Day 1: Simulating and Detecting Windows PowerShell events

Goal: Simulate suspicious PowerShell activity on Windows and detect it using logs.

Requirements:

- **Systems:** Windows 10/11 or Windows Server 2019/2022, Linux (Ubuntu or CentOS)
- **Tools:**
  - **Windows Event Viewer**
  - **PowerShell (Pre-installed on Windows)**

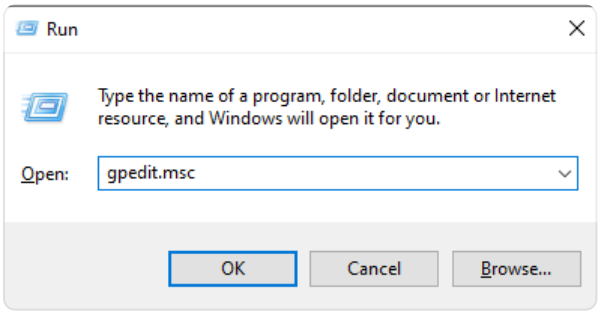
Step 1: Enable PowerShell Logging

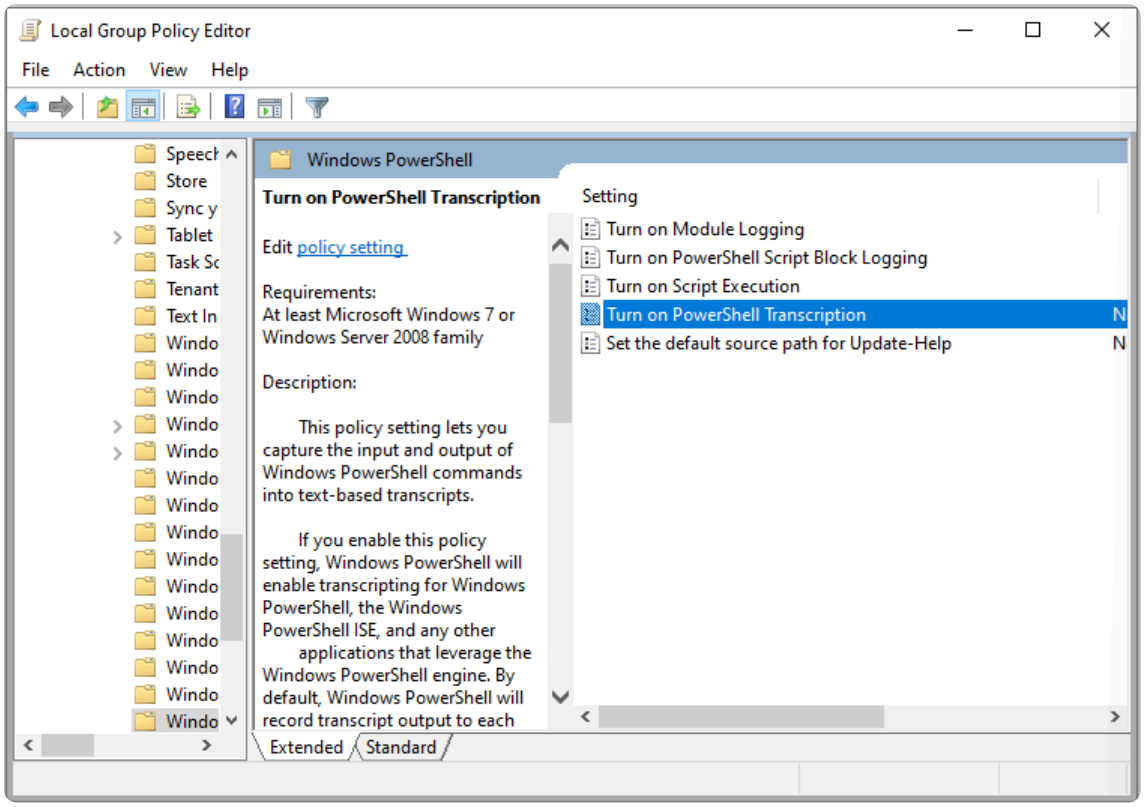
Press **Win + R**, type `gpedit.msc` → press Enter.
This opens the **Group Policy Editor**.

