Arhum Zahid

05/05/25

CrowdStrike Falcon – Malicious File EDR Detection & Response + PowerShell Automation

Purpose:

The goal was to understand how EDR (Endpoint Detection and Response) tools like **CrowdStrike Falcon** detect, flag, and respond to suspicious or malicious file activity, especially behaviors that mimic early-stage ransomware or unauthorized file modification.

- Detect the malicious activity.
- Visualize the process tree.
- Understand how EDRs trace, block, and respond to potential threats.

Objective:

The objective of this lab was to gain hands-on experience with a leading endpoint detection and response (EDR) solution commonly deployed in enterprise environments. I chose **CrowdStrike Falcon** due to its strong reputation in the industry and its use of AI/ML algorithms for real-time malware detection and antivirus capabilities.

In this lab, I simulated malicious activity by creating custom “infectious” files designed to trigger CrowdStrike’s detection mechanisms. Once flagged by the Falcon sensor, I conducted an in-depth analysis to better understand how the EDR correlates events, classifies threats, and responds to suspicious behavior.

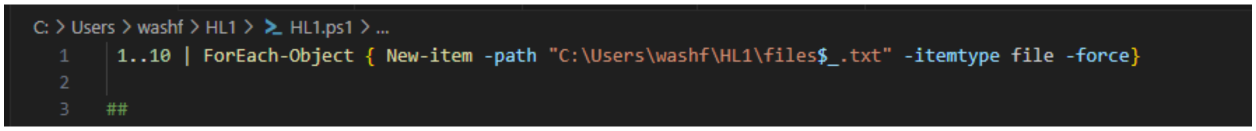
Additionally, I developed a PowerShell script that mimicked malicious behavior. The script was successfully identified and quarantined by the CrowdStrike agent, providing a practical demonstration of how Falcon protects endpoints against real-world threats. This experience not only strengthened my familiarity with EDR tools but also sharpened my skills in threat simulation, detection, and incident analysis.

Arhum Zahid

05/05/25

Script Walkthrough:

Figure 1.a: Creates Files - Initial Payload Simulation

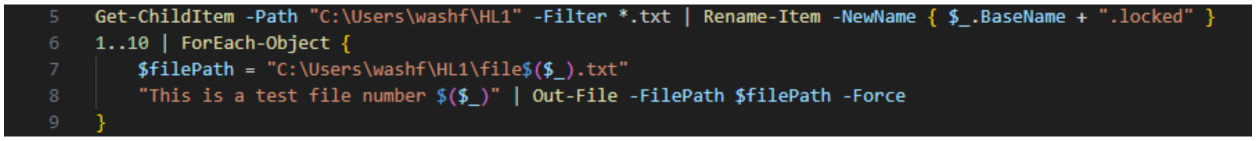


```
C: > Users > washf > HL1 > HL1.ps1 > ...
1..10 | ForEach-Object { New-item -path "C:\Users\washf\HL1\files$_.txt" -itemtype file -force}

##
```

Simulates a dropper creating 10 new files in a specific directory. This mimics an initial payload or malware file dump.

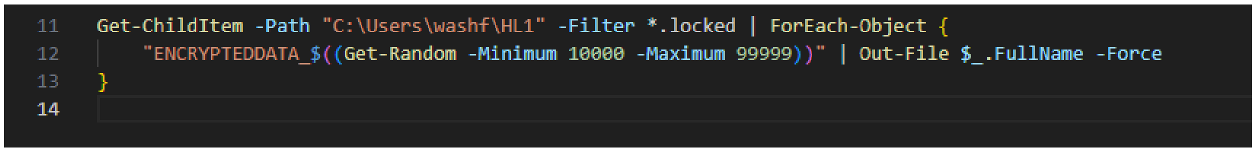
Figure 1.b: Renames Files to Obfuscate / Simulate Encryption (Stage 2 - Ransomware Behavior)



```
Get-ChildItem -Path "C:\Users\washf\HL1" -Filter *.txt | Rename-Item -NewName { $_.BaseName + ".locked" }
1..10 | ForEach-Object {
    $filePath = "C:\Users\washf\HL1\file$($_).txt"
    "This is a test file number $($_)" | Out-File -FilePath $filePath -Force
}
```

Renames all .txt files by appending .locked, a behavior typical of ransomware after encryption. Rewrites original files with new text. Adds another layer of file manipulation — often a red flag for data tampering or ransomware.

