

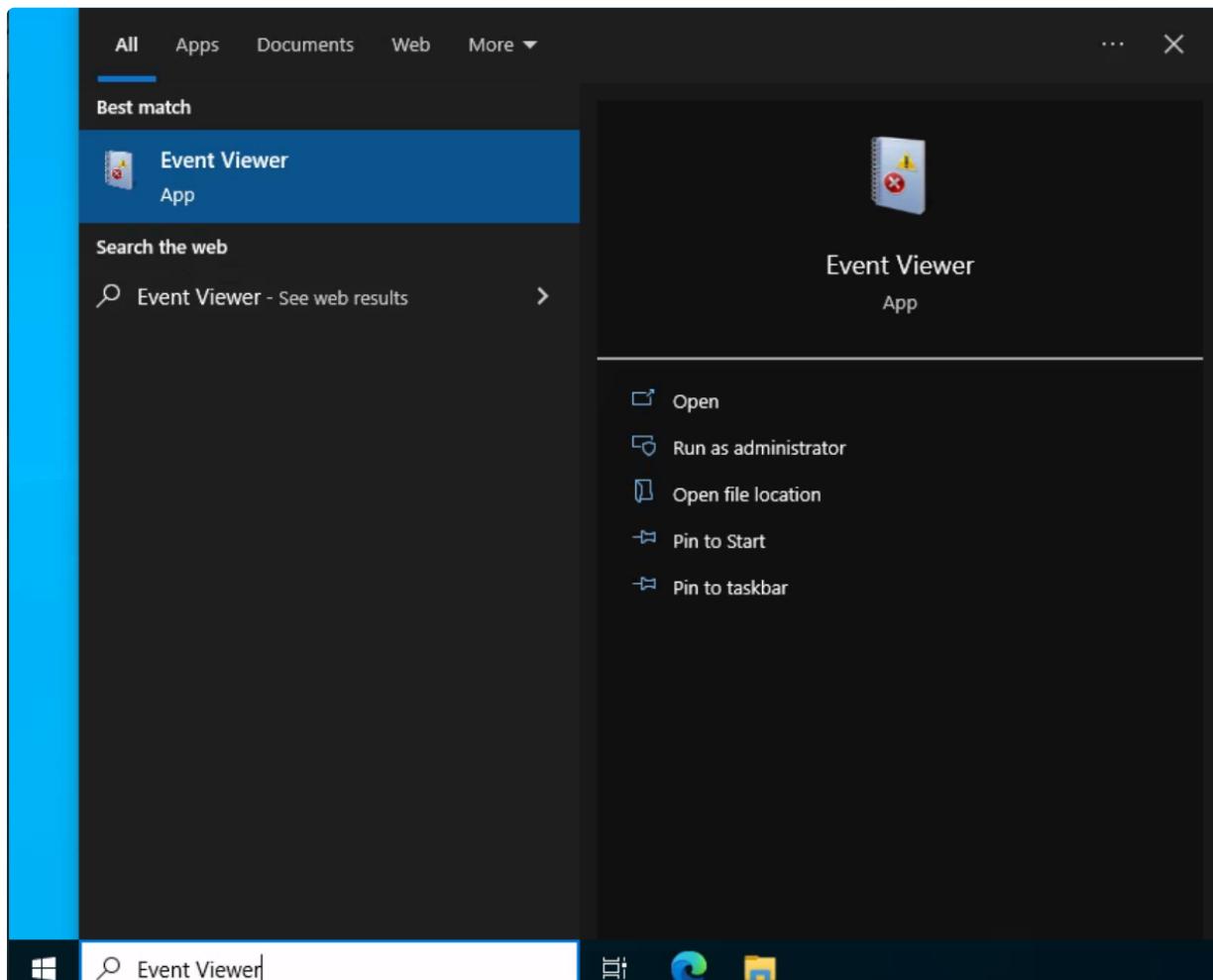
Windows Event Logs & Finding Evil

Introduction

Windows Event Logs

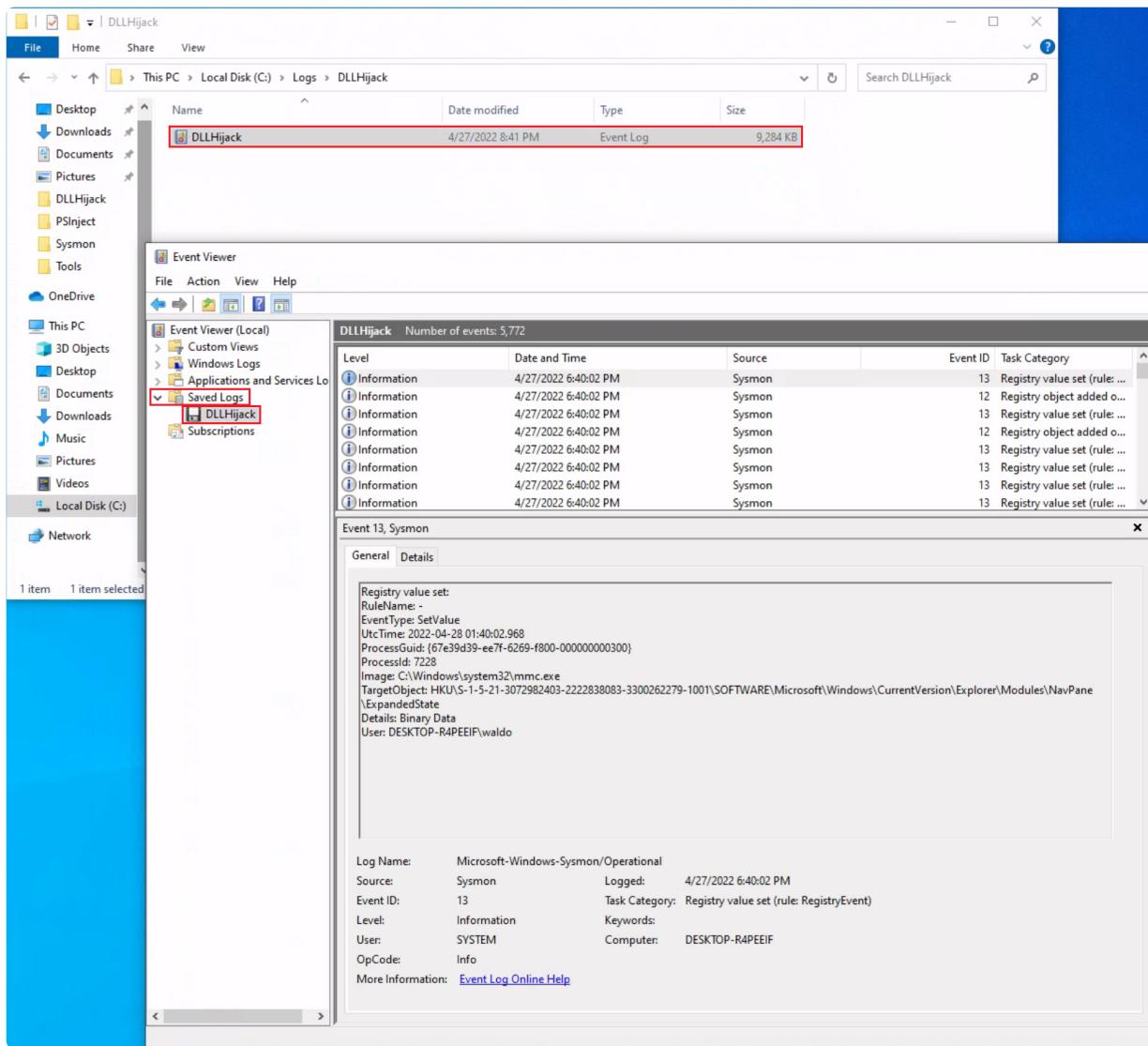
Windows Event Logging Basics

- Windows Event Logs are an intrinsic part of the Windows Operating System, storing logs from different components of the system including the system itself, applications running on it, ETW providers, services, and others.
- Windows event logging offers comprehensive logging capabilities for application errors, security events, and diagnostic information. As cybersecurity professionals, we leverage these logs extensively for analysis and intrusion detection.
