

# The PE Win32 File Infectors

Name: Sriram Parameshwaran

January 12, 2009

# Contents

<b>SNo</b>	<b>Title</b>	<b>Page No</b>
1.0	Abstract	3
1.1	Introduction	3
2.0	The Current Scenario of viruses	3
3.0	PE executable files of windows	4
4.0	Infection Strategy of viruses	4
4.1	Replication Classification of Viruses	5
4.1.1	Companion Viruses	5
4.1.2	Overwriting Viruses	6
4.1.3	Parasitic Viruses	6
4.1.3.1	Parasitic Prepending Viruses	6
4.1.3.2	Parasitic Cavity Viruses	6
4.1.3.3	Parasitic Appending Viruses	6
4.1.3.4	Infection Through TLS	7
4.1.4	Compiler Way	7
4.2	Encryption Viruses	7
4.3	Infection API's	8
5.0	Conclusion and Future Work	8
6.0	Annotations	9
7.0	Bibliography	9

## The PE Win32 File Infectors

### **1.0 Abstract**

File Infector type viruses modify the code in any frequently used legitimate files in such a way that they can be up before your original application's process starts. Malwares can achieve this by modifying or inserting the malicious code into any legitimate<sup>1</sup> files in such a way that it can be persistent to the system even after shutdown. This information is of importance for a forensic analyst to assess the ways havoc has been caused in case of a malware infection. This paper will discuss these topics starting from Introduction to PE Files, Different types of PE infectors, classification on the basis of infection, their behavior, with examples.

My work experience of above 3 years as senior threat research analyst with "Comodo antivirus"<sup>2</sup> enhanced me with lots of ideas, concepts, and kept me up to date with the latest malware and their the infection methodologies. I came up with this topic to share my experiences of my research on this topic since has been an area of limited research since it needs a lot of understanding of both assembly language and windows internal architecture.

### **1.1 Introduction**

What is a virus? "Infectious agent that replicates itself only within cells of living hosts" is the definition give to the viruses infecting a human body. This can similarly be extended to the computer viruses where the term "Infectious agent" refers to a computer program.

This term, "computer virus" is mostly used as a catch-all phrase to include all types of malware. Malwares are broadly classified into Viruses, Worms, and Trojans according to their infection /

propagation technique. All these types have a payload but the way they propagate differs. A worm can exploit security vulnerabilities in the OS or software to spread itself onto other computers through a network or other interactive media, while a Trojan horse is a program that appears harmless when it comes to the user but has a hidden payload. They come in as a beautifully crafted application or greeting message, like an e-card, bills, offers, etc. A virus propagates in a technically different way. They hide themselves inside legitimate files a person uses and creates in his day to day life, like web pages, Microsoft Word documents, Power point slides, executable files, etc. Viruses insert their code inside these types of files and use them as a carrier to propagate from one PC to another.

Malwares are written in many available languages that supports code execution. The reason for the files like MSWord, Power Point, Web Pages, and executables are chosen for malware writing is because of their wide usage and their support to execute HLL scripts. Windows PE executables are very vulnerable. Consider a situation where your executable gets infected and, you never get your data back? Your data is lost, might be permanent. Sometimes this can be of good evidence to forensic experts in analyzing the degree and the ways they have caused the damaged.

### **2.0 The current scenario of viruses**

The Creeper virus was first detected on ARPANET<sup>3</sup>, in the early 1970s. A program called "Rother J" was the first computer virus to appear "in the wild"<sup>4</sup> written in 1981 by Richard Skrenta. The first PC virus in the wild was a boot sector virus dubbed Brain, created in 1986

by the Farooq Alvi Brothers in Lahore, Pakistan. The file infector virus that made its mark of mass destruction was the Chernobyl aka CIH<sup>5</sup>, which overwrites critical information on infected system drives, and finally corrupting the system BIOS. It was created as a tribute to the Chernobyl, Ukraine<sup>6</sup> incident which took place on April 26, 1986. The next highly seen infection routine was the “Virus.Win32.Virut”<sup>7</sup> family, which is followed by “Virus.Win32.Sality”<sup>8</sup> and “Virus.Win32.Expiro” variants. The recently added to the infector list is the “Virus.Win32.Tenga”<sup>9</sup> variants.

