Computer virus strategies and detection methods

Article · October 2008

CITATIONS
77

READS
26,287

3 authors:

Essam Al Daoud
Zarqa University
51 PUBLICATIONS **418** CITATIONS
SEE PROFILE

Iqbal Jebril
Al-Zaytoonah University of Jordan
122 PUBLICATIONS **727** CITATIONS
SEE PROFILE

Belal Zaqaibeh
Jadara University
23 PUBLICATIONS **206** CITATIONS
SEE PROFILE

Int. J. Open Problems Compt. Math., Vol. 1, No. 2, September 2008

Computer Virus Strategies and Detection Methods

Essam Al Daoud[1], Iqbal H. Jebril[2] and Belal Zaqaibeh[3]
[1]Department of Computer Science, Zarqa Private University, Jordan
e-mail: essamdz@zpu.edu.jo
[2] Department of Mathematics, King Faisal University, Saudi Arabia
[3]Department of Computer Science, Jadara University, Jordan

Abstract

The typical antivirus approach consists of waiting for a number of computers to be infected, detecting the virus, designing a solution, and delivering and deploying the solution, in such situation, it is very difficult to prevent every machine from being compromised by virus. This paper shows that to develop new reliable antivirus software some problems must be solved such as: a new method to detect all metamorphic virus copies, new reliable monitoring techniques to discover the new viruses or attaching a digital signature and a certificate to each new software.

Keywords: *Antivirus, Digital Signature, Dynamic Detection, Metamorphic Virus and Static Detection.*

1 Introduction

In late 1992, the number of computer viruses was estimated from 1,000 to 2,300 viruses. In 2002 there were 60,000 known viruses, Trojans, worms, and variations. Today there are well over 100,000 known computer viruses [1]. Studies and researches show that a computer connected to the Internet may experience an attack every 39 seconds [2]. New vulnerabilities in the system are discovered every few days. These vulnerabilities are fixed by the software vendors who provide patches and updates for the system. However, during this process the vulnerabilities exploit by hackers, where malicious programs are installed on user machines to steal secret data for financial gains. The compromised machines can also be made a part of a huge botnet that can be used to launch Denial of Service attacks on servers, or be used in an attempt to intrude the computers of government agencies [3].

Unfortunately, our current ability to defend against new viruses is extremely poor and the basic approach of detection, characterization, and containment has not changed significantly over the last five years. The complexity of modern malware is making this problem more difficult. For example, Agobot, has been observed to have more than 580 variants since its initial release. Modern Agobot variants have the ability to perform denial of service attacks, steal bank passwords and account details, propagate over the network using a diverse set of remote exploits, use polymorphism to evade detection and disassembly [4].

Computer virus writers use many strategies to evade detection such as space filling, compressing and encryption, in another hand; the antivirus software trying to detect the viruses by using variant static and dynamic methods. However; all the existing methods are not adequate. To develop new reliable antivirus software some problems must be fixed. Antivirus open problems are introduced at the end of this paper.

2 Strategies of Computer virus

A computer virus is a computer program that can copy itself and infect a computer without permission or knowledge of the user. In order to avoid detection by users, some viruses employ different kinds of deception such as the following Strategies[5], [6]:

- **Overwriting Virus:** this type of virus overwrites files with their own copy. Of course, this is a very primitive technique, but it is certainly the easiest approach of all. Overwriting viruses cannot be disinfected from a system. Infected files must be deleted from the disk.
- **Companion Infection**: one approach to becoming a companion to an EXE file is to give the virus the same base name as the targeted program, but use a .COM extension instead of .EXE. This technique was employed by the Globe virus, first detected in 1992. When the victim attempts to launch an EXE program, he or she usually types its name without the extension. In such cases, Windows gives priority to a file with the .COM extension over a file with the same base name but with the .EXE extension.
- **Appending Virus:** In this technique, a jump (JMP) instruction is inserted at the front of the host to point to the end of the original host. A typical example of this virus is Vienna**.** The appender technique can be implemented for any other type of executable file, such as EXE, NE, PE, and ELF formats, and so on. Such files have a header section that stores the address of the main entry point, which, in most cases, will be replaced with a new entry point to the start of the virus code appended to the end of the file.
- **Prepending Virus:** This virus inserts its code at the front of host programs. This is a simple kind of infection, and it is often very successful. Virus

