Viruses using .NET Framework

Zsombor Zsolt Kurdi

John von Neumann Faculty of Informatics, Budapest Tech
Bécsi út 96/B, H-1034 Budapest, Hungary
kurdi.zsombor@nik.bmf.hu

Abstract: In our age, programming languages and development tools advance to make programmers' work much easier. The tools can assemble programs written different programming languages, they can build executables can be run independently of the operating system. This simplification of software development causes the simplification of writing programs with the prepense of malice (e.g. viruses). So developers of these tools have to care about the security. Farther this virtual equivalency of computers – and the spreading of Internet – make easier to viruses infecting a computer.

Keywords: .NET Framework, MSIL, Virus, Malware

1 Introduction

Viruses are small softwares attach themselves to other files (infection) and cause damage to the computer (this is their main goal) when a user executes the infected file. The concrete steps of the infection procedure depend on a lot of properties of the local environment (e.g. operating system, 'virus strain', file's type would be infected etc.).

Penetration of .NET Framework (and other development tools for writing platform-independent programs) causes a great simplification in virus development too. For example the viruses have not care about the type of operating system during the infection – and damaging – which is a great advantage for them.

This article is organized as follows: Section 2 gives an outline about malicious softwares (short description, history, types etc.), Section 3 treats of the .NET framework, it's usability and its similarity to Java Virtual Machine. The main section (Section 4) covers an introduction and analysis of viruses based on the services of the .NET framework, which is followed by the conclusions (Section 5).

2 Viruses and Other Malicious Softwares

A computer virus (or simply virus) is a self-replicating computer program that spreads by inserting copies of itself into other executable code or documents (definition from *wikipedia.org*). These small pieces of programs were named viruses because they behave very similar to biological viruses. The attaching procedure, when a virus copies itself to an executable is called ‘infection’, and the infected executable is the ‘host’.

The first computer virus was written in the early 1970s or even in the end of 1960s. But the term ‘computer virus’ was formally defined by Fred Cohen in 1983. Actually the number of malicious programs started increasing in this period. The most self-replicating programs were written with scientific view. Since that time this growing has accelerated and the goals of writing such a program has been changed ([1]).

Viruses are not the only malefic programs in the info-space, but they are the most well-known ones. There are a lot of other malicious softwares (malwares). The common property of these programs is to infiltrate or damage a computer system, without the owner’s consent. A malware can be

- a virus,
- a worm,
- a trojan horse,
- a spyware,
- an adware,
- any other program written with the idea of causing damage.

A worm is a special type of virus which travels over network (e.g. Internet) to find and infect files on other computers. Only viruses and worms are self-replicating programs from the enumeration above. The other malwares uses the credulousness of users to the reproduction (or the infection of the computer).

Spyware and adware programs are some special kinds of malware. They were created for commercial purposes. A spyware gathers information about computer users; an adware shows annoying advertisements to them. Since 2003 these are the most common types of malware.

Hereafter this article focuses on viruses (and partially on worms). Viruses can be sorted in many ways. There are viruses which uses existing executables (program virus) for running. Such a virus attaches itself to an executable file and it runs (infects and causes damage) when a user executes the host file. Other viruses overwrite the boot sector or the master boot sector of the disks (boot virus). These viruses are load into memory if the computer tries to read the disk while it is booting. Some viruses combine boot and program viruses (multipartite virus) or tries to avoid virus-detection using certain techniques (polymorphic and stealth viruses). Finally a new type of viruses is macro viruses that infect the macros within a document or template.

This high number of viruses and virus types calls for solutions for defend computer from the infection of any virus. There are a lot of commercial softwares (anti-virus programs) more or less efficient tools against malwares. The contest between developers of viruses and anti-virus softwares is a never-ending war which can be won by nobody.

Because softwares are often designed with security features (e. g. operating system, browsers, e-mail softwares etc.), software design and development methods and strategies have to prevent as many security holes in the software as possible. Programs should have no vulnerabilities which can be used by viruses.

3 The .NET Framework

Diversity is part of all segment of informatics. The hardware elements used for compiling computers can be very different and they can have unique characteristic depend on the manufacturer. One of the main goals of operating systems is to hide these dissimilarities and provide a uniform, hardware-independent interface for computer users (see Figure 1).

This virtual unification of the hardware-granted functions makes computer more usable, and software development will be much easier. Operating systems (their count is essentially less than the count of hardware elements) define virtual hardware units with generic properties and functions. They use the physical hardware or software emulation to grant these properties and functions. So people have to learn how they can avail themselves of the functions of the chosen operating system (they have not concern themselves about the functions of the hardware-layer).

Besides the operating systems, the most programming languages open the door to write so-called platform-independent programs (softwares uses resources available in every operating system). Up to now the programmers have had to write the program (source code) once and it has had to be compiled as many as the number of operating systems have the users (in each cases the result of this procedure is almost the same sequence of processor-instruments, but the executable file's format is different certainly).

For a long time past, programmers' wish is to make this procedure easier. They would like to write and compile the source code only once. This problem has been being solved far in the past. Interpreted programming languages are the solution.

In this case, the source code is the program which is executed by another program (interpreter). These codes can be executed with each operating system has the required interpreter.