### 3.0 PE Executable<sup>10</sup> files of Windows

EXE is the COFF executable file extension we see in windows. The other similar extensions that share the same PE COFF are SCR, CPL, DLL, VXD, OCX and SYS. The EXE and SCR files are loaded by windows using the PE loader program which loads the file contents into the memory but the extensions DLL, SYS, OCX and VXD are loaded when required by an executable while CPL files are control panel management files.

The windows PE COFF format is a collection of data structures which encapsulates all the necessary information for the windows OS to load and manage that wrapped executable code. All PE files start with the DOS header called IMAGE\_DOS\_HEADER which occupies the first 64 bytes of a file. Sometimes, in cases where a windows binary is made to run in a non windows subsystem like DOS, the user must be indicated that this file execution is incompatible in DOS, In this condition, the space next to IMAGE\_DOS\_HEADER contains code to display that warning message “This program cannot be run in DOS Mode”. The last 4 bytes in the dos header points to

the starting address of the PE header structure.

PE header starts with the PE header signature DWORD 50 45 00 00. This is the start of the IMAGE\_NT\_HEADER which consists of two more structure elements namely, IMAGE\_FILE\_HEADER, and IMAGE\_OPTIONAL\_HEADER32. The IMAGE\_FILE\_HEADER is 20 bytes long and contains the information about the physical layout and the properties of the file like Number of Sections, Machine code executable, characteristics of the file. The next 224 bytes is the IMAGE\_OPTIONAL\_HEADER32 which holds the main information of the file like AddressofEntryPoint, SectionAlignment, ImageBase, FileAlignment, SizeofImage, DataDirectory. Datadirectory is an array of sixteen IMAGE\_DATA\_DIRECTORY structures which stores address to different predefined items of which export, import, TLS, are of importance, and is illustrated later in the document. Following is the structure IMAGE\_SECTION\_HEADER which contains the section’s starting offset and size in both file and memory and the characteristics.

### 4.0 Infection Strategy of viruses

For a virus to spread / replicate itself, it must be either loaded into the memory or it must be loaded in a place where it is permitted or have enough privileges to access resources to infect. This is the main reason why they attach themselves into the legitimate executable files, so that when the infected file is executed they load into the memory with enough privileges and they can start their infection. A virus program has two main phases in their infection strategy. They are the Infection or the Replication module and the Payload module. Replication module is responsible in finding non-

infected files and infecting them while payload has the target objective of destruction. The replication can occur in slow or fast pace. In the fast replication module, they can infect more files and can use more system resources thus alerting the user of suspicious background activity. The corollary is in a slow replication module where they replicate slowly infecting less number of files also using less system resource thus making stealth background activity.

#### 4.1 Replication Classification of Viruses

With accordance to the replication type, the file infector viruses can be classified<sup>11</sup> into Parasitic, Overwriting, Companion, Entry Point Obfuscating viruses. Some programs which doesn't come under such classification are boot sector viruses, which modify the 512 byte boot sectors<sup>12</sup> of hard disks and floppy disks. These are of less prevalence now because modern BIOSes are equipped with read only Boot Sector security feature.

Infections are caused according to their design. These can be illustrated with simple illustrations in two phases, the original file and post infection phase when they are attacked by the different types of file infectors. This classification is based on location where the infection code is placed inside the target executable. Consider an executable file with sections .text, .idata, .rsrc, as illustrated in the figure below.