writers have implemented it on various operating systems, causing major virus outbreaks in many. An example of a COM prepender virus is the Hungarian virus Polimer.512.A, which prepends itself, 512 bytes long, at the front of the executable and shifts the original program content to follow itself.

- **Cavity** or **spacefiller Virus:** This virus attempts to install itself in this empty space while not damaging the actual program itself. An advantage of this is that the virus then does not increase the length of the program and can avoid the need for some stealth techniques. The Lehigh virus was an early example of a cavity virus. Because of the difficulty of writing this type of virus and the limited number of possible hosts, cavity viruses are rare**.**
- **Compressing Virus**: A special virus infection technique uses the approach of compressing the content of the host program. Sometimes this technique is used to hide the host program's size increase after the infection by packing the host program sufficiently with a binary packing algorithm.
- **Encrypted Virus**: consists of a constant decryptor, followed by the encrypted virus body. Relatively easy to detect because decryptor is constant. The first known virus that implemented encryption was Cascade on DOS. Oligomorphic virus changes its decryptors in new generations. The simplest technique to change the decryptors is to use a set of decryptors instead of a single one. The first known virus to use this technique was Whale. Whale carried a few dozen different decryptors, and the virus picked one randomly.
- **Boot Sectors Virus:** this virus takes advantage of the executable nature of master boot record (MBR) and partition boot sector (PBS). A PC infected with a boot sector virus will execute the virus's code when the machine boots up. Michelangelo virus is an example of a Boot Sectors Virus.
- **macro virus**: infects a Microsoft Word or similar application and causes a sequence of actions to be performed automatically when the application is started or something else triggers it. Macro viruses tend to be surprising but relatively harmless. A typical effect is the undesired insertion of some comic text at certain points when writing a line. A macro virus is often spread as an e-mail virus. A well-known example in March, 1999 was the Melissa virus.
- **Malicious mobile code** (MMC): mobile code is a lightweight program that is downloaded from a remote system and executed locally with minimal or no user intervention. Java applets, JavaScript scripts, Visual Basic Scripts (VBScripts), and ActiveX controls are some of the most popular examples of mobile code that you may encounter while browsing the Web or reading HTML-formatted e-mail. An attacker might use mobile code for a variety of nasty activities, including monitoring your browsing activities, obtaining unauthorized access to your file system, infecting your machine with a Trojan horse, hijacking your Web browser to visit sites that you did not intend to visit, and so on.

3 Static Detection Methods

With static analysis, a virus is detected by examining the files or records for the occurrences of virus patterns without actually running any code. Static Methods include the following methods [7]:

- **String Scanning method**: Searches for sequence of bytes (strings) that are typical of a specific virus but not likely to be found in other programs.
- **Wildcards method**: allows to skip bytes or byte ranges. For example "?" character are skipped and the wildcard % means that the scanner will try to match the next byte.
- **Mismatches method**: allows any given number of bytes in a string to be of arbitrary value, regardless of their position.
- **Generic Detection method**: This technique uses one common string to detect several or all known variants of a family of viruses.
- **Bookmarks method**: calculates the distance between the start of the virus body and the detection string.
- **Smart Scanning**: Smart scanning could skip junk instructions, such as NOPs, in the host file and also did not store them in the virus signature. To enhance the likelihood of detecting related variants of viruses, an area of the virus body was selected which had no references to data or other subroutines.
- **Skeleton Detection**: The scanner parses the statements of the virus line-by-line and drops all nonessential statements. What is left is the skeleton of the body that has only essential macro code common in macro virus.
- **Heuristics Analysis**: Heuristic analysis is an expert based analysis that determines the susceptibility of a system towards particular threat/risk using various decision rules or weighing methods. MultiCriteria analysis (MCA) is one of the means of weighing.
- **Virus specific detection**: There are cases when the standard algorithm of the virus scanner cannot deal with a virus. In cases like this, a new detection code must be introduced to implement a virus-specific detection algorithm. This method includes Filtering, Decryptor Detection and X-Ray scanning.

