BYPASSING ANTIVIRUS AND ANTIVIRUS VULNERABILITIES

Aryan Jadon[1], Dr. Gagandeep Kaur[2]
[1]JIIT, Noida, India , [2]Dept. of CSE & IT JIIT, Noida, India

ABSTRACT

This paper describes the findings of the research concentrating on Bypassing antivirus and defense against malware. The overall goal is to study the areas of penetration testing. The focus is on finding how the antivirus works and how to exploit its limitations and finding solutions to those limitations.

Malware is used to perform multiple offensive activities: launching distributed denial of service attacks, collecting classified information etc. Consequently, testing and understanding the efficiency and configurations of malware defense system is of uttermost importance.

The paper describes how antivirus programs can be bypassed. it also describes how to prevent your credentials from malware. The study of this paper will tell you that how an antivirus programs detects any suspicious activity in your machine.

Keywords: Bypassing antivirus, antivirus evasion, methods for antivirus evasion.

INTRODUCTION

Antivirus software is a program or set of programs that are designed to prevent , search for, detect and remove software viruses, and other malicious software like worms , Trojans, adware etc. There are various ways to analyses or scan any information based on where it comes from.

Antivirus is mandatory in defence in depth. After doing some research, we came to the conclusion that bypassing antivirus consists of two steps [6-10]:

1) Hide the code which may be recognized as malicious. This is generally done using encryption.
2) Code the decryption stub in such a way that it is not detected as a suspicious activity or virus not bypassed by emulation /sandboxing.

Antivirus has to deal with hundreds of file types and formats [1-6]:

1) Executable files like .exe, .dll, .msi, .com, .pif, .cpl, .elf , .ocx, .sys, .scr.
2) Documents files like .doc, .xls, .ppt, .pdf, .rtf, .chm, .hlp.
3) Compressed archives like .arj, .arc, .cab, .tar, .zip, .rar, .lzh , .ace, .iso.
4) Executable packers like. upx, .fsg, .mew, .nspack, .wwpack, .aspack
5) Media files like .jpg, .gif, .swf, .mp3, .rm, .wmv, .avi, .wmf

It is amply clear from recent research into antivirus vulnerabilities [13]. It reveals that most vulnerabilities exist in the executable decompression components and data compression components. Antivirus software will try to decompress the compressed executable file and data before processing them.

BACKGROUND STUDY

HOW ANTIVIRUS DETECTS MALWARE?

1) STATIC SIGNATURE ANALYSIS [10]

Signature analysis is based on a blacklist method. When a new malware is detected by antivirus analyst, a signature is issued. Most of the antivirus solutions are signature based.

These systems search executables and other documents for strings of character, known to occur in specific pieces of malware. If a file contains the exact same string as one the string in the antivirus database, the file will be detected as malicious otherwise it will not.

It is still possible to build a signature on an encrypted malicious code when looking at specific instructions in decryption stub.

From the offense side, studies have shown that approximately more than 20,000 new stains of malware appear every day [12]. For a signature based antivirus to accurately detect all these strains it would require knowledge of every single strain released, which is quite an impossible task. Some malware is bound to be missed.

2) STATIC HEURISTIC ANALYSIS [10]

Static Heuristic are a branch of heuristic techniques that try to determine if a suspect program is malicious by examining the structure and contents of the program in an inactive state and trying to find code fragments that have been commonly used for malicious end in the past.

This type of technique is especially dependent on having detailed knowledge not only of the contents past malware but also the contents of legitimate programs so as to avoid alerting on the presence of code that though heavily used by malware is also common in legitimate software.

One of the strengths of static heuristic is that unlike dynamic heuristic it is able to examine multiple possible program execution paths due to the fact that it's looking at all the contents of the program instead of just code that would get executed during one particular invocation of the program.

Unfortunately, malware writers have developed a number of techniques to obfuscate their code in such way as to present a heuristic engine from being able to see the actual code and thus preventing it from performing static analysis on that code.