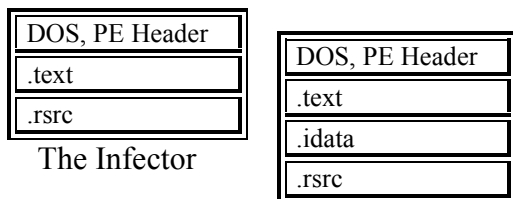


Fig 1. Example structure of a PE File and infector file

#### 4.1.1. Companion Viruses

**Companion viruses** are viruses which replace the original file with the infection executable and take a backup of the original file for execution of the original part. When the application is launched, the infection executable gets executed and this in turn, calls the original executable. The legitimate file is not aware that it is infected. This is illustrated in figure 2.

Another type of the companion viruses keeps the original executable in their overlay<sup>13</sup> part or in their resource part. When the file gets called, they create the original executable from the overlay as a new file in the same folder and execute it. This can be illustrated above.

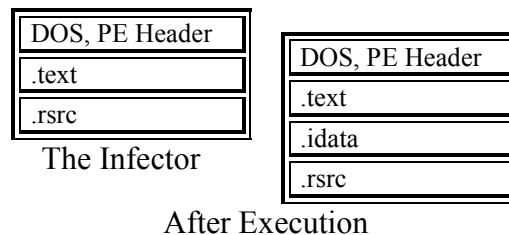
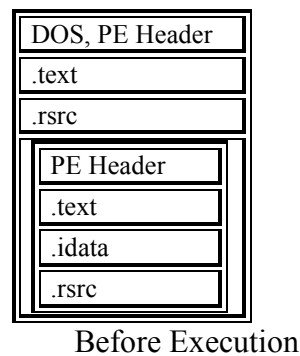
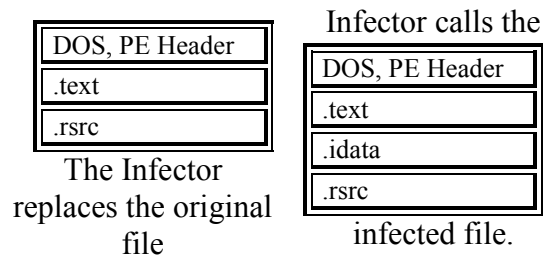


Fig 2: Infection of companion virus

The best and the latest type of infection which was seen recently had the infection routine and the payload as two different files. The infection routine finds infectable files, opens them, and places the payload into the executable in such a way that the payload is executed first every time.

#### 4.1.2 Overwriting Viruses

**Overwriting Viruses** replace the original executable code completely with the infector code. This type of infection, the original executable is completely deleted. It is very hard to get back the original file. This can be illustrated below:

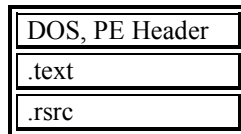


Fig 3: Overwriting Viruses after infection

#### 4.1.3 Parasitic Viruses

There are some types of viruses that parse your legitimate executable and insert their code into places that are unused or make themselves room and place their code. These types of viruses are called Parasitic Viruses. They are classified into 3 types according to their pattern of infection. They are prepending viruses, appending viruses and Insertion or cavity viruses.

##### 4.1.3.1 Parasitic Prepending virus

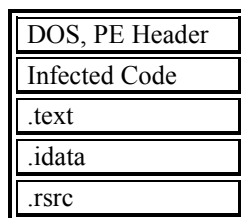


Fig 4: Prepending Viruses after infection

**Prepending viruses** insert the code into the executable by pushing the existing sections below. They may change the entry point and make the infection routine to execute first and then executes the

original code. In this type, the file size increases according to the infection size. This can be illustrated as in figure 4

##### 4.1.3.2 Parasitic cavity Virus

**Insertion or cavity viruses** insert their code between the spaces available in the executable. They search the header for the executable section and insert their code into it. If they don't find enough space to insert their code, they go to the next section. If they don't find any executable section, they make the section characteristics to executable and insert their code.

