WIN32 Executable Viruses Detection

Dr. Ayad A. AbdulSalam
College of Science for Women, University of Baghdad.

Falah Mahdi Abduallh
College of Islamic Science, University of Baghdad

Abstract

Computer viruses has been grown rapidly in last few years, corrupted huge number of applications and devices. In the other side, antivirus programming facing the spreading of all types of viruses. This work dedicated for detecting viruses that infect executable files namely (Win 32), because Win32 files like .EXE, .PE widely used in PC’s and networks. This work explains how viruses work, the behavior of viruses, and how infect different types of files, Illustrates many techniques of virus detection, our work is for detecting some types of infected file by exploiting the change in checksum, which occurs when the data of files has changed. Some measurements were using to evaluate our work.

Keywords
Executable virus, Win32 virus, Malware, Worms, Trojan horse, Logical bomb, Heuristic Detection, Behavior Detection, Change Detection, checksum, Portable executable.

الخلاصة:-

ادى تزايد ظهور فايروسات الحواسيب بشكل كبير في السنوات الاخيرة الى تدمير اعداد كبيرة من التطبيقات والبرامج والاجهزة. مما حدى بالعاملين على صناعة البرامج المضادة لفايروسات الحواسيب لمواجهة الانتشار السريع وايجاد برامج لمعالجة تاثير شتى انواع الفايروسات. خصصنا هذه الورقة البحثية للعمل على اكتشاف وتحديد الفايروسات التي تصيب الملفات من نوع (Win 32) كون هذه الانواع من الملفات خاصة ,EXE PE تستخدم بشكل واسع النطاق في الحواسيب الشخصية وتطبيقات الشبكة الدولية. توضح هذه الورقة كيفية عمل الفايروسات، وطريقة تصرفها، واسلوب اصابة الملفات المختلفة، استعرضنا بعض تقانات الكشف عن الفايروسات، تم استثمار حقيقة التغير في دالة (checksum) للكشف عن الفايروسات والذي يحدث عند تغيير اي جزء من المعلومات او البيانات الموجودة في الملف بسبب الاصابة باحد انواع الفايروسات. استخدمنا عدة مقاييس لتقييم النتائج المستخلصة من تجارب عديدة وكانت النتائج مشجعة جدا.

1. Introduction:

Computer viruses are those annoying and what are many times destructive programs that can reproduce itself by attaching its code to another program [1]. A computer virus must piggyback on top of some other program or document in order to get executed. Once it is running, it is then able to infect other programs or documents [2].

2. Types of Viruses:

We will first describe computer viruses and then position them in the more realistic and current context of computer infection programs (also referred to as malware). The notion of computer infection program corresponds to either simple program installation in case of simple malware or to code replication in case of self–reproducing program. Usually, all these programs spread and operate according to the following way:

1. The infecting program (the malware) is carried by a host program (called an infected program, the term of “dropper” is used to define the very first infection launched).

2. Whenever the dropper is executed:

a) The infection program takes control and acts according to its own operation mode. The host program is temporarily dormant.

b) Then the infecting program returns control to the host program. The latter is executed normally without betraying the presence of the infecting program.

The main feature which distinguishes self-replicating programs from simple infections is code replication. Whenever a program makes an exact copy its own code even only once, we have a true viral replication mechanism: at least two copies of the code are present on the machine at the same time. Such a phenomenon does not occur in the case of simple computer infection program.

2.1: Logical Bombs:

A logical bombs is a non self – reproducing malware, which installs itself into the system and waits for some trigger incident or event (some data which is present or absent in the system, action, a specific system data) before performing a damaging or an offensive function (trigger mechanism).

2.2: Trojan Horses:

Trojan horse is simply programs that feign, by their name or their documentation, to do one thing, when in fact they do something else entirely, something often very destructive [2]. Trojans’ spreading potential is not very big, because once they are run they give themselves away (cease to be Trojans), and the only way for a Trojan to propagate itself would be for a user to copy it to somewhere else.

2.3: Viruses:

Is a specific type of Trojan horse that can be used to spread its infection from one computer to another , in the days when a virus was still an interesting novelty perhaps because it appeared before many viruses had been discovered, his definition of a virus extends only to viruses that propagate by attaching themselves directly to other programs.

2.4: Worms:

A worm can be defined as a program propagating itself in a network of computers, using bugs, which are unforeseen (by the designers and users) side effects of the operating system, or breaking (guessing) passwords to gain access to other machines in the network. Contrary to viruses, no user interactions are needed for the worm to spread. Worms need no host program to propagate, viruses are parasitic, and worms are not.