3) Dynamic analysis [10]

Nowadays most of the antivirus will rely on a dynamic approach. When an executable is scanned, it is launched in a virtual environment for a short amount of time. Indeed, the code is self-decrypted in antivirus sandbox, then analysis of the new code can trigger some suspicious behavior. If one uses encryption/decryption stub to hide a malicious, most antivirus will be able to detect it provide they can bypass the decryption phase.

This means that bypassing dynamic analysis implies two things:

1. Having an undetectable self-decryption mechanism.
2. Prevent the antivirus to execute the decryption stub.

ANTIVIRUS LIMITATIONS

Dynamic analysis is a complex stuff, being able to scan millions of file, running them on emulated environment, checking all signatures, it also has limitations.

The dynamic analysis model has three main limitations which can be exploited [6-7]:

1. Scans has to very fast so there is a limit to the number of operations it can run for each scan at a particular time.
2. The environment is emulated not so aware of the specific of the machine and malware environment
3. The emulated/sandbox environment has some specification which can be detected by the malware

Recent researches [1-5] have shown that most of the malwares/vulnerabilities exist in executable decompression and data decompression components.

Any mistake in these throws open door for the vulnerabilities.

BYPASSING ANTIVIRUS

The more options you have to re-encode your malware, the better chance you have of re-encoding malware to get past of antiviruses.

For the antivirus evasion research purpose, we used ***shellter*** **[16]**. ***Shellter*** is a program, which is capable of re-encoding any native 32-bit standalone windows application.

Since we are trying to avoid antivirus detection, we need to avoid anything that might look suspicious to antivirus software such as packed applications or applications that have more than one section containing executable code.

Next, we have discussed various methods for bypassing antivirus

1. METHOD A:

We created a c code, Function 1, calling non encoded Meterpreter shellcode.

Here's the main function:

```
int main(void)
{
Decryptcodesection();
Startshellcode();
return 0;
```

Function 1: C code of Meterpreter shellcode.

This version of the code has a virus total score of **2/56 [17].**

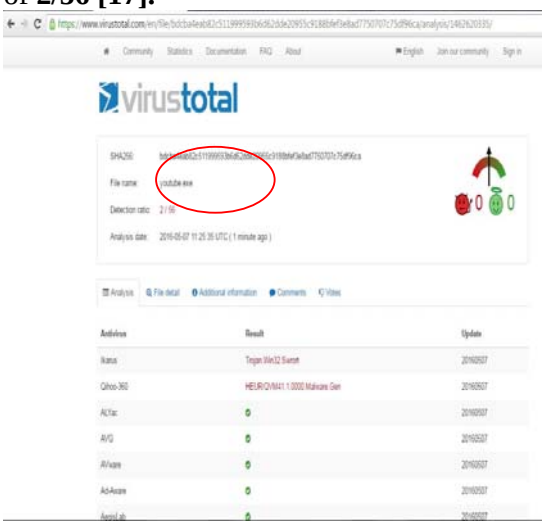

FIGURE 1- Screenshot of virus total with score of 2/56.

Figure 1 shows the screenshot of results received from Function 1.

This shows that nowadays antivirus relies more and more on dynamic analysis but it is not yet the case for the majority of them.

The main limit with antivirus scanner is the amount of time they can spend on each file. During a regular system scan, antivirus will have to analyse thousands of files.it just cannot spend too much time or power on a single file.

So based on this fact, the simplest way to bypass an antivirus just consist into buying enough time before the code is decrypted.

2. METHOD B:

We allocated and filled 100 megabytes of memory. This is enough to discourage any emulation antivirus out there.

