- SO
- BIN

Tbl-06

There are various free and commercial tools for URL filtering. Squid is a popular and stable open source web proxy that supports URL filtering through the use of lists. SquidGuard can be used to simplify the tasks of URL filtering. It is a combined *filter*, *redirector* and *access controller* plug-in for Squid, which can be used to create access rules according to

time, user groups and URLs. Various blacklists can also be used to do the URL filtering.

3.3.1.2.2 Internet restrictions using lists:

One of the easiest ways to achieve good URL filtering is to **use lists**. There are two types of lists, namely, blacklist and whitelist. A blacklist is a list, which contains all the URLs or sites that are barred. A whitelist is a list, which contains all the URLs or sites that are permitted. They can be referred to as 'Web ACLs'. In a restricted and secure environment, the practice of whitelisting is recommended. However, to create a whitelist, all the URLs that are needed for conducting business first have to be identified. If this list is finite, then using whitelists is the best way forward. If the users use the Internet through search engines, then whitelists cannot be created. In such cases blacklists will have to be created. This list should contain all the URLs that are to be blocked. This list is first checked by the web proxy before allowing access and if an URL is not found in the list, only then may it be allowed.

3.3.1.2.3 Disabling use of removable devices:

Most of the malware authors today have developed techniques to copy viruses to any removable devices and have them execute immediately on a fresh connection to a system. It is recommended to **disable all removable devices**, if there is no business requirement. This may be achieved by physical removing cable connections on the motherboard, disabling onboard ports (USB, Bluetooth, IR) in the BIOS and also at the OS level using GPO (Group Policy Objects) in windows and access restrictions in Linux (as all devices are also files). Sometimes disabling USB

ports at the BIOS level may not be feasible, if the system uses a USB keyboard and mouse. This problem can be overcome by using OS level restrictions (Moskowitz, 2007; Petri, 2007). There exists few products that are created for this purpose (like PointSec, Safend, Safeboot), which have much better efficiency and features than the native methods of blocking as discussed above.

3.3.1.2.4 Hashes of system files:

Another important step in the preparation stage is the collection of **hashes for important files**, mainly system files. So, if the machine behaves abnormally or a malware infection is suspected, the modified files can be detected by comparing their hashes with the pre-recorded hashes of the original. These files can also be checked for any malware infections using known online antivirus scan services or can be submitted to the antivirus vendor for analysis.

3.3.1.2.5 Host based Intrusion Detection System:

One easy way of checking for any changes to system files is to use a Host based Intrusion Detection System (HIDS). HIDS initially calculates the hash of all system files and keep it in a database. The hash of the file changes whenever any system file is modified. This way any unauthorized changes can be identified. Another way is to alert on any calls made from ring3 to ring0, which is not normal. They also check for any hidden processes, parse logs for suspicious activity. There are many free and commercial HIDS software. Open source software like samhain, OSSEC and Osiris are some of the client server based HIDS.

3.3.1.2.6 Antivirus:

Organizations must have an antivirus in place, mainly for all those systems that have either an Internet connection or Removable devices (USB, writeable DVD drives etc.) enabled. It is recommended to have a **client-server model** that is much easier to manage. Status of the working of the antivirus clients, remote installation of clients and remote scanning on systems are some of the advantages of using a server based solution. If in-house skills are not present, a managed antivirus model can be opted for.

3.3.1.2.7 Online Antivirus Scanners:

There are two types of online antivirus scanners, each for a slightly different purpose.

3.3.1.2.7.1 File Scanners: Once a malicious file or a malware **infected file** is identified, it can be scanned using multiple antivirus engines available online. This is useful, if you suspect the malware executable was not getting identified by the current antivirus engine. It may also happen that the malware will go undetected by a few AV engines[8] as no antivirus can detect all of the existing malware at any given time. If the malware is detected by any of the antivirus engines, the incident handling becomes easy.

Service	Engines	URL
VirScan	36	http://www.virscan.org
VirusTotal	32	http://www.virustotal.com
VirusScan	21	http://virusscan.jotti.org
VirusChief	10	http://www.viruschief.com

Tbl-07: Online Multiple Engines Scanning Services

[8] AV Engine detection statistics available from www.virustotal.com/estadisticas.html

Some online websites that provide free scanning using multiple antivirus engines are provided in the table above.

3.3.1.2.7.2 System Scanners: These scan the **entire system** for the presence of malware. This is useful, if the system is completely infected and the software installation is not possible. This can be done either to identify the malware or to check for the success of the eradication process. In this method, the antivirus engine is downloaded followed by the virus definitions file. These will be done automatically using ActiveX technology.

The limitations with these scanners are they are browser dependent and cannot scan the entire malware spectrum.

AV Engine(A-Z)	URL
BitDefender	http://www.bitdefender.com/scan8/ie.html
eTrust	http://www3.ca.com/securityadvisor/virusinfo/scan.aspx
Ewido (AVG)	http://www.ewido.net/en/onlinescan/
Kaspersky	http://www.kaspersky.com/virusscanner
McAfee	http://us.mcafee.com/root/mfs/default.asp
Panda	http://www.pandasoftware.com/activescan/activescan/
Trend Micro	http://housecall.trendmicro.com/

Tbl-08: Online Antivirus Scan URLs

3.3.1.2.8 Virus Submissions URLs:

If new malware is detected but cannot be identified or removed, it can be **submitted to the antivirus research labs** for analysis. If at least one of the antivirus engines in the online multiple engines services detect the malware, it will be automatically sent to all other antivirus research labs for analysis. The malware submission URLs of some of the popular antivirus and antimalware companies are listed in table Tbl-09.