3. How is Virus Work?

Computer viruses are usually written in assembler or developed with one of the many virus creation toolkits. Some virus creation toolkits now come with help files and contact information on channels [4], after a virus is created it must be put into a program with something called a dropper. After the virus is properly packaged then it can be distributed. Viruses are often distributed through e-mails, bulletin boards, or free programs. Once a virus gets into a system, it does not do anything until the infected program is executed. When the virus starts executing, it usually infects other files before executing the malicious part of its code, which is known as the payload [5]. In many instances the payload does not do anything for quite a while, in order to ensure widespread infection of a system before it fully exposes itself.

3.1: Viruses Life Cycle:

The phase in the virus is conceived and tested; three main stages in their “life” can be distinguished. Their life can be more or less long, depending on the type of the virus or the desired effect. First stage is the **infection phase:** During this stage, the virus will spread throughout the target environment. Then the **incubation phase:** This phase represents the longest one in the life of a virus. It is worth mentioning the examples of spy viruses which are an exception to the rule insofar as they keep their stay in this infected environment down to a minimum and disinfect themselves once their offensive action has been completed. The main purpose here is the virus’s survival in the infected system. Accordingly, it must escape detection by either the user himself, or antivirus programs. Last phase is the **disease phase;** the way it is triggered depends on various factors and especially on the location where the offensive routine was inserted in the code.

4. Virus Classification:

Viruses are usually classed according to the **target formats**: executable viruses or document viruses, or according to **target component** or device: for example viruses which take over either boot sectors (boot viruses) or device drivers. Or according to the **programming language**: assembly viruses, code source viruses, interpreted language viruses (or script viruses). Or according to the **behavior** of the virus: armored viruses, slow or rapid viruses, retroviruses, resident viruses, polymorphic viruses, stealth viruses, etc. Or according to the **nature of the final payload**: spy viruses, corrupting viruses, deletion viruses, destruction viruses, etc. Or according to the **way they operate**: combined viruses, psychological viruses (hoaxes, jokes) [6].

5. How we can detect viruses:

There are basically three ways to detect viruses: detection by appearance, detection by behavior, and detection by change.

5.1: Detection by Appearance:

It is purely passive defense method; it can only detect an infection after the infection has already taken place. It is never the less the most widely used today, not least because of its ease to apply. Detection by appearance involves looking in executable programs for what might be suspicious code. The ideal detection by appearance program would be able to sniff out all virus code, while leaving uninfected programs alone. One of detection by appearance methods is **Heuristic Detection**, it looks for characteristics viruses often have. It uses an AI-search (Artificial Intelligence search).

5.2: Detection by Behavior:

For a virus to propagate, it needs to perform a number of specific actions, such as reading directory entries, opening files, writing to files, etc. most of these actions are performed all the time by the system itself and other programs alike, although a few would not be performed under ordinary conditions. Detection by behavior entails monitoring the system, to watch out for such dubious action.

5.3: Detection by Change:

All viruses cause some change to the system in which they infect. The change is always very easy to spot with normal DOS tools: increase in file length, decrease in memory, change of time /date stamp, etc. For detect any change, it is of course to detect the initial state, file length, respectively memory-size before infection, to compare with new state of system.

6. The Win32 API and Platforms That Support It:

In 1995, Windows 95 was introduced by Microsoft as a new major operating system platform. The Windows 95 system is strongly based on Windows 3.x and DOS technologies, but it gives real meaning to the term Win32.

What is Win32? Originally, some programmers did not even understand the difference between Win32 and Windows NT. Win32 is the name of an API (Application Programming Interface) no more, no less. The set of system functions available to be called from a 32-bit Windows application is contained in the Win32 API. The Win32 API is implemented on several platforms, one of them being

Windows NT, the most important Win32 platform. Besides DOS programs, Windows NT also is capable of executing 16-bit Windows programs. In addition, Windows NT introduced the new Portable Executable (PE) file format that can run Win32 applications (which call functions in the Win32 API set). As the word portable indicates, this format is supposed to be an easily portable file format, which is actually the most common and important one to run on Windows NT.

Until Windows NT gained more momentum, Windows 9x was Microsoft's Win32 platform. After Windows NT, Windows 2000 and Windows 98/me gained popularity and were replaced by Windows 2003 and the more secure Windows XP, and Windows Vista server editions, which support the .NET extension by default. All of these systems supported a form of Win32 API that, in most cases, provides binary compatibility among all of these systems.