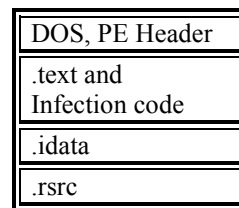


Fig 5: Cavity Viruses after infection

Free spaces between two consecutive sections are caused mainly due to the values of filealignment and the sectionalignment in the PE header structure. If "filealignment" is high, the larger is the gap between the sections. This space is more than enough to place the malware code. Cavity type file-infectors make use of this space. The main advantage of this technique is that the file size does not increase i.e. physical modification of the file is zero and hence the user is less alerted. Disadvantages are that if the space available to infect is less, then the file cannot be infected. It is necessary that the infecting section has to be an executable section.

##### 4.1.3.3 Parasitic Appending Virus

**Appending viruses** are viruses that insert the code into the end of the infected file. The infected file grows according to the amount of code inserted. The code is inserted by creating a new

section at the end of the file. They may modify the value of EP to execute the infection code first, after the infection is done, they jump to the original code and continue the original execution. This can be illustrated as.

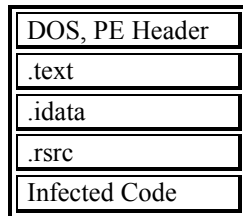


Fig 6: Appending Viruses after infection

#### 4.1.3.4 Infection Through TLS

**TLS** is one of the low explored and highly exploitable sections in a PE file. TLS is less documented section of a PE file. TLS is a special callback function implemented<sup>14</sup> to execute a piece of code when an executable is loaded, even before the WinMain() function. A malware writes its code in this part of the executable. The advantage of this infection is that they can execute as a separate thread. They aren't visible when loaded in a debugger. This place also has the technique to check if the file is being started and debugged by a debugger, so that a malware can zero out the malware content or do any other mischievous activity. However the code has to be inserted in free space in an executable.

#### 4.1.4 Compiler Way

One of the most recent and innovative ways of file infection is by infecting the developer library files found in compilers and linkers. They place the payload in such a place that when linker is called to create the new executable, the infection code is automatically inserted into the executable and the infection is thus carried into the new executable. The variant, "Virus.Win32.Induc", which

infects Pascal libraries is an example of such virus

## 4.2 Encryption Viruses

All the above technique infect different executable in a similar way with a regular pattern. These types of infection can be scanned, identified with a pattern based signature and are simple to clean by an AV engine. The original data can be restored by deleting the infection part section and restoring the Original Entry Point if necessary.

Some malwares try to escape from such signature based detection pattern done by the AV engines by modifying their code using some encryption algorithm. They normally make use of a decryption key to decrypt the code. These kinds of viruses are called encryption Viruses. In some cases, the infection is not common. They change the decryption key from file to file. Some even have implementation of different algorithm to generate the decryption key, called as decryption routine. The collection of such polymorphic code, polymorphic key generation, infection routine and payload are collectively called as polymorphic engine or mutation engine. These kinds of viruses are called polymorphic viruses. "Virus.Win32.Magic.1590" can be said as the best example for polymorphic viruses. Some more advanced encryption viruses don't have such a common decryption key or a decryption routine. Some get the name of the computer using "GetComputerNameA" API and uses it as the key to encrypt. Some variants of these viruses implement brute-force algorithm to find their encryption key in order to decrypt their content. These kinds of infections are difficult to decrypt and clean. They are hence called metamorphic viruses.

Some malwares do not want them to be executed in virtual OS environments like VMWare or VirtualPC, or their code to be debugged. The malware writers insert a routine for anti debugging techniques and routines that detects the VM environment so that they are not easily identified by honeypots, honeynets or automatic systems or malware analysts. Themida<sup>15</sup>, commercially available software, enables malware writers to prevent their code from being exploited or being detected.