- The logs are categorized into different event logs, such as "Application", "System", "Security", and others, to organize events based on their source or purpose.
- Event logs can be accessed using the Event Viewer application or programmatically using APIs such as the Windows Event Log API.
- Accessing the Windows Event Viewer as an administrative user allows us to explore the various logs available



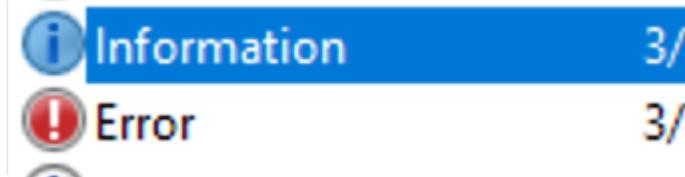
Windows Logs				
Name	Type	Number of Events	Size	
Application	Administrative	1,773	4.07 MB	
Security	Administrative	21,608	20.00 MB	
Setup	Operational	6	68 KB	
System	Administrative	1,026	1.07 MB	
Forwarded Events	Operational	0	0 Bytes	

- The default Windows event logs consist of `Application`, `Security`, `Setup`, `System`, and `Forwarded Events`.
 - While the first four logs cover application errors, security events, system setup activities, and general system information, the "Forwarded Events" section is unique, showcasing event log data forwarded from other machines.
 - This central logging feature proves valuable for system administrators who desire a consolidated view. In our current analysis, we focus on event logs from a single machine.
- It should be noted, that the Windows Event Viewer has the ability to open and display previously saved `.evtx` files, which can be then found in the "Saved Logs" section.