But the interpreted programs have disadvantages: the executing procedure is too slow and the source code's syntax can not be checked fully. Sun Microsystems Inc. has given a solution for this. The programmers of Sun have created the *Java* programming language which integrate interpretation and compilation during software development and execution. After the Java source code has been written, it has to be compiled (the result is a sequence of *Java Byte Code*), so the syntax and semantics of the source code can be checked, and the code can be made more efficient. The Java Byte Code constitutes the instrument set of the *Java Virtual Machine* (JVM), so the result of compilation can be interpreted by a JVM. Because Sun has published JVMs for the most common operating systems, the once written and compiled program can be executed by all the computers run JVM.

Programmers of Microsoft solved the problem otherwise. They created the *Microsoft .NET Framework*. This solution is very similar to Java, JVM and Java Byte Code, but it is much more. It exploits the hidden power of JVM, and eliminates its weaknesses. The .NET Framework is development and execution environment for several programming languages (C\#, C++, Visual Basic etc.), so software component written in different programming languages can be assembled by .NET Framework. Moreover, the compiled source would not be interpreted; it will be compiled to native code before execution using the Just-In-Time (JIT) compiling strategy. The first compilation produces a code written using *Microsoft Intermediate Language* (MSIL).

The .NET Framework – as the Java environment – constitutes a new layer – over the operating system – which hides the dissimilarities of various operating systems (this architecture is shown on Figure 1).

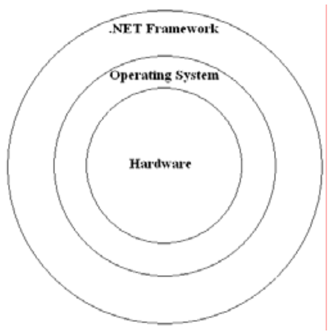

Figure 1
Layered Architecture

4 .NET Viruses

Executable files written for .NET Framework can be infected by viruses as other executables. In this case there are two ways for the infection: malicious code can be added to the file contains MSIL code-sequence or to the native code after the JIT compilation (see Figure 2).

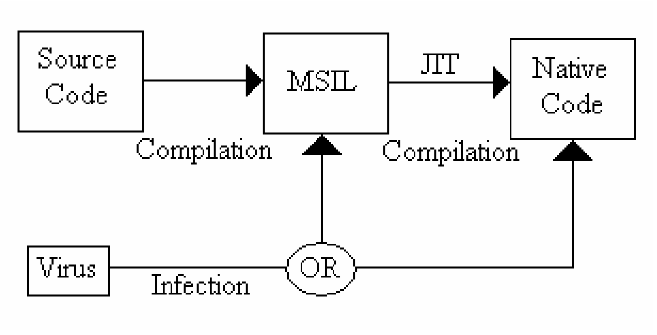

Figure 2
Infection types

If the virus infects an MSIL file, the infection will be platform independent too. The infected PE (portable executable) file can execute – without any changes – by other operating systems, so the infection can be spread very fast. While the infection of native code is 'localized' for the operating system using the same executable format.

Because lot of web services use the .NET architecture, they create new infection mechanisms unwittingly or almost unwittingly ([2]). Anti-virus professionals' opinion is the security model of .NET Framework can stop several attacks of viruses, but not all. There are other possible ways to take advantages of .NET services. Still the malwares and .NET Framework's 'collaboration' is an unanswered question.

There are several viruses using the .NET Framework in the info-space. The firs one was 'Donut' (also known as 'dotNET') which was exposed in 2002. It is a native executable targets PE files written for .NET Framework. It overwrites the initial jump to the *_CorExeMain()* function (located in *mscoree.dll*) with a jump to the end of the file where the malicious code is located. (This process is similar to the standard infection of native executables.) The virus also injects short MSIL code into the PE file for display a message that akes the fact of infection known to the user (source: virus escription from *Symantec*).

The basic form of Donut is a native executable which is an e-mail irus (it uses the address book in Microsoft Outlook to spread via -mails sent without the user's will

([3]). If this file is executed on a local computer, it infects the executables in the same folder and up to 20 parent folder.

However Donut is defined as a low-risk virus, it is a good example, that the viral infection of .NET executables are possible indeed.

Conclusions

The efforts make the program development easier faster and comfortable draw down the simply way to develop viruses and other malwares. So developers of programming frameworks (e.g. .NET) and creators of programming languages interpreted by virtual machines (e.g. Java) have to care the security leaks and features of these development tools through all ages.

The popularity of Internet and the new computing model – based on .NET Framework – used for web application creates fresh vulnerabilities can be exploit by virus (and malware) writers day-by-day. Donut is an example that good security conception (as the security of .NET Framework) can reduce this vulnerabilities, but they can be never completely removed.

References

[1] B. Krebs: A Short History of Computer Viruses and Attacks, The Washington Post, 14th February 2003

[2] J. Layden: .NET may Lead to Fewer Viruses, The Register, 28th September 2001

[3] J. Layden: C\# Virus Pitched against .NET, The Register, 4th March 2002

[4] Microsoft Knowledge Base: Information About the .NET W32.Donut Virus, Q316287, 8th July 2005