### 4.3 Infection API's

As a first step, malware infection routine needs to find the file to infect, then it needs to check if the file is suitable for infection, if yes, inserts the infection routine and the payload. Malware writers cannot write their own code for implementing all these tasks because they cause their code to grow. So they make use of the pre defined OS libraries called API to serve their purpose. If a file infector's replication module is analyzed, we can find that they always start with API "FindFirstFileA" followed by "FindNextFileA". These are the only API's found for this purpose. File operations use "CreateFileA", "ZwCreateFileA" or "OpenFile" or "\_lopen" functions, which returns the handle to the file. Then it makes use of

"MapViewOfFile", "SetFilePointer", "ReadFile", and "WriteFile" to check if the file can be infected. This includes the check if it has been already infected. If the file is good to infect, it infects the target file according to its design and then closes the handle with "CloseHandle". It also keeps mutex names using "CreateMutexA", "OpenMutexA" as a flag which says that the infector program is already loaded into the memory to avoid multiple infection processes. These are the API's that an operating system has, in order to the logic these file infectors possess.

### 5.0 Conclusion and future work

I hope this report would have given a brief idea of what a file infector is and how a file infector infects other files. This report also briefed the different behavior of different types of viruses and the ways they can be handled in case of analysis by a forensic expert or an automated system or an AV engine. In case of a forensic expert, one could use this classification to categorize, analyze, and define the evidence and to calculate the damage done by the file. In case of an analyst, one can categorize the file infector accordingly since there is no defined classification for viruses, and he can write dis-infection routines for the analyzed file infectors.

## 6.0 Annotations

- PE – Portable Executable
- COFF – Common Object File Format
- DOS – Disk Operating System
- AV – Anti Virus
- HLL – High Level Language (Visual Basic)

---

## 7.0 Bibliography

<sup>1</sup> Dictionary: Safe file, Clean File

<sup>2</sup> Comodo Antivirus more details available at [www.comodo.com](http://www.comodo.com) [Jan 01 2010]

<sup>3</sup> Further info on introduction to ARPANET Available at: <http://en.wikipedia.org/wiki/ARPANET> [Jan 01, 2010]

<sup>4</sup> More info on current spread of malware and its pattern available at: [www.wildlist.org](http://www.wildlist.org) [Jan 01 2010]

<sup>5</sup> CIH Virus infection technique details available at: [http://en.wikipedia.org/wiki/Chernobyl\\_Virus](http://en.wikipedia.org/wiki/Chernobyl_Virus) [January 01, 2010]

<sup>6</sup> The Chernobyl Disaster more info available at: [http://en.wikipedia.org/wiki/Chernobyl\\_accident](http://en.wikipedia.org/wiki/Chernobyl_accident) [January 01,2010]

<sup>7</sup> More information on the infection technique and disinfection available at: [http://www.f-secure.com/v-descs/virus\\_w32\\_virut.shtml](http://www.f-secure.com/v-descs/virus_w32_virut.shtml) [Jan 12 2010]

<sup>8</sup> More information on the infection technique and disinfection available at: [http://www.f-secure.com/v-descs/virus\\_w32\\_sality\\_aa.shtml](http://www.f-secure.com/v-descs/virus_w32_sality_aa.shtml) [Jan 12 2010]

<sup>9</sup> More information on the infecting technique and vulnerabilities exploited and its payload available at: <http://www.viruslist.com/en/viruses/encyclopedia?virusid=88153> [Jan 12 2010]

<sup>10</sup> Win32 PE COFF and structures defined by Microsoft Available at: <http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx> [January 03, 2010]

<sup>11</sup> Classification based on kaspersky's detection methodology available at: <http://www.viruslist.com/en/virusesdescribed?chapter=152540474> [January 02, 2009]

<sup>12</sup> Boot sector viruses Available at: [http://en.wikipedia.org/wiki/Boot\\_sector\\_virus](http://en.wikipedia.org/wiki/Boot_sector_virus) [January, 02, 2010]

<sup>13</sup> 2.6 Overlay explained Available at: [http://www.sunbelt-software.com/ihs/alex/vb07\\_paper.pdf](http://www.sunbelt-software.com/ihs/alex/vb07_paper.pdf) [Jan 05 2010]

<sup>14</sup> Implementing a TLS manually available at: <http://www.cyberarmy.net/library/article/1653> [Jan 9, 2010]

<sup>15</sup> Themida packer information available at: <http://www.oreans.com/themida.php> [Jan 12 2010]