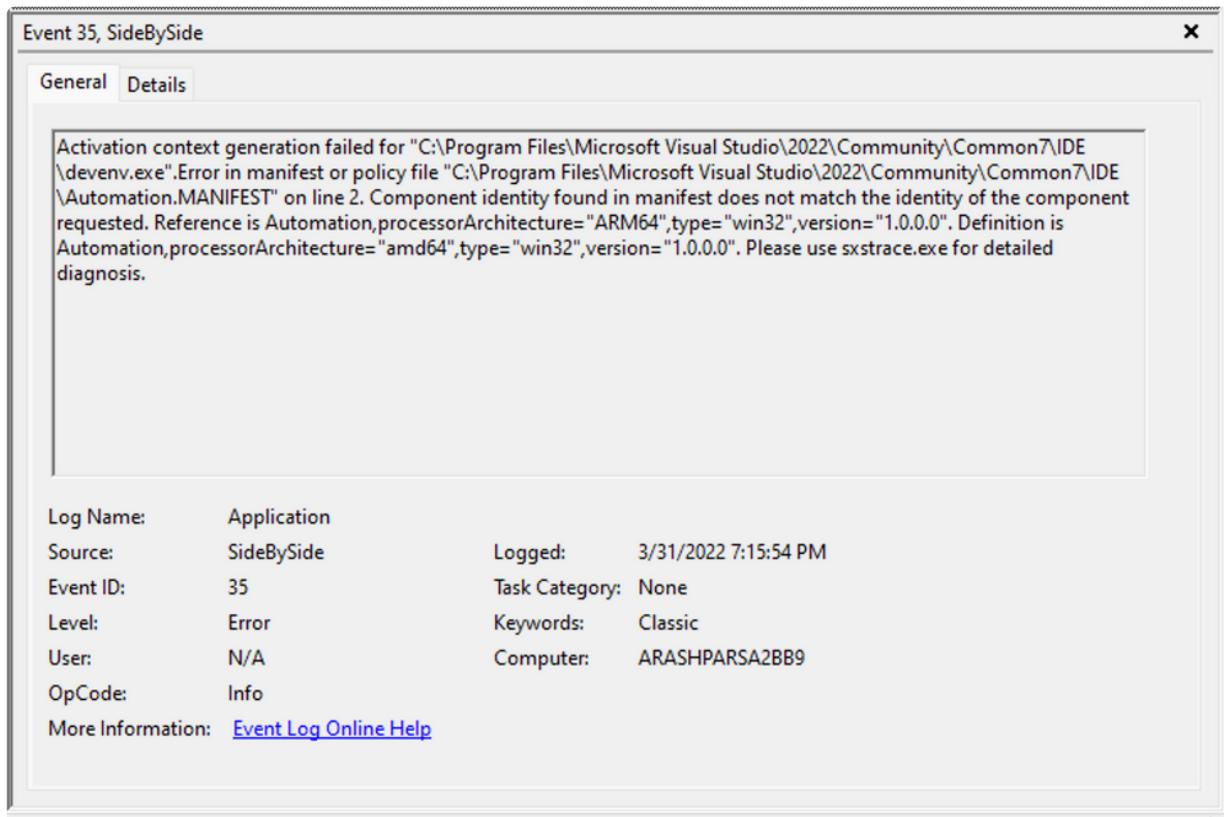
The Anatomy of an Event Log

- When examining Application logs, we encounter two distinct levels of events: information and error.



- Information events provide general usage details about the application, such as its start or stop events.
 - Conversely, error events highlight specific errors and often offer detailed insights into

the encountered issues.



- Each entry in the Windows Event Log is an "Event" and contains the following primary components:
 1. **Log Name** : The name of the event log (e.g., Application, System, Security, etc.).
 2. **Source** : The software that logged the event.
 3. **Event ID** : A unique identifier for the event.
 4. **Task Category** : This often contains a value or name that can help us understand the purpose or use of the event.
 5. **Level** : The severity of the event (Information, Warning, Error, Critical, and Verbose).
 6. **Keywords** : Keywords are flags that allow us to categorize events in ways beyond the other classification options. These are generally broad categories, such as "Audit Success" or "Audit Failure" in the Security log.
 7. **User** : The user account that was logged on when the event occurred.
 8. **OpCode** : This field can identify the specific operation that the event reports.
 9. **Logged** : The date and time when the event was logged.
 10. **Computer** : The name of the computer where the event occurred.
 11. **XML Data** : All the above information is also included in an XML format along with additional event data.
- The **Keywords** field is particularly useful when filtering event logs for specific types of events.

- It can significantly enhance the precision of search queries by allowing us to specify events of interest, thus making log management more efficient and effective.
- Taking a closer look at the event log above, we observe several crucial elements.
 - The `Event ID` in the top left corner serves as a unique identifier, which can be further researched on Microsoft's website to gather additional information.
 - The "SideBySide" label next to the event ID represents the event source. Below, we find the general error description, often containing rich details.
 - By clicking on the details, we can further analyze the event's impact using XML or a well-formatted view.
- The `Keywords` field is particularly useful when filtering event logs for specific types of events.
 - It can significantly enhance the precision of search queries by allowing us to specify events of interest, thus making log management more efficient and effective.
- Taking a closer look at the event log above, we observe several crucial elements. The `Event ID` in the top left corner serves as a unique identifier, which can be further researched on Microsoft's website to gather additional information.
 - The "SideBySide" label next to the event ID represents the event source.
 - Below, we find the general error description, often containing rich details.
 - By clicking on the details, we can further analyze the event's impact using XML or a well-formatted view.
- Additionally, we can extract supplementary information from the event log, such as the process ID where the error occurred, enabling more precise analysis.