Unfortunately, over the period from 1995 to 2004, virus writers utilized these platforms aggressively, resulting in the appearance of more than 16,000 variants of 32-bit Windows viruses. However, the principles of these viruses have not changed much. [8].

6.1 Infection Techniques on 32-Bit Windows:

A 32-bit Windows virus can infect different kinds of executable programs used by Windows 95, NT, XP, and Vista. Because the most common file format is the PE format, the PE format makes it possible for viruses to jump easily from one 32-bit Windows platform to another. We shall concentrate on Infection techniques that attack this particular format because these viruses have a strong chance of remaining relevant in the future.

6.2 Portable Executable File Format:

To understand how Win32 viruses work, you need to understand the PE format, we will provide a tour of the PE file format that Microsoft designed for use on all its Win32 operating systems (Windows NT, Windows 95, and Windows XP Win32s).

For Win32, all the memory used by the module for code, data, resources, import tables, and export tables in one continuous range of linear address space. The only thing that an application knows is the address where the loader mapped the executable file into memory. When the base address is known, the various pieces of the module can easily be found by following pointers stored as part of the image [8].

Another idea we should become familiar with is the Relative Virtual Address (RVA). Many fields in PE files are specified in terms of RVAs. An RVA is simply the offset of an item to where the file is mapped.

Another concept to be familiar with when investigating PE files and the viruses that infect them is the section. A section in a PE file is roughly equivalent to a segment in a 16-bit NE file. Sections contain either code or data (and occasionally a mixture of both). Some sections contain code or data declared by the actual application, whereas other data sections contain important information for the operating system, Figure (1) shows the overall structure of a PE file.

Offset (hex)	Meaning
00-01	0x4d, 0x5a. This is the "magic number" of an EXE file. The first byte of the file is 0x4d and the second is 0x5a.
02-03	The number of bytes in the last block of the program that are actually used. If this value is zero, that means the entire last block is used.
04-05	Number of blocks in the file that are part of the EXE file. If [02-03] is non-zero, only that much of the last block is used.
06-07	Number of relocation entries stored after the header. May be zero.
08-09	Number of paragraphs in the header. The program's data begins just after the header, and this field can be used to calculate the appropriate file offset. The header includes the relocation entries.
0A-0B	Number of paragraphs of additional memory that the program will need. The program can't be loaded if there isn't at least this much memory available to it.
0C-0D	Maximum number of paragraphs of additional memory. Normally, the OS reserves *all* the remaining conventional memory for your program, but you can limit it with this field.
0E-0F	Relative value of the stack segment. This value is added to the segment the program was loaded at, and the result is used to initialize the SS register.
10-11	Initial value of the SP register.
12-13	Word checksum. If set properly, the 16-bit sum of all words in the file should be zero.
14-15	Initial value of the IP register.
16-17	Initial value of the CS register, relative to the segment the program was loaded at.
18-19	Offset of the first relocation item in the file.
1A-1B	Overlay number. Normally zero, meaning that it's the main program.

Figure (1) a high-level view of the PE file image

7. Verifying a Checksum within Executable File:

A checksum is a value calculated from a file's content by a special algorithm. This algorithm is designed in very sensitive way, so that modifying even a single bit in the file data will cause a different checksum value. However, most algorithms permit the skipping of certain file fragments from the calculation.

Another feature of the checksums is that their length does not depend on the size of a source file. It does not matter what the size of a source file is, 1 kilobyte or 100 megabytes; the checksum length always has a fixed size. The exact checksum length depends on the algorithm and varies from 32 to 512 bits.

Usually checksums are applied to detect changes or errors in data. For example, the checksums are compared when you copy, archive, or download files. Besides assuring data integrity, this could also protect your application from hackers. Attempts to crack an application generally modify its executable.

8. Performance:

To measure the performance of proposed method program we take in our consideration the accuracy, and time consuming, we work to find high accuracy and short detecting time.

8.1 Time consuming:

By using the time function, calculate the real time of running program, in other words the time to detect virus. By execute ten experiments, that obvious from Table (1) the increasing of block size increase consuming time in semi linear manner. Figure (2) shows the semi linearity relationship between total size and consuming time.