Navigate to:

```
Computer Configuration → Administrative Templates → Windows Components → Windows PowerShell
```

Enable these settings:

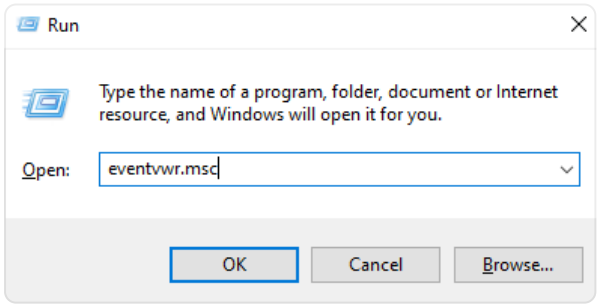
- **Module Logging** → Enabled
- **Script Block Logging** → Enabled
- **Script Execution** → Enabled
- 

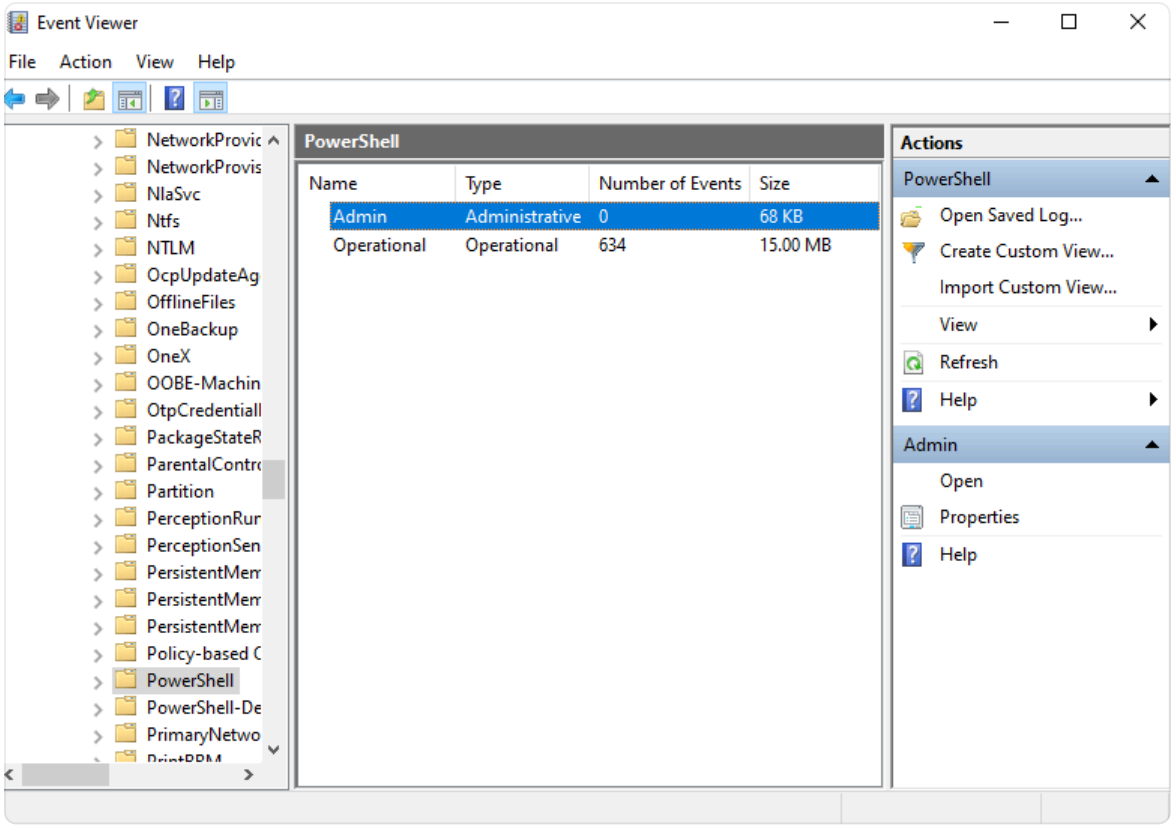


Step 2: Open Event Viewer

1. Press **Win + R**, type `eventvwr.msc` → press Enter.
2. Navigate to:

```
Applications and Services Logs → Microsoft → Windows → PowerShell → Operational
```