- TimeCreated	
	[SystemTime] 2022-03-31T23:15:54.1031089Z
EventRecordID	1773
Correlation	
- Execution	
[ProcessID]	636
[ThreadID]	0
Channel	Application
Computer	ARASHPARSA2BB9
Security	

- Switching our focus to security logs, let's consider event ID 4624, a commonly occurring event (detailed at [https://docs.microsoft.com/en-us/windows/security/threat-](https://docs.microsoft.com/en-us/windows/security/threat)

protection/auditing/event-4624).

The screenshot shows the Windows Event Viewer interface with the title "Event 4624, Microsoft Windows security auditing". There are two tabs: "General" (selected) and "Details".

General Tab:

- An account was successfully logged on.
- Subject:**
 - Security ID: SYSTEM
 - Account Name: ARASHPARSA2BB9\$
 - Account Domain: WORKGROUP
 - Logon ID: 0x3E7
- Logon Information:**
 - Logon Type: 5
 - Restricted Admin Mode: -

Details Tab:

Log Name:	Security		
Source:	Microsoft Windows security	Logged:	3/31/2022 7:23:09 PM
Event ID:	4624	Task Category:	Logon
Level:	Information	Keywords:	Audit Success
User:	N/A	Computer:	ARASHPARSA2BB9
OpCode:	Info		

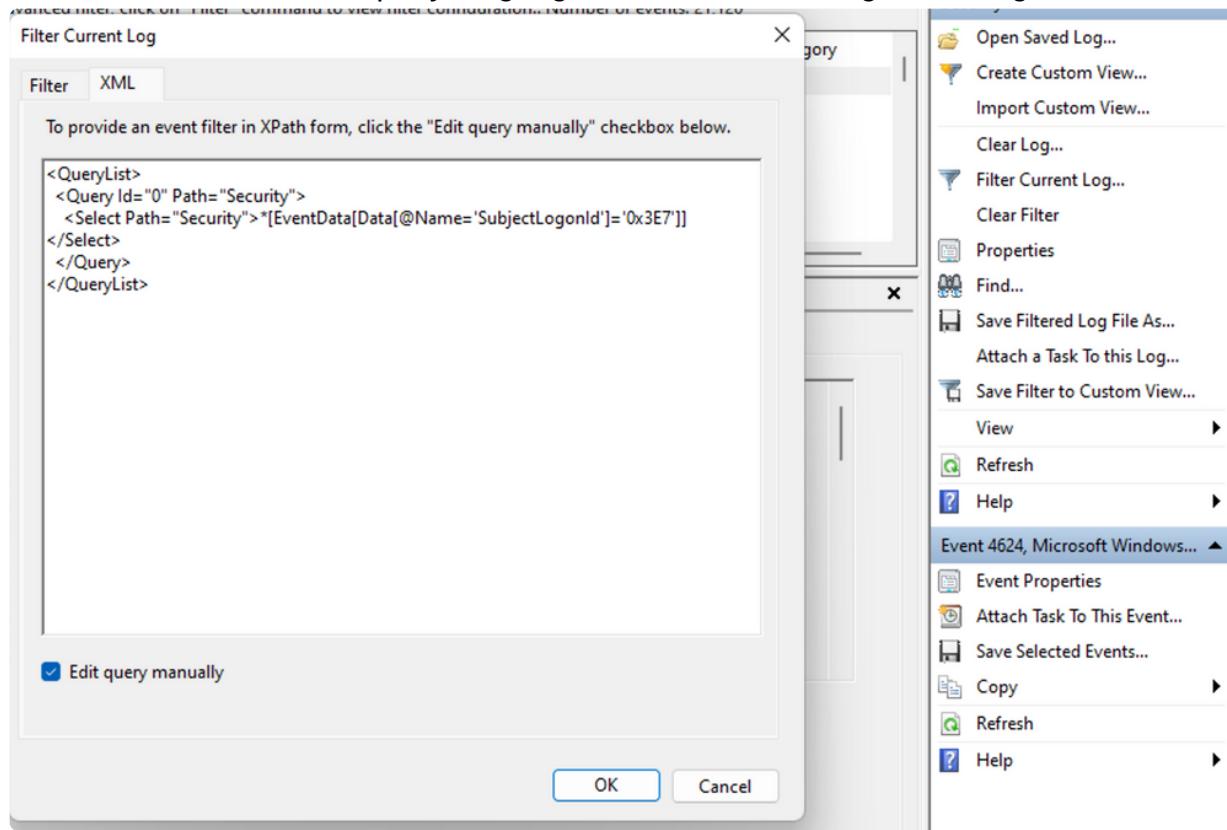
More Information: [Event Log Online Help](#)