Table (1) Relation between block size and consuming time

Experiment	Block size (KB)	Time (ms)	Experiment	Block size (KB)	Time (ms)
1	3881.211	0.343997	6	11573.066	0.989301
2	4612.330	0.392290	7	13544.764	1.133500
3	5409.026	0.505311	8	14232.552	1.217759
4	7255.397	0.640285	9	16626.706	1.424343
5	9441.087	0.88293	10	17008.229	1.590116

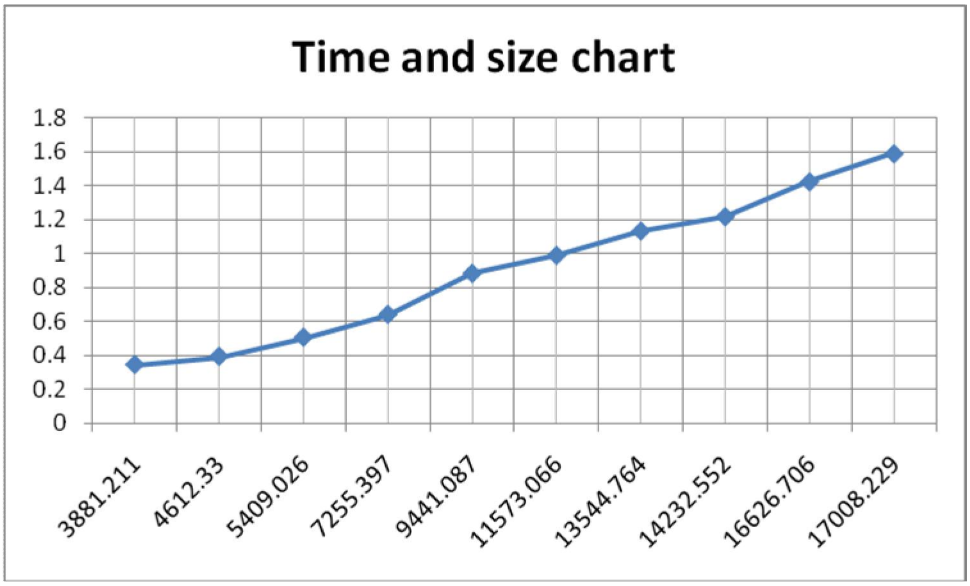

Figure (2) Time Vs. size chart

8.2 Accuracy:

We make change in the checksum of the executable files; begin with large change, then reducing gradually until found little change the program can detect. Table (2) illustrates the small difference in consuming time when the large change in the checksum, and vice versa, that means the detector needs more time when increasing sensitivity (small change).

Table (2) Time consuming Vs. Checksum changes

Experiment	Block size (KB)	Time consuming (ms) when change the no. of bytes					
		5 bytes	4 bytes	3 bytes	2 bytes	1 byte	0 byte
1	3881.211	0.378396	0.385276	0.399036	0.426556	0.481595	0.343997
2	4612.330	0.431519	0.439364	0.455056	0.486439	0.549206	0.392290
3	5409.026	0.555842	0.565948	0.586160	0.626585	0.707435	0.505311
4	7255.397	0.704313	0.717119	0.742730	0.793953	0.896399	0.640285
5	9441.087	0.971223	0.988881	1.024198	1.094833	1.236102	0.88293
6	11573.066	1.088231	1.108017	1.147589	1.226733	1.385021	0.989301
7	13544.764	1.24685	1.26952	1.31486	1.40554	1.5869	1.133500
8	14232.552	1.339534	1.363890	1.412600	1.510021	1.704862	1.217759
9	16626.706	1.566777	1.595264	1.652237	1.766185	1.994080	1.424343
10	17008.229	1.749127	1.780929	1.844534	1.971743	2.226162	1.590116

REFRENCES

[1] Tanenbaum, “Modern Operating Systems.” New Jersey: Prentice Hall. A. S. (2001).

[2] Brain, “How Computer Viruses Work”, Marshall Brain’s How Stuff Works , M. (2001).

[3] Kephart, J. O., Sorkin, G. B., Chess, D. M., & White, “S. R. Fighting Computer Viruses”, Scientific American. Viewed on May 26, 2008 at WWW: www.sciam.com/1197issue/1197kephart.html

[4] Gold, “New Virus Creation Utility Set To Wreak Havoc”, Newsbytes, S. (2001). Viewed on May 25, 2008 at

WWW: http://www.newsbytes.com/news/01/163077.html

[5] Symantec,” Reference Area “, Viewed on May 26,2008 at

WW: http://www.symantec.com/avcenter/refa.html

[6] Eric Filiol, “Computer viruses: from theory to applications”, Sprenger, 2005.

[7] Peter Szor, "Attacks on Win32," Virus Bulletin Conference, 1998.

[8] Peter Szor, “the art of computer virus research and defense”, additional Wesley professional, February 03, 2005.