Company	URL
ClamAV	http://cgi.clamav.net/sendvirus.cgi
F-Secure	http://www.f-secure.com/samples/index.html
ThreatExpert	http://www.threatexpert.com/submit.aspx
McAfee	http://vil.nai.com/vil/submit-sample.aspx
Sophos	http://www.sophos.com/support/samples/
Symantec	https://submit.symantec.com/websubmit/retail.cgi
Sunbelt	http://research.sunbelt-software.com/Submit.aspx

Tbl-09: Online Malware Submission URLs

3.3.1.2.9 Virus Removal Tools:

Another must have component in the toolkit are the removal tools from various antivirus companies for various malware. Removal tools are effective, efficient and easier to work than the full antivirus engines. But, they are limited to mostly one family of malware. McAfee Stringer is a removal tool for a group of malware. Instructions on using the tools must also accompany the tool as some of the tools need certain requirements to work effectively. Removal tools from some of the popular antivirus companies can be downloaded from the URLs in table Tbl-10.

AV Vendor	URL
BitDefender	www.bitdefender.com/site/Download/browseFreeRemovalTool/
F-Secure	www.f-secure.com/download-purchase/tools.shtml
Kaspersky	www.kaspersky.com/removaltools
McAfee	us.mcafee.com/virusInfo/default.asp?id=vrt
McAfee	vil.nai.com/vil/stinger/
Microsoft	www.microsoft.com/security/malwareremove/default.mspx
Symantec	www.symantec.com/business/security_response/removaltools.jsp

Tbl-10: Virus Removal Tools Download URLs

3.3.1.2.10 Test Machines:

Test machines are those systems, ideally isolated, where the malware is allowed to run and simultaneously or subsequently analyzed. Virtual machines software like VmWare, MS VPC, Xen can

be used to create virtual machines for such analysis. Virtual machines save lot of time in setting up the test lab and also in restoring to a previous or uninfected state. However, having a few physical machines is also recommended as most of the sophisticated malware use virtual machine detection techniques. If the malware is sophisticated enough and identifies the virtual machines, it may either become dormant or may destroy the virtual machine. We can overcome this either by 'Tweaking virtual machines' or by patching (replacing the code doing the check with NOP instructions) the malware. If these steps also fail, the only option is to use physical machines. And for making the job easier, a disk to disk imaging solution like Symantec Ghost will come in very handy. Setting up the lab and restoring to a clean state is just a few minutes of work compared to the hours of work involved in installing the operating system, drivers, utilities and tools.

3.3.1.2.11 Operating System Utilities:

When a virus outbreak happens, the utility programs present in the operating systems are crippled. In such situations, a non-infected source can be very helpful. Native operating utilities along with third party utilities can be copied to read only media like CD-ROMs or DVD-ROMs, so they don't get infected when they are run. For Microsoft Windows, SysInternals hosts a lot of powerful and simple utilities. These can be downloaded free of cost and added to the "Utility Toolkit". These can be used to collect samples of malware for analysis or to identify, contain and eradicate the malware.

3.3.1.2.12 Reverse Engineering Tools:

For analysis of malware, we need to have a "Reverse Engineering Toolkit" to reverse engineer the malware sample. Executable analysis tools like PEInfo, PEiD ExeInfo, BinText can give some initial information about the malware executable like the packing algorithm used or the strings found. Then accordingly the various unpackers can be used to unpack the executable and the unpacked executable can be analyzed using reverse engineering tools like IDA. Sometimes, when the unpackers are not available or an unknown algorithm is used, dynamic or runtime analysis is used. For this analysis, a debugger is required. OllyDbg and Immunity Debugger are some of the best debuggers that exist at present. Softice is another commercial alternative. Some of the malware comes with a debugger detection routine. If a debugger is present, it will either destroy the operating system or goes dormant (few may self destroy) to oppose any analysis of its executable.

Many of the tools in the SysInternal Suite help in various stages of analysis. Process Explorer, Process Monitor, File Monitor, Registry Monitor, Streams are few of the "must have" tools in any reverse engineer's toolkit.

Debuggers	OllyDbg, Immunity Debugger, Softice
Disassembler	IDA Free/Pro
Unpackers	Unpckarc, upx, aspackdie and others
PE Analysis	PEInfo, PEiD, Exeinfo, BinText, SysAnalyzer, LordPE
Utilities	SysInternals, HijackThis
Misc	Regshot, MAP, WinHex

Tbl-11: Reverse Engineering Tools

3.3.1.3 People:

Even though the technical and process controls are robust, security can be compromised by exploiting people and making them do actions that are otherwise not permitted. Skilled Incident Handling Teams and the Incident Handling Escalation Matrix constitute key components of an effective handling & containment strategy. A good incident handling team is an incredibly valuable resource when it comes to handling any malware situation that may arise, in an efficient and effective manner. As people are the main organizational resource that are eventually harmed by malware infections, Security Awareness is one of the key (people based) issues that need to be constantly monitored and improved for proper protection from various attacks.

3.3.1.3.1 Security Awareness:

This may be considered as the **most important** of all the preparation measures. It helps in identifying and preventing most of the problems. It educates the user on how to protect the information, what to do and what not to do, whom to call in emergency and how to analyze if an action can land them in trouble.