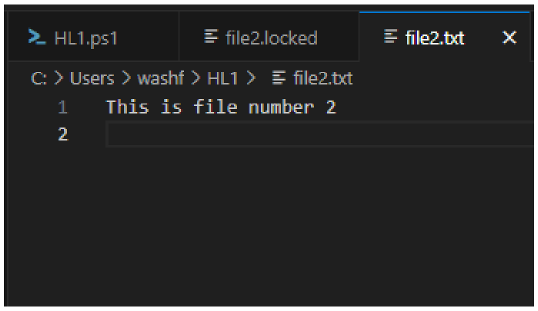
Figure 1.c: Overwrites "Encrypted" Files with Random Data (Stage 4 - Final Ransomware Payload Simulation)



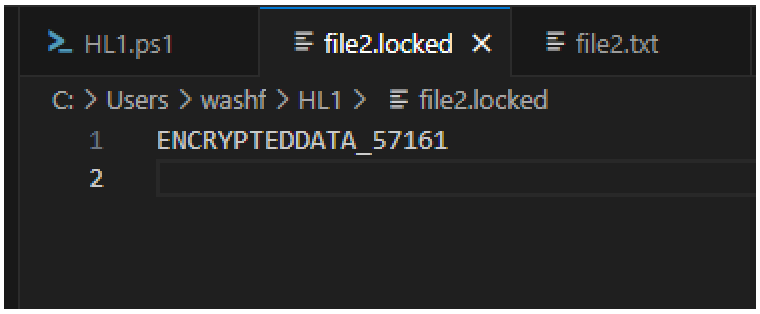
```
Get-ChildItem -Path "C:\Users\washf\HL1" -Filter *.locked | ForEach-Object {
    "ENCRYPTEDDATA_$((Get-Random -Minimum 10000 -Maximum 99999))" | Out-File $_.FullName -Force
}
```

Overwrites all .locked files with random encrypted-like strings. Mimics a final ransomware stage where original content is encrypted and unreadable.

Figure 1.d: Example File Output



```
This is file number 2
```



```
ENCRYPTEDDATA_57161
```

Arhum Zahid

05/05/25

Findings:

Figure 2.a: Process Map

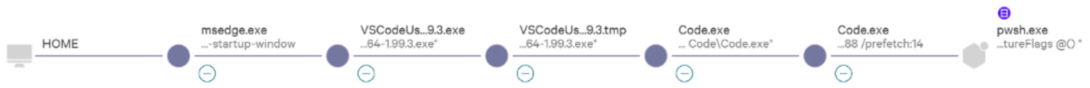
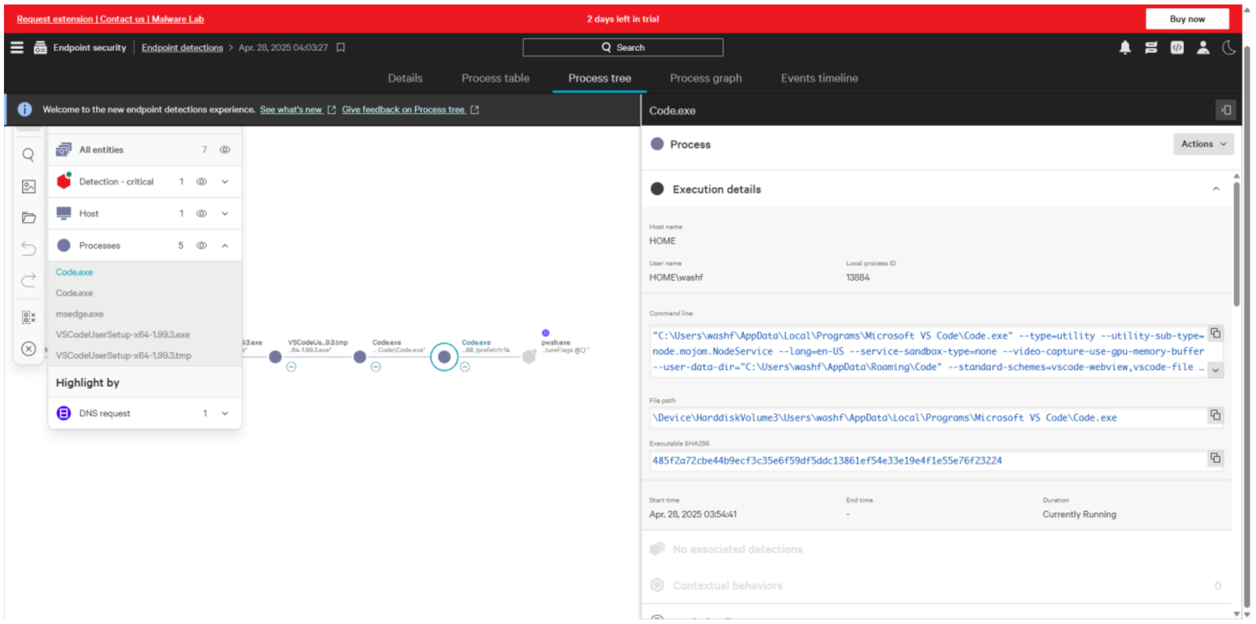


Figure 2.b: Process Map cont'd.