4 Dynamic Detection Methods

Dynamic detection method decides whether or not code is infected by running the code and observing its behavior. The program monitors known methods of virus activity including attempts to infect and evade detection. This may also include attempts to write to boot sectors, modify interrupt vectors, write to system files, etc. For example, most virus activity eventually needs to call some system functionality, like I/O operations - only these actions have to be considered. No matter how obfuscated the I/O calls are statically, the calls will appear clearly when the code runs. Software monitors work best when the normal usage

characteristics of the system are vastly different from the activity profile of an infected system. A virus might exhibit a dynamic signature like [6]:

- Opening an executable, with both read and write permission.
- Reading the portion of the file header containing the executable's start address.
- Writing the same portion of the file header.
- Seeking to the end of the file.
- Appending to the file.

A behavior blocker is antivirus software which monitors a running program's behavior in real time, watching for suspicious activity. If such activity is seen, the behavior blocker can prevent the suspect operations from succeeding, can terminate the program, or can ask the user for the appropriate action to perform. Behavior blocking allowed code to run on the real machine. In contrast, antivirus techniques using emulation let the code being analyzed run in an emulated environment. The hope is that, under emulation, a virus will reveal itself. Because any virus found wouldn't be running on the real computer, no harm is done.

5 Metamorphic Virus

Metamorphic Virus can reprogram itself. it use code obfuscation techniques to challenge deeper static analysis and can also beat dynamic analyzers by altering its behavior, it does this by translating its own code into a temporary representation, edit the temporary representation of itself, and then write itself back to normal code again. This procedure is done with the virus itself, and thus also the metamorphic engine itself undergoes changes. Metamorphic viruses use several metamorphic transformations, including Instruction reordering, data reordering, inlining and outlining, register renaming, code permutation, code expansion, code shrinking, Subroutine interleaving, and garbage code insertion. The altered code is then recompiled to create a virus executable that looks fundamentally different from the original. For example, here is the original code of a target before instruction replacement [7], [8]:

```
55          push ebp
8BEC        mov ebp, esp
8B7608      mov esi, dword ptr [ebp + 08]
85F6        test esi, esi
743B        je 401045
8B7E0C      mov edi, dword ptr [ebp + 0c]
09FF        or edi, edi
7434        je 401045
31D2        xor edx, edx
```

and here is the original code of a target after instruction replacement

```
55          push ebp
54          push esp ;      register move replaced by
5D          push/pop
8B7608      pop ebp ;       register move replaced by
09F6        push/pop
743B        mov esi, dword ptr [ebp + 08]
8B7E0C      or esi, esi ;     test/or interchange
85FF        je 401045
7434        mov edi, dword ptr [ebp + 0c]
28D2        test edi, edi ;  test/or interchange
            je 401045
            sub edx, edx ; xor/sub interchange
```

Figure 1 shows an example of inserting jumps into code (Zperm virus) [9]:

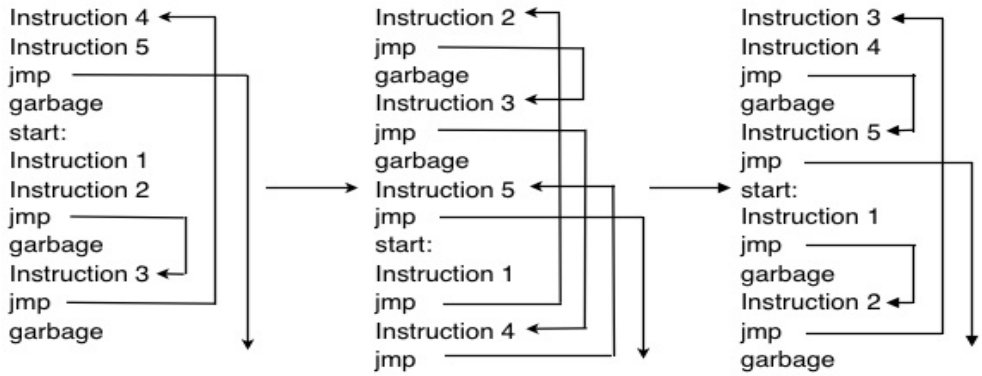

Figure 1: Example of Zperm inserting jumps into its code

In general any metamorphic virus must perform the following steps[10]:

- **Locate own code**: Each time a metamorphic engine is called to transform some code, it must be able to locate their own code in the new variants.
- **Decode**: Next, the engine needs to decode the information required to perform the transformations. In order to transform itself, the engine must have some representation of itself so that it knows how to make the transformations.
- **Analyze**: In order for the metamorphic transformations to work correctly, certain information must be available. For some transformations to be performed correctly, the engine must have register liveness information available. If such information is not available, the metamorphic engine itself must construct it. A piece of information that is frequently required for analysis and transformation, is the control flow graph (CFG) of the program.

- **Transform**: This unit is responsible for transforming the code into equivalent code. This is done usually by replacing instruction blocks in the code with other equivalent.
- **Attach**: The last step is attaching the new generation of the virus to a new host file.

6 Antivirus Open Problems

Detection Methods have some major problems. Firstly, they are only good against known viruses and not very good against evolutionary or new viruses. Secondary, they tend to take a noticeable amount of time to scan a system or networks for the patterns. Thirdly, a scanner or its virus pattern database must be updated very often to remain effective. Subsequence the following problems must be solved:

- If the virus is cleverly written to always stay within this normal behavior, it may be difficult to detect its presence using the current monitoring techniques. Can we introduce new reliable monitoring techniques to discover the new viruses?
- Metamorphic viruses are difficult to detect because their creators have the advantage of knowing the weaknesses of antivirus scanners. The limits of antivirus scanners come from the limits of static and dynamic analysis techniques. If we have some copies of a metamorphic virus, is there a new method to detect all metamorphic virus copies?
- Can we use public key cryptography to solve the computer virus problem? In this case all the developers must embed their digital signature within their software and they must prepare a certificate that is signed by a will known certificate authority. The developers of operating systems must offer a new procedure to copy, download and install the new software.

References

[1] http://www.cknow.com/vtutor/NumberofViruses.html. Last access on July (2008)

[2] http://csdl2.computer.org/comp/mags/it/2007/02/f2004.pdf. Last access on July (2008)

[3] R. Srinivasan , *Protecting Anti-Virus Software Under Viral Attacks*, Master Degree of Science, Arizona State University (2007).

[4] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario, "Automated classification and analysis of internet malware", In *Proceedings of the 10th Symposium on Recent Advances in Intrusion Detection (RAID'07)*, (2007), pp 178–197.

[5] J. cock, *Computer Viruses and Malware*, Springer (2006)

[6] E. Skoudis and L. Zeltser, *Malware: Fighting Malicious Code*, Prentice Hall (2003)
[7] P. Szor, *The Art of Computer Virus Research and Defense*. Addison Wesley, (2005)
[8] E. Konstantinou, "Metamorphic Virus: Analysis and Detection", *Technical Report RHUL-MA-2008-02*, (2008)
[9] P. Szor and P. Ferrie, "Hunting for metamorphic", *In Virus Bulletin Conference, Prague, Czech Republic Vecna (1998). Miss Lexotan,8, 29A E-Zine* (2001)
[10] A. Walenstein, R. Mathur, R. Mohamed, R. Chouchane and A. Lakhotia. "The design space of metamorphic malware", *In Proceedings of the 2nd International Conference on Information Warfare*, *(*2007).