3.3.1.3.2 Incident Handling Escalation Matrix:

Every organization must have an Incident Handling Escalation Matrix that clearly **defines who should be contacted** in case of an incident. It also shows the escalation level for further involvement according to the complexity or impact of the incident.

3.3.1.3.3 Skilled Incident Handling Team:

A knowledgeable incident handling team can reduce the business impact to a great extent. The incident handling team should posses an excellent understanding & skill levels in the various technologies used by the enterprise. Since, many enterprises have offices located in different geographic areas, a central command team and local / regional teams is recommended, where appropriate. The Central command team of course, should guide the local teams in handling the incidents.

"The enlightened ruler lays his plans well ahead; the good general cultivates his resources." -Sun Tzu.

3.3.2 Identification

This is the stage where malware identification and confirmation of its presence is conducted using different techniques. The following is a list of either tell-tale signs or behavior observed or methods of identification which can help confirm the presence of malware.

3.3.2.1 Antivirus NOT functioning as expected:

Some viruses (also known as Retro viruses) destroy the existing antivirus installation by corrupting the executable, changing registry keys or corrupting definition files. Other viruses may disable the update of the signature file. One way to do this is by changing the 'hosts' file of the operating system. This file is used by the operating system for name resolution and has higher preference than that of a name server resolution. It is the file that does local name resolution.

The host file is *C:\windows\system32\drivers\etc\hosts* in MS Windows and */etc/hosts* in Linux. A virus can add a line to this file to disable all online updates of any software. If a line such as "*127.0.0.1 avupdate.av_vendor.com*" is added by the virus, all requests to the antivirus update definitions website will resolve to the local system and will subsequently fail. So, if the antivirus is found to be working properly but is not receiving updates, checking the 'host file' for a bogus entry might help solve the problem.

If a virus can capture these requests and reply with its version of signature files, the virus can easily evade detection by the antivirus.

3.3.2.2 Unusual / Unfamiliar Files:

Certain viruses are known to create unusual files in the root and system directories. These files have names that may tempt a user into copying and executing on other systems. Such filenames may include the next versions of popular software or adult content. On clicking some of these, viruses create autorun files in the directories and drives. These files ask the operating system to execute the virus file immediately on connecting the device or opening the folder. This way, if the device is connected to another machine, it gets executed immediately without the requirement of user executing the infected file manually.

3.3.2.3 Files with double extensions:

One good way to trick users into executing a malicious executable is by using double extensions. In the windows operating system, only the last extension is taken as the file extension; and the remaining name is taken as the file name. By default, known extensions are hidden. So, known extensions like exe, com, scr are all hidden. So, when a file *filename.jpg.exe* is downloaded, the user sees *filename.jpg* as the file downloaded. If the icon is replaced with a jpg icon, the user can be deceived easily. The user thinks that he/she has downloaded a jpg file and tries to open it by clicking it. Therefore either the 'Hide extensions for known file types' option in the folder properties should be disabled or the user should check the file if he/she sees a known extension in the file name. This type of infection mechanism is commonly used to spread viruses, mostly through warez sites.

3.3.2.4 Unknown Processes:

Some of the malware (Section 3.2.1.5) start certain processes that help in either staying stealth or in spreading to other files. Generally these processes have names that are similar to system processes names like svchost, smss, lsass to avoid easy identification. However, these processes can be identified by looking at the process owners and the executable the process is attached to. The malinfo.bat script (Appendix B) can be used to confirm the presence of malware as it gives the processes running in the system along with the executables it is attached to.

3.3.2.5 Failure to open system utilities:

Some of the viruses try to hide their presence by stealth, either preventing users from identifying their components or terminating their processes. The 'Task Manager' is the most common system utility in Microsoft Windows. Other tools like SysInternals Process Explorer are also popular. Most viruses disable these utilities by closing or minimizing them, if opened. Viruses have even been known to disable other configuration utilities like the control panel, folder options and even the command prompt.

These utilities are immediately closed, if opened or corrupted so they can't open. This way any process started by them will be difficult to identify. One way this is accomplished is by the use of a 'killer' process that maintains a database of windows, websites (URLs) or keywords and kills any such process as preprogrammed by the virus author.

3.3.2.6 Slow CPU Response:

At times due to virus activity, the user might sometimes experience a slow response from the CPU and the system may hang for few seconds in between different tasks. as the reason being the virus is consuming most of the CPU cycles for its infection activity. If all of a sudden, one fine day the computer starts behaving slowly, there might be a chance of an infection. You would need to verify if any running process is resulting in this abnormal behavior and if required other identification procedures should also be used to confirm the presence of the malware.

3.3.2.7. Unexpected events:

Sometimes applications might automatically exit or new windows start popping up on booting; randomly or periodically when in use, due to the active presence of malware. When such events are analyzed, care should be taken to eliminate false positives arising due to the installation of new software or the use of clashing utility programs.

3.3.2.8. System / Application crashes:

System and application executables may sometimes get corrupted due to virus infections. When the application is started, the infected executable is run. The executable might not run properly and may crash because of the changes in the code. Similarly, if operating system executables are infected, whenever the executable is run, that processes might crash [which sometimes may even crash the operating system].

3.3.2.9. Alerts from peers:

Sometimes when the virus tries to spread to other systems with better security levels (user with better security awareness or updated security software) they might be spotted. These instances include attacks to other systems when the infected files are copied to them or emails with unknown attachments are received. When the source is found, the user of the source is notified.

3.3.2.10. Information security forums:

One way to identify new malware is by checking various security forums for newly released viruses and their symptoms. If any similar symptoms are found in the network, then further investigation can be carried out with the information available on the forums.