Here's the copy of the main function:

```
#define MEM 100000000
int main()
{
char * memdmp=NULL;
memdmp=(char *) malloc(MEM);
if(memdmp!=NULL){
memset(memdmp,00,MEM);
free(memdmp);
decryptCodeSection();
startShellCode();
}
return 0;
}
```

Function 2: C code of allocating 100 megabytes of memory

This version of the code has a virus total score of **0/56[17].**

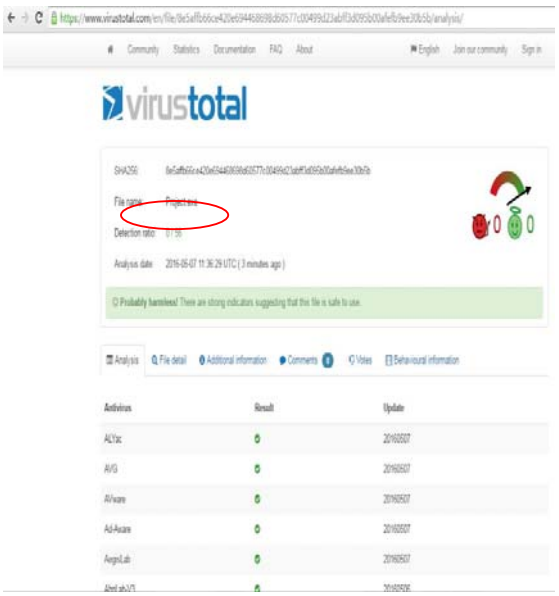

FIGURE 2- Screenshot of virus total with Score of 0/56.

Figure 2 shows the screenshot of results obtained from Function 2.

This method relies on classic and very common malloc function and does not need any strings which could be used to build signature.

3. METHOD C:

This time we will use non uniform memory access. It is a method to configure memory management in multiprocessing systems.it is

linked to a whole set of functions declare in Kernal32.dll.

Here's the copy of the main function :

```
int main( void ) {
LPVOID mem = NULL;
mem =
VirtualAllocExNuma(GetCurrentProcess
(), NULL, 1000, MEM_RESERVE |
MEM_COMMIT,
PAGE_EXECUTE_READWRITE,0);
if (mem != NULL)
{
decryptCodeSection();
startShellCode();
}
return 0;
}
```

Function 3: C code to configure memory management.

This version of the code has a virus total score of **0/56[17].**

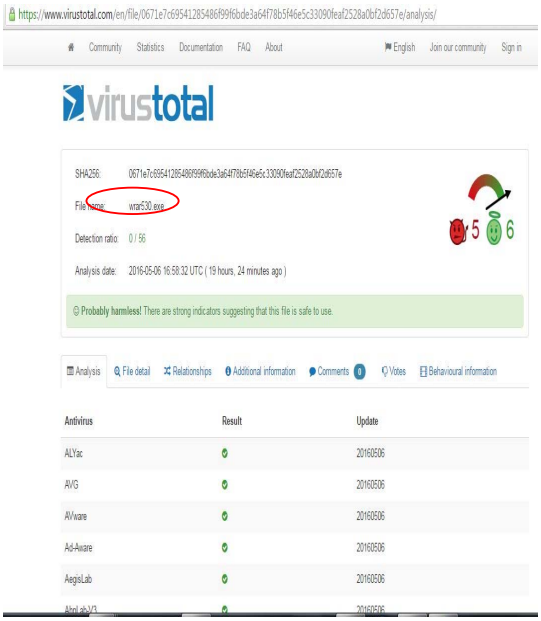

FIGURE 3- Screenshot of virus total with score of 0/56.

Figure 3 shows the screenshot of results obtained from Function 3.

4. METHOD D:

Antivirus are generally not able to follow parent and child process. They will scan the parent and not the child process even if it is in fact the same code.

In this code, it will only start decryption code if a certain mutex object already exists on the system. The method is that if the object does not exist, this code will create and call a new instance to itself. The child process will try to create the mutex before the father process dies and will fall in to the error code branch.
Here's the copy of the main function:

```
int main()
{
HANDLE mutex;
mutex = CreateMutex(NULL, TRUE,
"muuuu");
if (GetLastError()
==ERROR_ALREADY_EXISTS) {
decryptCodeSection();
startShellCode();
}
else
{
startExe("File.exe");
Sleep(100);
}
return 0;
}
```

Function 4 : C code for decryption of mutex object.

