Evading Virus Detection Using Code Obfuscation

Khurram Murad, Syed Noor-ul-Hassan Shirazi, Yousaf Bin Zikria, and Nassar Ikram

National University of Science and Technology (NUST), Islamabad
khurramjarral@gmail.com, noorshirazi@gmail.com,
yusi_2@hotmail.com, dr_nassar_ikram@yahoo.com

Abstract. The conflict between malware authors and analysts is heating up as both are coming up with new armaments in their armory. Malware authors are employing novel sophisticated techniques like metamorphosis to thwart detection mechanisms while security professionals are budding new ways to confront them. In this paper we formally treat diverse mechanisms of making malware undetectable in general and code mutation techniques in particular. We also supported our argument where possible, through different tools and have revealed their outcome. In the end we give our methodology to make any virus undetectable using amalgamation of hex editing and metamorphic techniques.

Keywords: Computer virus, Polymorphism, Metamorphism, Obfuscation, Hex editing, Virus signature.

1 Introduction

A computer virus is a malicious piece of software that modifies other files to inject its code [1]. A virus can change its code on each infection [2]. Virus detection is an uncertain process [2]. Viral mutation techniques are continuously evolving and progressing to evade antivirus algorithms and tools. This resulted in more and more complex virus families of which metamorphic viruses are the most sophisticated one.

Antivirus systems use various detection techniques including signature detection and code emulation to detect malware. Signature based tools look for particular signature while code emulators execute virus in a virtual environment for detection.

To evade signature detection, virus writers continuously change virus using metamorphic techniques while keeping the same functionality. Metamorphic viruses use different code obfuscation techniques to change the structure of the code. These techniques include code reordering through jumps, subroutine permutation, dead code insertion, equivalent instruction substitution, and rearrangement of instruction order (transposition).

To evade code emulation techniques, various anti-emulation techniques have been developed by the malware writers. These include, Entry Point Obscuring (EPO) techniques, decrypting and running code chunk by chunk, using odd instructions that would deceive an emulator, random concealing of decryption, long looping through dead code, multiple encryption layers. Aforementioned techniques have some drawbacks i.e., considerable increase in size of the morphed copy and loss of functionality.

T.-h. Kim et al. (Eds.): FGIT 2010, LNCS 6485, pp. 394–401, 2010.

In this work, we focus on making virus undetectable using combination of hex editing and metamorphic techniques to address previous shortcomings.

This paper is organized as follows. Section 2 gives brief description of virus detection techniques. Section 3 give summary of techniques used to evade virus detection mechanisms. Section 4 gives brief introduction of hex editing and code obfuscation techniques and their short comings. Section 3 gives detail implementation of our proposed methodology. Finally in section 4, we present our conclusion and future work.

2 Computer Virus Architecture

Generally computer virus has the following three basic building blocks [1].

```
def virus ():
    infect ()
    if trigger () is true then
      payload ()
```

Fig. 1. Pseudo code of a computer virus [1]

In Fig 1 Infect module defines how virus spreads. It selects the target to infect and defines criteria for target selection. Trigger is the condition to decide to deliver the payload or not. Payload defines the damage done by the virus. Trigger and payload are optional.

2.1 Virus Detection Techniques

This section shows some common techniques employed by virus detection tools to detect malware.

2.1.1 Signature Detection

A signature is a string of bits found in virus [3]. Signatures are found in viruses which uniquely identify them and set them apart from normal programs. Signatures are stored in signature database and antivirus tools search for these signatures in files.

2.1.2 Heuristic Analysis

New and unknown viruses can be detected using heuristic analysis techniques. It can be static or dynamic. In heuristic analysis we can analyze file format, code structure as well as we can do code emulation for virus detection. Heuristic analysis can be very noisy sometimes as it creates many false positives. Heuristic analysis is not an accurate method of detecting viruses.

Following section discusses techniques employed by malware writers to thwart detection by signature detection and heuristic analysis.

3 Code Obfuscation Techniques

To evade antivirus tools malware use different obfuscation techniques. Some are listed below.

3.1 Encryption

Encryption changes virus appearance. It consists of small decryptor stub and encrypted virus code. Virus body can be changed using different keys but decryptor remains the same, so it can be detected.

3.2 Polymorphism

To evade detection, polymorphic virus changes decryptor and virus body as well. To detect such virus code emulation techniques are used because virus body decrypts into the same virus code so it can be detected.

3.3 Metamorphism

Metamorphic viruses do not apply encryption. They change appearance of code while keeping the functionality intact. They use several code obfuscation techniques including garbage code insertion, Instruction reordering, data reordering, register renaming, subroutine in-lining, subroutine outlining, code permutation, and instruction substitution.

Commonly metamorphic viruses have embedded metamorphic engines which generate morphed copy of it using metamorphic engine. A metamorphic engine has following typical functional units as shown in Fig 2.