Forum / Site	Forum URL
SANS ISC Handler's Diary	http://isc.sans.org/diary.html
Stay Safe Online	http://www.staysafeonline.info/
Security Focus	http://www.securityfocus.com
US-CERT	http://www.us-cert.gov/
FrSIRT	http://www.frsirt.com/english/
Packetstorm	http://www.packetstormsecurity.org/
The Register	http://www.theregister.co.uk/security/
TrustedSource	http://www.trustedsource.org/
McAfee	http://vil.nai.com/vil/default.aspx
Dark Reading	http://www.darkreading.com/default.asp
Symantec	http://www.symantec.com/enterprise/security_response/weblog/
AusCERT	http://www.auscert.org.au/
Talisker	http://www.securitywizardry.com/radara.htm(all in one place)

Tbl-12: Security Forums and News Sites

NOTE: While some of these events might occur sometimes, it of course does not always mean that malware is present.

3.3.3 Containment

This is the first active phase which involves changes in the environment to stop or literally contain the spread of malware. Methods used could include isolating the infected system from the network. Also, a prudent move would be to take a complete backup of the system for analysis later as well as recovery of data to the maximum extent possible, probably at a later stage depending on it criticality.

3.3.3.1. Permission for Containment

The first thing after confirmation of existence of malware of the malware is to notify the appropriate personnel and take necessary permissions to isolate the system. Permission from the respective business units is critical as the impact due to isolation is imminent and the owner of the system needs to be notified of the situation.

3.3.3.2. System Isolation

Once permission is obtained, the infected system is isolated. Isolation can be done either by physically disconnecting the system (can also be achieved by disabling the network card) or quarantining the system from the network by moving the system into a different VLAN. Remember to save the network connection information present on the system before disconnecting from the network which would enable you to do a complete analysis.

3.3.3.3. Check for Similar Symptoms

Once the basic symptoms are recorded, other systems present in the network need to be checked to see if they are exhibiting similar symptoms. If positive, those systems are also to be isolated and analyzed for existence of malware.

3.3.3.4. Check the Past Incidents (Knowledge Base)

The next step after identifying the basic symptoms of the malware is to search the knowledgebase that contains all the incidents that have occurred in the past. If the incident is a repetition, the procedures followed previously are to be executed after a thorough analysis of each step to identify the reason for reoccurrence of the incident and ascertaining whether such steps are adequate or if the procedures require an overhaul in their entirety.

3.3.3.5. Backup of all User Data

Before entering the eradication phase, all user data present is taken as a backup and kept isolated from other similar backups as it might be infected with malware components. This is to retrieve any lost data, if possible after complete analysis of malware. Once the malware analysis is successfully done, all the malware components present in the backup can be removed and the user data can be recovered up to varying amounts and in rare cases , completely.

3.3.4 Eradication

This is the stage where different techniques are used to analyze the malware and clean the malware from the infected systems. Once the infected files are identified, the symptoms of the malware are carefully noted and the malware executables identified are analyzed. After the analysis, all the malware executables and artifacts (dropped or downloaded items) left by the malware are removed and the holes that allowed the infection are patched.

3.3.4.1 System Files Integrity Check

All system files are checked for any unauthorized or unwarranted modification (Integrity check). This can be done by comparing the hashes of these files with their previously collected hashes thereby identifying the files that were infected by the malware.

3.3.4.2 Identify Newly Created Files

Most malware create new files, which help it in accomplishing its task locally, spread to other systems and make cleaning them difficult. To properly eradicate the malware, all these files must be identified and removed from the system.

3.3.4.3 Identify any other symptoms

To properly eradicate and also to identify the infection in future, all symptoms of the malware must be identified. This is achieved by careful observation of either the infected system or a test system infected with the sample collected. Some of the

malware that are released have virtual machine detection features and some also have anti-virtual machine capabilities[9]. Most of the malware with such features turn off some of their characteristics to avoid revealing their symptoms to AV researchers. Behavioral analysis techniques need to be employed to identify all the symptoms of such malware.

3.3.4.4 Analyze the files

In this activity, the malware executables that are collected by the previous activities are thoroughly analyzed. This is done by reverse engineering the executables using disassemblers, debuggers and utilities (Section 3.3.1.2.12). This facilitates identifying the inner functionality of the malware and may guide us in the process of identification and cleaning the malware. It also helps in adding to the list of malware symptoms collected so far.

3.3.4.5 Network Checks

Once all the symptoms are collected, the prevention mechanisms are developed and implemented. Using these symptoms any traces of the malware on other systems in the network are identified. If found, these systems are also handled according to the process derived. For example, if the virus is a dropper and drops a bot or a backdoor, network scans for the open malware port or firewall logs showing suspicious traffic needs to be analyzed.

[9] some malware (virtual machine detecting) shuts down their services, if they identify a virtual machine. Few malware with anti-virtual machine techniques destroy the virtual machine, if found (Storm Worm).

3.3.4.6 Check Backups

Next is to take a recent working backup and check for any traces of the malware. After confirming the backup set to be clean, it is restored and any lost data is added.

3.3.4.7 Finding the Cause

This is the most crucial activity and also one of the toughest activities in the eradication phase. The cause of the incident (or infection) is to be found; so that the incident will not occur in future. To do this, the logs of the system, proxy servers and perimeter devices are to be checked as applicable. System logs include event logs, antivirus logs and logs from any other security controls (Software or devices) present. These may sometimes possess evidence of any unexpected or malicious activity that happened previously. Proxy logs can be used to check if the source of infection is from the Internet by reviewing the URLs visited. Email server logs can be checked if an email carried the malware inside the network. Perimeter device logs can also be checked for the traces of the entry.