This version of the code has a virus total score of **0/56[17].**

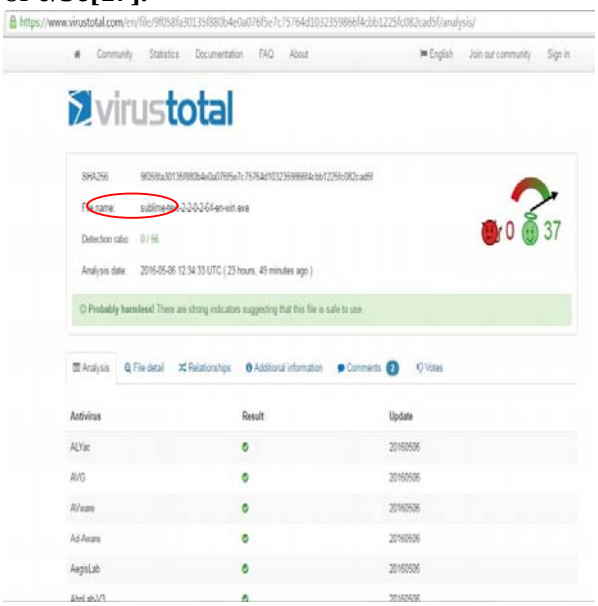

FIGURE 4- Screenshot of virus total with score of 0/56.

Figure 4 shows the screenshot of results obtained from Function 4.

DEFENSE AGAINST MALWARES

Recommendations against malwares:

1. Never run as administrator. It's a golden rule, it can avoid 99% malwares without having an AV.
2. Harden the systems, with recent versions of Windows have really strong security features, use them.
3. Invest in Network Intrusion Detection Systems and monitor your network. Often, malware infections are not detected on the victim's PC but thanks to weird NIDS or firewall logs.
4. Use several AV products from different vendors. One product can cover the weakness of another

CONCLUSION:

In this paper we have examined the techniques of finding vulnerabilities in antivirus software.
Bypassing antivirus is simple when you exploit their weakness.it requires some knowledge on windows system and how antivirus works. Antivirus is useful in detecting millions of wild bots which are already in database, also they are useful for system recovery.

REFRENCES:

1. Lawrence Teo, Yuliang Zheng ,Gail-Joon Ahm.Intrusion Detection Force: An Infrastructure for Internet-Scale Intrusion Detection.Proceedings of the first IEEE Internation Workshop on Information Assurance,2003.
2. Skoudis E. Counter Hack Reloaded: A step-by -step Guide to Computer Attacks and Effective Defense, 2nd ed. Prentice Hall ,2006
3. I'Connor, R. (2011). Violent Python: A Cookbook for Hackers, Forensic Analysts, Penetration Testers and Security Engineers. In *Violent Python: A Cookbook for Hackers, Forensic Analysts, Penetration Testers and Security Engineers* (pp. 245-256). Syngress.
4. Jon Oberheide, M. B. (2009). PolyPack: an automated online packing service for optimal antivirus evasion. *USENIX association Berkeley*.
5. Maciej Korczynski, G. B.-S. (2013). Two Methods for Detecting Malware. *Multimedia Communications, Services and Security*.
6. Antivirus in security- http://events.ccc.de/camp/2007/Fahrplan/attachments/1324AntivirusInSecuritySergioshadownAlvarez.pdf
7. Bypassing antivirus packet storm- https://dl.packetstormsecurity.net/papers/bypass/bypassing-av.pdf
8. Windows Sysinternals- https://technet.microsoft.com/enus/sysinternals/default.aspx
9. National Vulnerabilities database- http: //nvd.nist. gov/home. cfm
10. Searchsecurity- http://www. searchsecurity. techtarget.com/
11. Help net Security https://www.helpnetsecurity.com/
12. Antivirus ware- https://www.antivirusware.com/
13. Symantec Analysis – https://www.symantexanalysis.com
14. Search-Security-http://searchsecurity. techtarget.com/tip/ How-antivirus-software-works-Virus-detection-techniques
15. Panda Security- https://en.wikipedia.org/wiki/Panda_Security
16. Shellter Project – https://www.shellterproject.com
17. Virus Total – https://www.virustotal.com