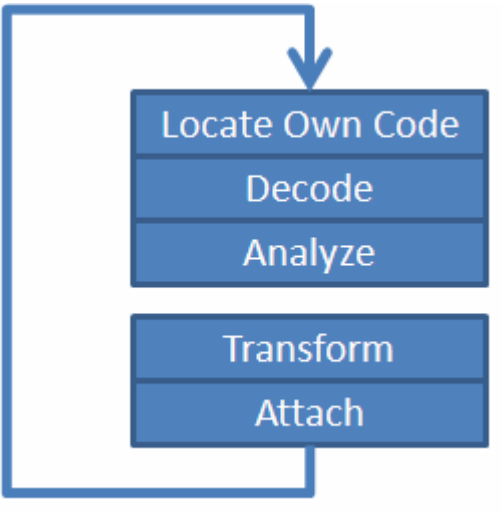

Fig. 2. Metamorphic engine functional units [4]

Metamorphic engine takes virus as input locates code to be transformed using its own customized rule set. Decode module extracts the rules by disassembling and analyze module determine transformation to be applied. Transform module applies actual transformations and attach module attaches morphed copy to a host.

Code obfuscation techniques operate on both control flow and data section of the program in assembly programs [5]. Table 1 gives summary of well known metamorphic viruses and code obfuscation techniques used by them.

Table 1. Metamorphic viruses and code obfuscation Techniques [5]

	EVOL (2000)	ZMIST (2001)	ZPERM (2000)	REGSWAP (2000)	METAPHOR (2001)
Instruction Substitution				✓	
Instruction Permutation	✓	✓			✓
Dead code Insertion	✓	✓			✓
Variable Substitution	✓	✓		✓	✓
Changing the Control Flow		✓	✓		✓

4 Hex Editing and Metamorphism

The idea behind hex editing is to find the signature in a virus code that antivirus software uses for detection and then change it [6], however, this cannot guarantee 100% virus functionality.

Commonly metamorphic viruses have embedded metamorphic engines which allow them to generate morphed copy on the fly. These metamorphic engines enable them to use extensive code obfuscation to make viruses undetectable. This technique essentially makes viruses undetectable but there is a considerable increase in virus size after each iteration [7].

5 Making It Happen: Practical Approach

We have used an amalgamation of hex editing and code obfuscation techniques to guarantee the functionality of a virus while making minimal changes to limit the size of virus in considerable bounds. Increase in size can make a virus resource-hungry. Therefore, we have used hex editing to locate the virus signature where code obfuscation needs to be applied. We have taken Evol which is a metamorphic virus itself. We will not discuss its metamorphic engine or behavior because it is out of scope of our research. The proposed technique is highly independent of Evol and can be applied on any virus to make it undetectable.

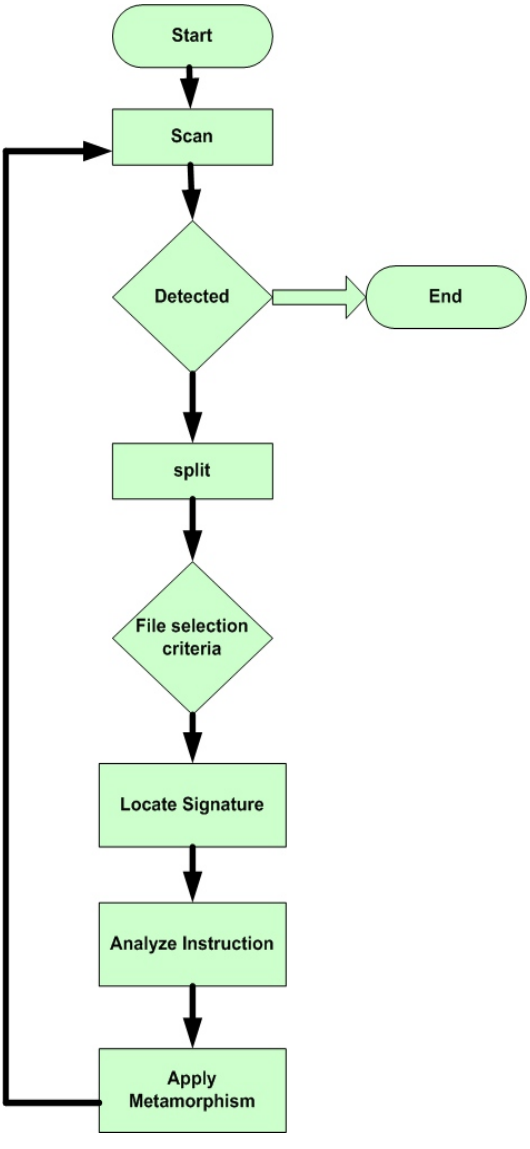

Fig. 3. Methodology Flow Diagram

5.1 Methodology

Procedure of applying metamorphic techniques is as follows:

1. Scan virus file 'V' which is a virus executable with an antivirus software.
2. If file 'V' is detected as virus
   Then

i. Split file 'V' in multiple file {V1, V2… Vn}
Such that consecutive files have one byte difference and their size is in incrementing order.

3. Scan files again.
4. Pick 2 files V(p-1) and Vp
Such that p > 1
Which satisfy these conditions
   i. V(p-1) and Vp have 1 byte difference.
   ii. Size(V(p-1)) < size(Vp)
   iii. V(p-1) is not detected by the antivirus software.
   iv. Vp is detected as a virus by the antivirus software.
5. Find location of last byte of Vp in 'V' and locate the assembly instruction that contains this byte.
6. Analyze instruction and apply appropriate code obfuscation technique
7. Repeat step 1 to 6 till 'V' is fully undetectable by the antivirus software.