3.3.4.8 Improving Defenses

After the cause of infection is found, the next step is to strengthen the defenses and prevent the malware from entering again. This is done by modifying access rules at the perimeter devices, filtering emails with particular words or attachments, blocking certain URLs or file types and removing access to certain devices like USBs and DVDs.

3.3.5 Recovery

In this stage, the recovered systems are validated by the application user and decisions are made regarding when to restore the systems complete operation. The system is also kept under observation in this phase, to check for any malware components that escaped detection during the previous phases.

3.3.5.1 System Validation

The recovered systems are validated against any mis-configuration or deficiencies. If any deficiency of software or data is found, it is added. A user sign off is taken confirming the complete recovery and the normalcy of the system.

3.3.5.2 Restoration of Operations

Once the validation of the recovered system is complete, the owner of the system decides when to put the system back online. Recommendations regarding the security of the system may be given to the owner of the system. The owner should acknowledge these recommendations through a signed memo.

3.3.5.3 Monitoring the System

The final and important activity in the recovery phase is to monitor the system carefully for any new attacks. Sometimes, the analysis done in the previous phases might not have revealed all of the malware executables still present in the system. These stealth malware executables will try to infect the system once again and careful monitoring can help identify any such components left behind.

3.3.6 Lessons learned

This is the documentation phase where all the activities that are carried out are recorded for future reference. This phase gives inputs to the preparation phase to improve the defenses.

3.3.6.1 Additions to the Incident Handling Knowledgebase

One of the essential things to do after successful handling of an incident is updating the knowledgebase. This report should be added to and reviewed by all the involved parties. This would help in handling similar incidents in the future easily, efficiently and quickly.

3.3.6.2 Antivirus Signature Creation and Inclusion

If the malware is not detectable by the antivirus, the malware samples and the analysis done should be sent to the antivirus vendor. Once the signature is created, all the antivirus clients should be updated with the new signature files, which will make them, detect and hopefully remove the malware successfully.

3.3.6.3 Training to the Incident Handling Team

The handler or handler team should train all other handler in the team on handling this malware incident. This would help them better understand the incident handling process and also help in tackling any similar incidents in the future more skillfully.

3.3.6.4 Updating Filtering Rules

All the ingress paths of the malware identified should be appropriately blocked to prevent malware from entering into the network in the future. This may be done by adding new rules in the perimeter and other filtering devices (like URL filters, email filters, IDS).

3.3.6.5 User Education and Malware Identification

All information regarding identification of the malware should be published in the company newsletter. In this manner, the users will be aware of different malware symptoms and can report the same to the helpdesk, if spotted.

3.3.6.6 Improving the Defenses Accordingly

Once the handling is complete, the Root Cause Analysis is used to harden the various security controls present in the company. The technical teams can be made aware of the symptoms of the malware to check for similar entities, the incident handling team can be given similar incidents to practice and the management can introduce new security controls to mitigate such risks in the future.

4. CONCLUSION

Incident handling due to malware infections need a lot of **preparation**, **patience** and **persistence**. Preparation is to prevent the entry of malware into the network and to clean if they enter into the network. Patience is needed to formulate an effective strategic solution instead of a quick and hasty step. Persistence is needed to continue analyzing the malware sample until you succeed, even if it is designed to be complex and hard to analyze.

This paper gave the reader an idea of the different types of known virus that exist at present, how they are designed and what properties they exhibit. This knowledge will help the incident handler better his or her understanding on the type of malware being handled and the way it behaves in the environment. It also describes the activities in the incident handling process for malware incidents.

Any feedback and suggestions to improve the process is welcome as this helps all of us to fight the evil doers and help provide a safer digital environment to all concerned.

Hoping for a safe cyber world ……

5. REFERENCES

5.1. Books, Articles & Presentations

5.1.1. Aycock, J. (2006). *Computer viruses and malware*. Springer.

5.1.2. Computer Knowledge. (2006). *Virus tutorial*. Retrieved April 12, 2008, from http://www.cknow.com/vtutor/index.html

5.1.3. Filiol, E. (2005). *Computer viruses: From theory to applications.* Springer-Verlag.

5.1.4. Moskowitz, J. (2007). *Managing hardware restrictions via group policy*. Retrieved April 12, 2008, from www.microsoft.com/technet/technetmag/issues/2007/06/GroupPolicy/default.aspx

5.1.5. NIST. (2004). *Special publication SP800-61: Computer security incident handling guide*. Retrieved April 12, 2008, from csrc.nist.gov/publications/nistpubs/800-61/sp800-61.pdf

5.1.6. NIST. (2005). *Special publication SP800-83: Guide to malware incident prevention and handling*. Retrieved April 12, 2008, from csrc.nist.gov/publications/nistpubs/800-83/sp800-83.pdf

5.1.7. Petri, D. (2007). *Disable USB disks with GPO*. Retrieved April 12, 2008, from www.petri.co.il/disable_usb_disks_with_gpo.htm

5.1.8. SANS. (2006) *Security 504: Incident handling step-by-step and computer crime investigation (Book 1).* SANS Institute.