Host name: HOME

User name: HOME\washf

Local process ID: 13884

Command line:

```
"C:\Users\washf\AppData\Local\Programs\Microsoft VS Code\Code.exe" --type=utility --utility-sub-type=
node.mojom.NodeService --lang=en-US --service-sandbox-type=none --video-capture-use-gpu-memory-buffer
--user-data-dir="C:\Users\washf\AppData\Roaming\Code" --standard-schemes=vscode-webview,vscode-file …
```

File path: `\Device\HarddiskVolume3\Users\washf\AppData\Local\Programs\Microsoft VS Code\Code.exe`

Executable SHA256: `485f2a72cbe44b9ecf3c35e6f59df5ddc13861ef54e33e19e4f1e55e76f23224`

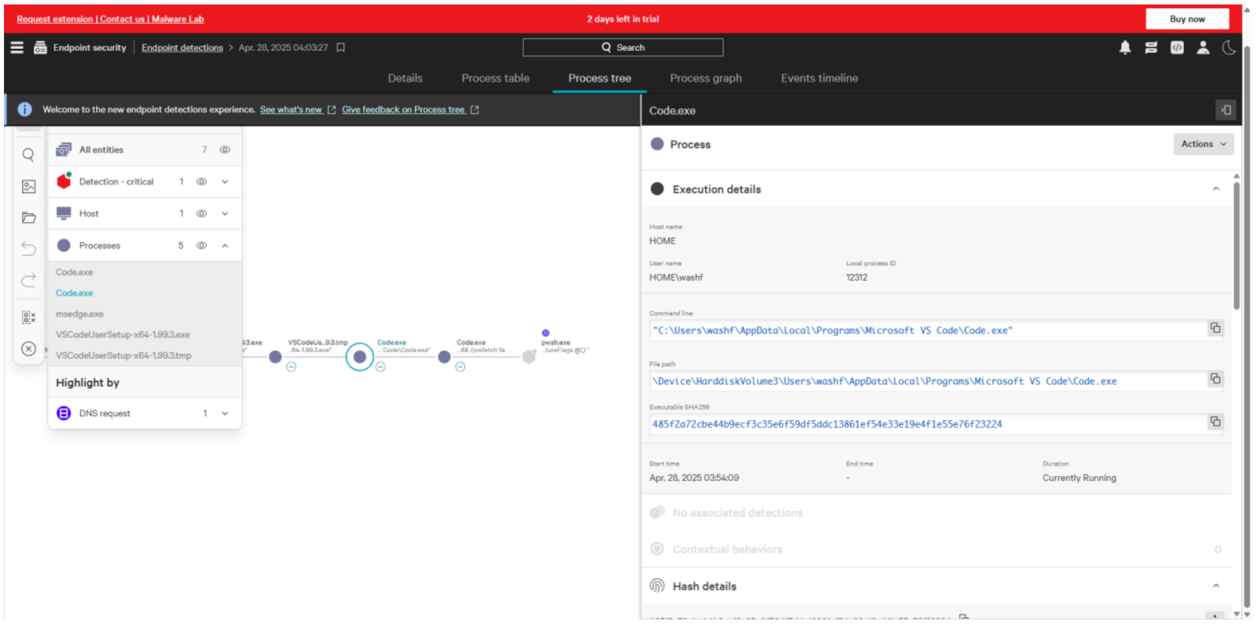
Start time: Apr. 28, 2025 03:54:41

End time: -

Duration: Currently Running

To initiate the environment, I downloaded and executed the Visual Studio Code installer (VSCodeUserSetup-x64-1.99.3.exe) from the official Microsoft source. The executable was placed in the local Downloads directory (C:\Users\washf\Downloads\) to simulate the behavior of a typical user setting up a development workspace. This step served as the origin point for the simulated attack chain. CrowdStrike Falcon recorded this initial action and established a baseline for the process activity, but no suspicious behavior was flagged during installation, as the executable was recognized as legitimate.