Flow of our methodology is shown in Fig 3.

5.2 Implementation

We have taken a variant of Evol virus Virus.Win32.Evol.a as a test case which is detectable by our antivirus software as shown in Fig 4.

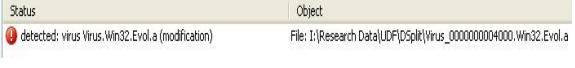

Fig. 4. Initial scan Evol detected

First we scanned the base virus with antivirus software and it was detected. This variant of Evol has size of 12,288 bytes. Then we split Evol viruses into files having size difference of 1000 bytes from preceding file. After splitting 13 files are generated. Each file has size 1000 bytes greater than the preceding file. Then after splitting, we scanned our files as shown in Fig 5.

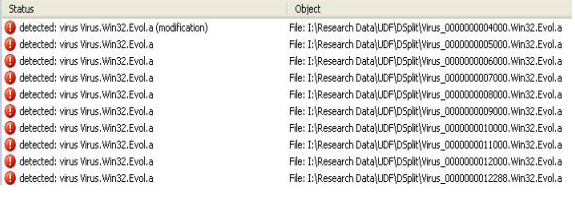

Fig. 5. Scan results after splitting

Antivirus detected all files from 4000.Win32.Evol.a to 12288.Win32.Evol.a which reveals that virus signature is certainly present in 4000.Win32.Evol.a and is not present in 3000.Win32.Evol.a. Now we repeated same process of splitting, making 3000

as start byte and 4000 as max byte having size difference of 100 bytes to narrow down the search for virus signature. After repetition of splitting and scanning process, we have narrowed down our search of signature up to file size difference of 1 byte as shown in Fig 6. The first detectable split file in incrementing order contains virus signature in the last bytes that we have to change in order to make Evol undetectable.

Fig. 6. Scan results after splitting of file size difference of 1 byte.

Now our challenge is to identify the address of signature byte and assembly language instruction which contains this byte. To locate the address of byte in executable of Evol, we can use any portable executable (PE) format editor as shown in Fig 7, whereas for assembly instruction any debugger utility can be used.

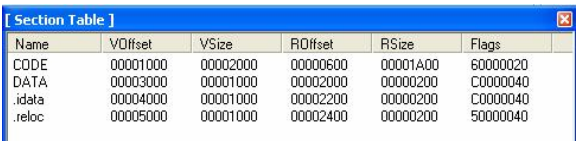

Fig. 7. Evol segments sizes

Signature byte address can be located in the executable using the following formula.

$$\alpha = \beta - \gamma + \tau + \Omega. \quad (1)$$

Where
α = RVA of Signature Byte
β = Raw Offset of Signature byte
γ = Raw Offset of Section
τ = Virtual Offset of Section
Ω= Image Base

Once we have identified the location, we need to apply one of above mentioned code obfuscation technique. In our case we have applied subroutine in-lining technique to change the signature and keep the functionality of Evol unaltered.

After successful implementation of code obfuscation, our scanning results show that Evol is undetectable and fully functional as shown in Fig 8.

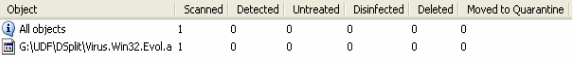

Fig. 8. Scan Result after applying code obfuscation

6 Conclusion

We have demonstrated successfully that malware can be made undetectable using code obfuscation techniques applying minimal changes by locating the signature. It is a challenge for antivirus community to cater this new generation of virus species that employ advanced metamorphic techniques. We have applied code obfuscation using subroutine in-lining on Evol which showed that there was no considerable increase in size. Original Evol has size of 12,288 bytes and our variant of Evol has the same size and functionality as of the original Evol virus while achieving its un-detect ability.

We proposed a methodology for producing morphed copies of a base virus that have the same functionality as the base virus and have minimal impact on size of the morphed copies. Code obfuscation is applied only where signature is detected in the base virus. Future work would be to develop a metamorphic engine that automates this process.

References

1. Aycock, J.: Computer Viruses and malware, Springer Science+Business Media (2006)
2. Cohen, F.: Computer viruses: theory and experiments. Computer Security 6(1), 22–35 (1987)
3. Stamp, M.: Information Security: Principles and Practice (August 2005)
4. Walenstein, R., Mathur, M., Chouchane, R., Lakhotia, A.: The design space of metamorphic malware. In: Proceedings of the 2nd International Conference on Information Warfare (March 2007)
5. Borello, J., Me, L.: Code Obfuscation Techniques for Metamorphic Viruses (Feburary 2008), http://www.springerlink.com/content/233883w3r2652537
6. Techotips (2009), http://techotips.blogspot.com/2009/10/tutorial-hexing-using-dsplit-hide.html
7. Desai, P.: Towards an Undetectable Computer Virus, Master's thesis, San Jose State University (December 2008)