5.1.9. Skoudis, E. & Zelster, L. (2003). *Malware: Fighting malicious code.* Prentice Hall PTR.

5.1.10. Szor, P. (2005). *The art of computer virus research and defense.* Addison Wesley.

5.2. Internet (Multiple pages/references)

5.2.1. SANS Internet Storm Center. (2000). SANS Internet Storm Center. Retrieved April 12, 2008, from http://isc.sans.org

5.2.2. SANS Sample Policies. (2000). SANS Sample Policies. Retrieved April 12, 2008, from http://www.sans.org/resources/policies

5.2.3. OpenRCE Forum. (2005). OpenRCE Forum. Retrieved April 12, 2008, from http://www.openrce.org

5.2.4. Worm Blog. (2004). Worm Blog. Retrieved April 12, 2008, from http://www.wormblog.com

5.2.5. Kaspersky Virus Encyclopedia. (1996). Kaspersky Virus Encyclopedia. Retrieved April 12, 2008, from http://www.viruslist.com

5.3. Further Study

5.3.1. ERESI Reverse Engineering Software Interface project. (2001). ERESI Reverse Engineering Software Interface project. Available April 12, 2008, at http://www.eresi-project.org/

5.3.2. Malware Collection. (2006). Malware Collection. Available April 12, 2008, at http://www.mwcollect.org/

5.3.3. Sunbelt CWSandbox. (2007). Sunbelt CWSandbox. Available April 12, 2008, at http://www.cwsandbox.org/

5.3.4. Norman Sandbox Malware Analyzer. (2006). Norman Sandbox Malware Analyzer. Available April 12, 2008, at http://www.norman.com/microsites/malwareanalyzer/

5.3.5. CSRRT Malware Sandbox. (2006). CSRRT Malware Sandbox. Available April 12, 2008, at http://www.csrrt.org.lu/wiki/index.php/Malware/CSRRT_Sandbox

5.3.6. Huge list of Unpackers and other resources available at http://www.exetools.com/unpackers.htm

5.3.7. SANS Incident Handling process. (2007). SANS Incident Handling process. Available April 12, 2008, at http://www.giac.org/resources/whitepaper/network/17.php

5.3.8. SANS Incident Handling sample forms. (2003). SANS Incident Handling sample forms. Available April 12, 2008, at http://www.sans.org/score/incidentforms/index.php

5.3.9. Liston, T. & Skoudis, E. (2006). *Thwarting VM Detection*. Available April 12, 2008, at http://handlers.sans.org/tliston/ThwartingVMDetection_Liston_Skoudis.pdf

5.4. URLs of Software mentioned

TOOL / SOFTWARE	URL
ExeInfo PE	http://www.exeinfo.go.pl/
IDA Pro/Free	http://www.hex-rays.com/idapro/
Immunity Debugger	http://www.immunitysec.com/products-immdbg.shtml
OllyDbg	http://www.ollydbg.de/
Osiris	http://osiris.shmoo.com/
OSSEC	http://www.ossec.net/
Process Explorer	http://www.microsoft.com/sysinternals
Samhain	http://www.la-samhna.de/samhain/
SquidGuard	http://www.squidguard.org
Virtual PC	http://www.microsoft.com/windows/products/winfamily/virtualpc/default.mspx
VMWare	http://www.vmware.com
Xen	http://www.citrixxenserver.com/Pages/default.aspx

APPENDIX – A: BOOT PROCESS

BOOT PROCESS	
Power – When the system is switched on, power reaches the motherboard through SMPS	
BIOS – BIOS present on the motherboard is activated; Does the POST check; then check for devices connected and passes control to the relevant device (boot device) for the next stage of booting.	
MBR – MBR of the boot device gets activated and checks for any boot loaders or active partitions. If a boot loader is present, control is passed to it. Else the control is passed to the active partition specified.	
Active Partition BR – The boot loader of the active partition is activated when it gets control.	
MS WINDOWS	**LINUX**
NTLDR (NT Boot Loader) in the system volume is loaded and passed the control [SYSPART:\ntldr]	GRUB Stage 1(a small machine code binary enough to fit in a boot sector) is loaded from the boot sector whose purpose is to load the next stage boot loader
NTLDR reads the 'boot.ini' in C drive. If more than one OS is present, a choice is requested. Else it continues booting from the boot partition as found in the boot.ini file. [SYSPART:\boot.ini]	It then loads the GRUB Stage 1.5 located in the first 30 kb of the partition after the boot sector. This stage may or may not be present in some cases.
Then NTDETECT from the system partition is loaded which is device detection program. [SYSPART:\NTDETECT.COM]	This then loads GRUB Stage 2, which presents the graphical screen with options to load a particular operating system.
It then loads NTOSKRNL (Kernel), HAL (Hardware Abstraction Layer) from the boot partitions. [%systemroot%\system32\ntoskrnl.exe and %systemroot%\system32\hal.dll]	It then decompresses the kernel and loads it. Init ram disk also gets decompressed and loaded.
Then SYSTEM Hive is loaded and all boot drivers is loaded. [%systemroot%\system32\config\system]	The kernel then checks all the hardware and loads the respective drivers for the devices found.
After that the boot loader (NTLOADER) passes control to Kernel (NTOSKRNL)	Then the root file system is mounted as per the parameters in /etc/fstab
Kernel then loads the logo screen and initializes the sub-system	Then the kernel start the init process that becomes the first process(pid = 1) [/sbin/init]
It then loads SMSS (Session Manager Subsystem Service) with priority 11 and passes control to it. [%systemroot%\system32\smss.exe]	The kernel then passes the control to the init process, which starts the other processes.
SMSS initializes the pagefile and other registry hives.	Init loads the sysinit file specified in the inittab [/etc/rc.d/rc.sysinit]
Starts the 32bit windows kernel (WIN32K.SYS) [%systemroot%\system32\win32k.sys]	Sysinit mounts /proc, enables swap, starts network services, checks and mounts other file systems …