Figure 2.c: Process Map cont'd.



Host name: HOME

User name: HOME\washf
Local process ID: 12312

Command line:
```
"C:\Users\washf\AppData\Local\Programs\Microsoft VS Code\Code.exe"
```

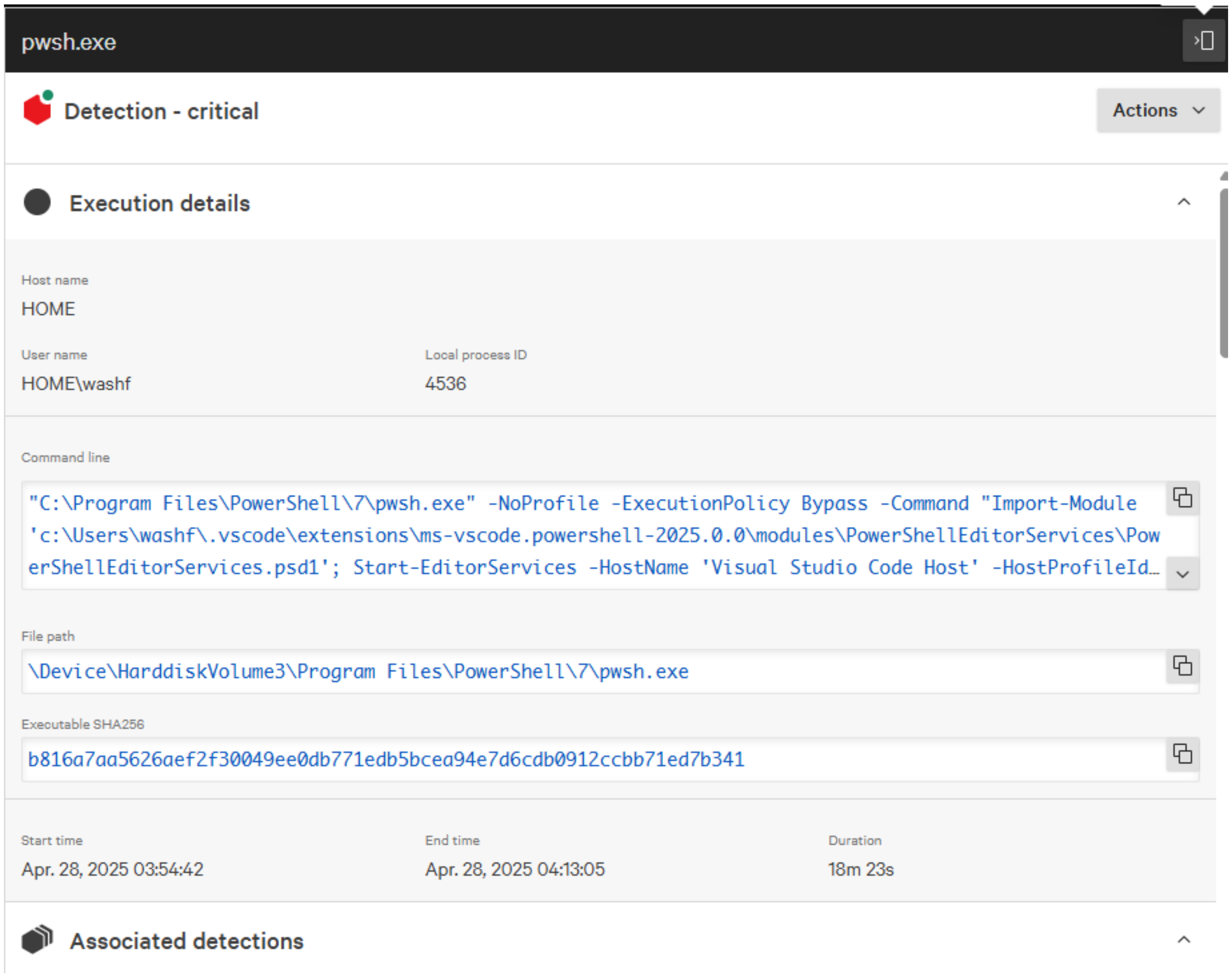
File path:
```
\Device\HarddiskVolume3\Users\washf\AppData\Local\Programs\Microsoft VS Code\Code.exe
```

Executable SHA256:
```
485f2a72cbe44b9ecf3c35e6f59df5ddc13861ef54e33e19e4f1e55e76f23224
```

Start time: Apr. 28, 2025 03:54:09
End time: -
Duration: Currently Running

Once installation was completed, Code.exe—the Visual Studio Code application—was launched from C:\Users\washf\AppData\Local\Programs\Microsoft VS Code\. The application launched with its standard runtime arguments and began functioning as an IDE. This step marked the transition from benign setup activity to the environment where a potentially malicious script would be executed. Falcon recognized the execution of VS Code as a known and trusted application, allowing it to run without intervention or alerts.