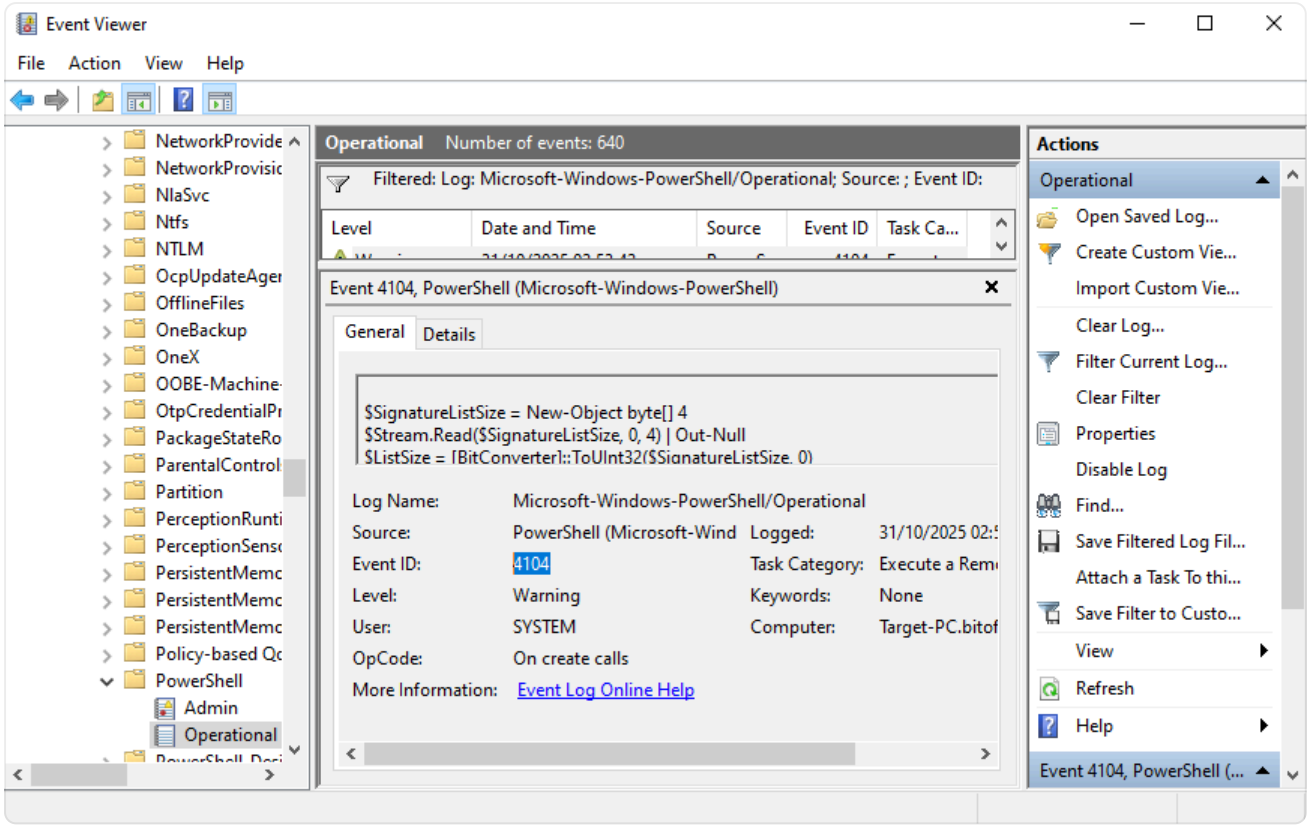


Step 3: Simulate a Suspicious PowerShell Command

1. Open **PowerShell as Administrator**:
   - Search `PowerShell` → Right-click → Run as administrator
2. Run this command:

```
Get-LocalUser | Select-Object Name, Enabled
```

   - What it does: Lists all user accounts on your system.
   - Why it's "suspicious": Attackers often enumerate users after gaining access.



In Conclusion

This lab demonstrated how to simulate an attack using a PowerShell command and then detect it through Windows logs. I learned the importance of enabling logging, understanding log entries, and connecting system events to potential security threats. Logs serve as the main tool for SOC analysts to **observe, detect, and respond** to malicious activity.