Starts CSRSS (Client Server Runtime Sub System) with priority 13. [%systemroot%\system32\csrss.exe]	Init process then reads inittab file to decide the runlevel (initdefault) and other processes to load. [/etc/inittab]
Then it starts WINLOGON with priority 13 and passes control to it. [%systemroot%\system32\winlogon.exe]	Then init reads the runlevel to boot the system and starts all the scripts according to the runlevel in the /etc/rc.d/rcX.d (X = runlevel)
WINLOGON then starts LSASS [Local Security Authorization Subsystem Service] with priority 9. [%systemroot%\system32\lsass.exe]	Then init process starts the mingetty process (one for each terminal), which opens communication paths to tty devices [/sbin/mingetty]
WINLOGON then loads MSGINA (Graphical user Identification aNd Authentication), which presents the login screen to the user. [%systemroot%\system32\msgina.dll]	It then starts /bin/login; if GUI is present, prefdm script is read and the preferred desktop manager (gdm, kdm, xdm) is loaded [/etc/prefdm]
It then loads SERVICES (Services Controller) with priority 9. [%systemroot%\system32\services.exe]	Once the user logs in, /etc/profile and ~/.profile, ~/.login, ~/.bashrc, ~/.bash_login are executed to set the user environment
Once the user logs in, SERVICES takes control and loads all the necessary 'automatic' services for that user.	

SYSPART = C: or C Drive (System Partition)
BOOTPART = Partition where Windows is loaded (Boot Partition)
%systemroot% = BOOTPART:\WINDOWS
Default Priority (Windows) = (Normal) 8 [1 – 15]

All the processes started in the windows boot process are owned by 'SYSTEM' user.

APPENDIX – B: malinfo.bat

```
dir c:\ /ah > malinfo.rtf
dir %windir% /ah >> malinfo.rtf
dir %systemroot%\system32 /ah >> malinfo.rtf
dir "%userprofile%\Start Menu\Programs\Startup" >> malinfo.rtf
dir "%userprofile%\Start Menu\Programs\Startup" /ah >> malinfo.rtf
dir "C:\Documents and Settings\All Users\Start Menu\Programs\Startup" >> malinfo.rtf
dir "C:\Documents and Settings\All Users\Start Menu\Programs\Startup"/ah>> malinfo.rtf
START wmicinit.bat
wmic /append:malinfo.rtf process
tasklist >> malinfo.rtf
netstat -nab >> malinfo.rtf
ECHO Open malinfo.rtf in wordpad
PAUSE
```

The output of the script is directed to the file "malinfo.rtf".

```
dir c:\ /ah > malinfo.rtf
dir %windir% /ah >> malinfo.rtf
dir %systemroot%\system32 /ah >> malinfo.rtf
dir %userprofile%\"Start Menu"\Programs\Startup >> malinfo.rtf
dir %userprofile%\"Start Menu"\Programs\Startup /ah >> malinfo.rtf
dir "C:\Documents and Settings\All Users\Start Menu\Programs\Startup" >> malinfo.rtf
dir "C:\Documents and Settings\All Users\Start Menu\Programs\Startup"/ah>> malinfo.rtf
```

These commands will display all hidden files in the root directory, windows directory, system directory and startup directories. They also display all files in windows startup folder. This information is redirected to a text file for later analysis.

```
START wmicinit.bat
```

This calls another shell to install wmic as it is not installed by default. Close the window once the installation is over.

WMICINIT.BAT

```
wmic
```

```
wmic /append:malinfo.rtf process
tasklist >> malinfo.rtf
```

These commands will list the executables running in the system along with the path and the process id.

```
netstat -nab >> malinfo.rtf
```

This command will save all the network connections that are present. This is useful in identifying any malware that is listening for a connection. This command also output the executables listening on the ports. [Eg: Bots or backdoors dropped by viruses or worms]

APPENDIX – C: malinfo.bat Output

The following is an output of malinfo.bat script from a system infected with Autorun.abt virus. The virus executables and processes are highlighted.