Figure 2.d: Process Map cont'd.



pwsh.exe

Detection - critical

Actions

Execution details

Host name
HOME

User name
HOME\washf

Local process ID
4536

Command line

```
"C:\Program Files\PowerShell\7\pwsh.exe" -NoProfile -ExecutionPolicy Bypass -Command "Import-Module
'c:\Users\washf\.vscode\extensions\ms-vscode.powershell-2025.0.0\modules\PowerShellEditorServices\Pow
erShellEditorServices.psd1'; Start-EditorServices -HostName 'Visual Studio Code Host' -HostProfileId…
```

File path

```
\Device\HarddiskVolume3\Program Files\PowerShell\7\pwsh.exe
```

Executable SHA256

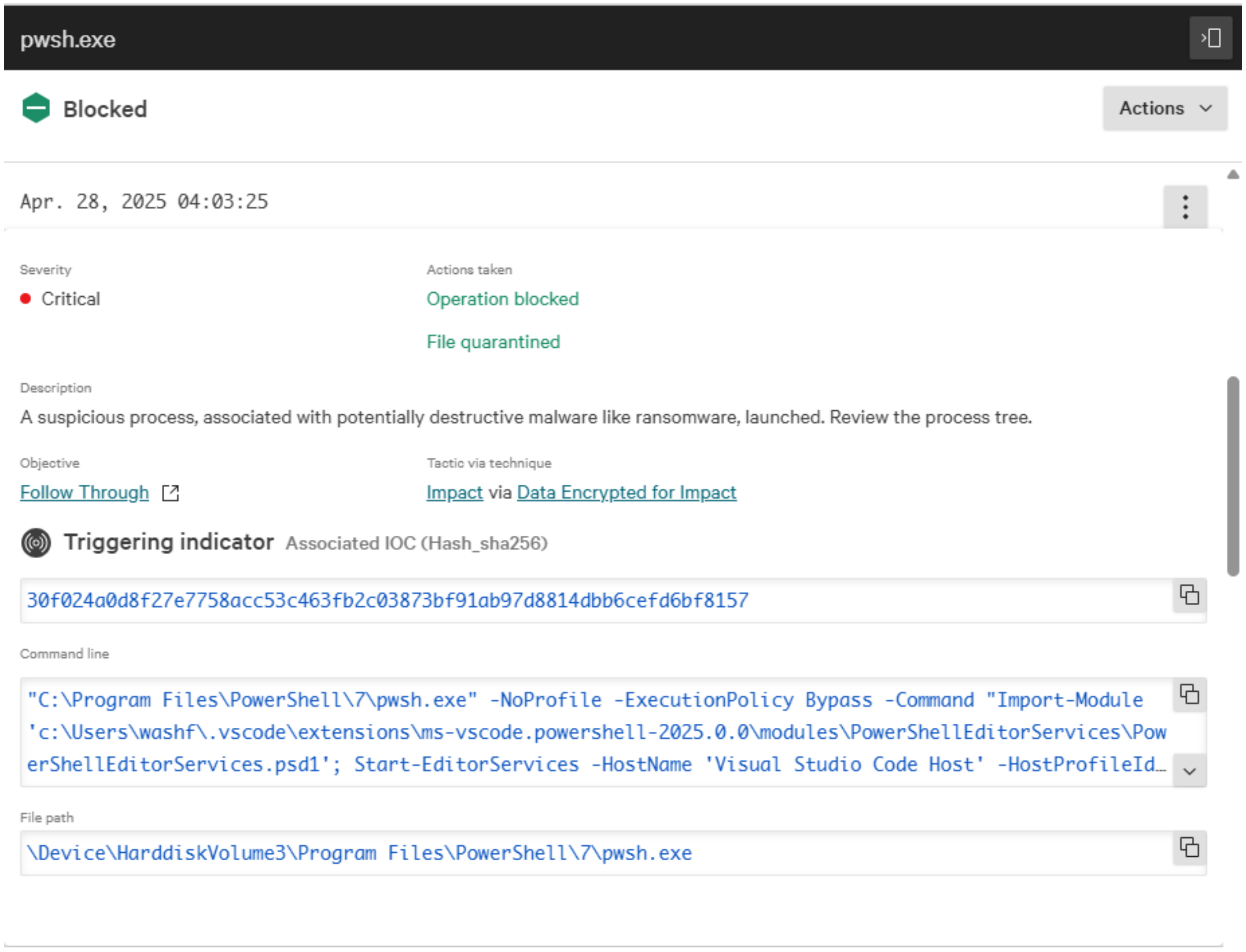
```
b816a7aa5626aef2f30049ee0db771edb5bcea94e7d6cdb0912ccbb71ed7b341
```

Start time	End time	Duration
Apr. 28, 2025 03:54:42	Apr. 28, 2025 04:13:05	18m 23s

Associated detections

The second image presents CrowdStrike Falcon's response once the suspicious PowerShell behavior was detected. The activity was classified as **"Detection - Critical"**, and the automated response included **blocking the operation** and **quarantining the file**. The description confirms that Falcon flagged this process as exhibiting ransomware-like behavior — specifically aligned with **MITRE ATT&CK tactic T1486 (Impact via Data Encrypted for Impact)**. The IOC (Indicator of Compromise) associated with the behavior was the hash 30f024a0d8f27..., further confirming the system's correlation of the behavior with known encryption-based threats. The command line and file path once again validate that the launch originated from PowerShell, reinforcing the severity of this detection.

Figure 2.e: Process Map cont'd.



pwsh.exe

Blocked

Actions

Apr. 28, 2025 04:03:25

Severity	Actions taken
● Critical	Operation blocked
	File quarantined

Description

A suspicious process, associated with potentially destructive malware like ransomware, launched. Review the process tree.

Objective	Tactic via technique
Follow Through	Impact via Data Encrypted for Impact

Triggering indicator Associated IOC (Hash_sha256)

```
30f024a0d8f27e7758acc53c463fb2c03873bf91ab97d8814dbb6cefd6bf8157
```

Command line

```
"C:\Program Files\PowerShell\7\pwsh.exe" -NoProfile -ExecutionPolicy Bypass -Command "Import-Module
'c:\Users\washf\.vscode\extensions\ms-vscode.powershell-2025.0.0\modules\PowerShellEditorServices\Pow
erShellEditorServices.psd1'; Start-EditorServices -HostName 'Visual Studio Code Host' -HostProfileId…
```

File path

```
\Device\HarddiskVolume3\Program Files\PowerShell\7\pwsh.exe
```

This screenshot captures the detailed execution metadata of the pwsh.exe process. It shows that the process was executed on host HOME by the user washf with process ID 4536. The command line used PowerShell 7 Core with the -NoProfile and -ExecutionPolicy Bypass flags — common in evasive or post-exploitation scenarios. The command attempts to import a VS Code-related PowerShell module (PowerShellEditorServices.psd1) and initiate a script session. This behavior, while legitimate in developer environments, is often abused to mask malicious scripting. The file path traces back to the standard PowerShell installation directory, and a unique SHA256 hash is assigned to the executable for tracking integrity and reputation in the EDR backend.

Arhum Zahid

05/05/25

Conclusion:

This lab provided a comprehensive end-to-end walkthrough of how a simulated ransomware-like script is traced, flagged, and contained by an enterprise-grade EDR platform CrowdStrike Falcon. By initiating a development environment using Visual Studio Code and then escalating to the execution of a PowerShell script mimicking malicious encryption behavior, I was able to observe Falcon's detection logic in real time. The simulation successfully triggered a critical detection under MITRE technique T1486, demonstrating Falcon's ability to autonomously block and quarantine threats based on behavioral indicators. This exercise reinforced key cybersecurity concepts such as process tree analysis, script-based threat simulation, command-line obfuscation, and IOC correlation. It also validated the importance of endpoint visibility and response automation in detecting early-stage threats. Overall, the lab not only strengthened my technical fluency in EDR environments but also deepened my practical understanding of threat detection, mitigation strategies, and post-exploitation forensics.

References:

https://attack.mitre.org/techniques/T1486/

https://www.crowdstrike.com/en-us/resources/videos/falcon-platform-demo/

Recoverable Signature

X Arhum Zahid

Arhum Zahid
Cyber Student
Signed by: d40a3175-63f7-455b-a4c9-93d4184fff4e