malinfo.rtf

```
Volume in drive C has no label.
Volume Serial Number is 4414-C977

 Directory of c:\

2008-03-08  23:00               144 autorun.inf
2007-12-01  23:49               232 boot.ini
2007-12-01  23:47                 0 IO.SYS
2007-12-01  23:47                 0 MSDOS.SYS
2004-08-12  06:00            47,564 NTDETECT.COM
2004-08-12  06:00           250,032 ntldr
2008-03-08  22:32       805,306,368 pagefile.sys
2007-12-01  23:56    <DIR>          RECYCLER
2008-02-13  00:53           229,621 smss.exe
2007-12-01  23:53    <DIR>          System Volume Information
               8 File(s)    805,833,961 bytes
               2 Dir(s)   1,213,009,920 bytes free


Volume in drive C has no label.
 Volume Serial Number is 4414-C977

 Directory of C:\WINDOWS

2008-03-08  23:00               144 autorun.inf
2008-03-01  04:19    <DIR>          CSC
2007-12-01  23:51    <DIR>          ie7
2008-03-07  11:29    <DIR>          inf
2008-03-07  12:01    <DIR>          Installer
2008-02-13  00:53           229,621 killer.exe
2008-02-13  00:53           229,621 smss.exe
2007-12-01  23:46               749 WindowsShell.Manifest
2004-08-12  06:00            48,680 winnt.bmp
2004-08-12  06:00            48,680 winnt256.bmp
               6 File(s)        557,495 bytes
              40 Dir(s)   1,213,009,920 bytes free


Volume in drive C has no label.
 Volume Serial Number is 4414-C977

 Directory of C:\WINDOWS\system32

2007-12-01  23:46               749 cdplayer.exe.manifest
```

```
2007-12-01  23:46               488 logonui.exe.manifest
2007-12-01  23:46               749 ncpa.cpl.manifest
2007-12-01  23:46               749 nwc.cpl.manifest
2007-12-01  23:46               749 sapi.cpl.manifest
2007-12-01  23:46               488 WindowsLogon.manifest
2007-12-01  23:46               749 wuaucpl.cpl.manifest
               7 File(s)          4,721 bytes
               0 Dir(s)   1,213,009,920 bytes free



Volume in drive C has no label.
 Volume Serial Number is 4414-C977

 Directory of C:\Documents and Settings\admin\Start Menu\Programs\Startup

2007-12-01  17:41    <DIR>          .
2007-12-01  17:41    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)   1,213,009,920 bytes free



Volume in drive C has no label.
 Volume Serial Number is 4414-C977

Directory of C:\Documents and Settings\admin\Start Menu\Programs\Startup

2007-12-01  23:47                84 desktop.ini
               1 File(s)             84 bytes
               0 Dir(s)   1,213,009,920 bytes free



Volume in drive C has no label.
 Volume Serial Number is 4414-C977

 Directory of C:\Documents and Settings\All Users\Start Menu\Programs\Startup

2007-12-01  17:41    <DIR>          .
2007-12-01  17:41    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)   1,213,009,920 bytes free



Volume in drive C has no label.
 Volume Serial Number is 4414-C977

Directory of C:\Documents and Settings\All Users\Start Menu\Programs\Startup

2007-12-01  23:47                84 desktop.ini
2008-02-13  00:53           229,621 lsass.exe
               2 File(s)        229,705 bytes
               0 Dir(s)   1,213,009,920 bytes free
```

```
Caption              ExecutablePath
System Idle Process  System Idle Process
System               System
smss.exe             C:\WINDOWS\System32\smss.exe
csrss.exe            C:\WINDOWS\system32\csrss.exe
winlogon.exe         C:\WINDOWS\system32\winlogon.exe
services.exe         C:\WINDOWS\system32\services.exe
lsass.exe            C:\WINDOWS\system32\lsass.exe
svchost.exe          C:\WINDOWS\system32\svchost.exe
svchost.exe          C:\WINDOWS\system32\svchost.exe
svchost.exe          C:\WINDOWS\system32\svchost.exe
svchost.exe          C:\WINDOWS\system32\svchost.exe
svchost.exe          C:\WINDOWS\system32\svchost.exe
spoolsv.exe          C:\WINDOWS\system32\spoolsv.exe
alg.exe              C:\WINDOWS\System32\alg.exe
explorer.exe         C:\WINDOWS\Explorer.EXE
killer.exe           C:\WINDOWS\killer.exe
ctfmon.exe           C:\WINDOWS\system32\ctfmon.exe
smss.exe             C:\WINDOWS\smss.exe
lsass.exe            C:\Documents and Settings\All Users\Start
                            Menu\Programs\Startup\lsass.exe
rundll32.exe         C:\WINDOWS\system32\rundll32.exe
cmd.exe              C:\WINDOWS\system32\cmd.exe
cmd.exe              C:\WINDOWS\system32\cmd.exe
wmic.exe             C:\WINDOWS\System32\Wbem\wmic.exe
wmic.exe             C:\WINDOWS\System32\Wbem\wmic.exe
wmiprvse.exe         C:\WINDOWS\system32\wbem\wmiprvse.exe
```

```
Image Name                     PID Session Name     Session#    Mem Usage
========================= ====== ================ ======== ============
System Idle Process              0 Console                 0         28 K
System                           4 Console                 0        236 K
smss.exe                       588 Console                 0        388 K
csrss.exe                      648 Console                 0      4,392 K
winlogon.exe                   672 Console                 0      2,584 K
services.exe                   716 Console                 0      3,388 K
lsass.exe                      728 Console                 0      2,400 K
svchost.exe                    884 Console                 0      5,028 K
svchost.exe                    968 Console                 0      4,112 K
svchost.exe                   1064 Console                 0     29,160 K
svchost.exe                   1244 Console                 0      2,944 K
svchost.exe                   1344 Console                 0      4,368 K
spoolsv.exe                   1488 Console                 0      4,896 K
alg.exe                        272 Console                 0      3,532 K
explorer.exe                  1960 Console                 0     26,644 K
killer.exe                    1508 Console                 0      4,260 K
ctfmon.exe                    1984 Console                 0      3,372 K
smss.exe                       416 Console                 0      4,280 K
lsass.exe                      424 Console                 0      4,148 K
rundll32.exe                  1056 Console                 0     13,056 K
cmd.exe                       2012 Console                 0      1,796 K
cmd.exe                       1860 Console                 0      2,640 K
wmic.exe                      1428 Console                 0      4,884 K
wmiprvse.exe                   944 Console                 0      5,900 K
tasklist.exe                  1380 Console                 0      4,436 K
```