- According to Microsoft's documentation, this event signifies the creation of a logon session on the destination machine, originating from the accessed computer where the session was established.
 - Within this log, we find crucial details, including the "Logon ID", which allows us to correlate this logon with other events sharing the same "Logon ID".
 - Another important detail is the "Logon Type", indicating the type of logon.
 - In this case, it specifies a Service logon type, suggesting that "SYSTEM" initiated a new service.
 - However, further investigation is required to determine the specific service involved, utilizing correlation techniques with additional data like the "Logon ID".

Leveraging Custom XML Queries

- To streamline our analysis, we can create custom XML queries to identify related events using the "Logon ID" as a starting point.
 - By navigating to "Filter Current Log" -> "XML" -> "Edit Query Manually," we gain

access to a custom XML query language that enables more granular log searches.



```
<Select Path="Security">*
[EventData[Data[@Name='SubjectLogonId']='0x3E7']]</Select>
```

- In the example query, we focus on events containing the "SubjectLogonId" field with a value of "0x3E7".
 - The selection of this value stems from the need to correlate events associated with a

specific "Logon ID" and understand the relevant details within those events.

The screenshot shows the Windows Event Viewer interface for Event 4624, Microsoft Windows security auditing. The event details are displayed in Friendly View. The System section contains the following key information:

- SubjectUserId**: S-1-5-18
- SubjectUserName**: ARASHPARSA2BB9\$
- SubjectDomainName**: WORKGROUP
- SubjectLogonId**: 0x3e7
- TargetUserId**: S-1-5-18
- TargetUserName**: SYSTEM
- TargetDomainName**: NT AUTHORITY
- TargetLogonId**: 0x3e7
- LogonType**: 5
- LogonProcessName**: Advapi
- AuthenticationPackageName**: Negotiate
- WorkstationName**: -
- LogonGuid**: {00000000-0000-0000-0000-000000000000}

- It is worth noting that if assistance is required in crafting the query, automatic filters can be enabled, allowing exploration of their impact on the XML representation.
 - For further guidance, Microsoft offers informative articles on [advanced XML filtering in the Windows Event Viewer](#).
- By constructing such queries, we can narrow down our focus to the account responsible for initiating the service and eliminate unnecessary details.
 - This approach helps unveil a clearer picture of recent logon activities associated with the specified Logon ID.
 - However, even with this refinement, the amount of data remains significant.
- Delving into the log details progressively reveals a narrative.
 - For instance, the analysis begins with [Event ID 4907](#), which signifies an audit policy

change.

Security Number of events: 21,186 (!) New events available

Filtered: Advanced filter, click on "Filter" command to view filter configuration.. Number of events: 21,155

Keywords	Date and Time	Source	Event ID	Task Category
Audit Success	3/31/2022 7:22:26 PM	Microsoft Window...	4907	Audit Policy Change
Audit Success	3/31/2022 7:22:26 PM	Microsoft Window...	4907	Audit Policy Change
Audit Success	3/31/2022 7:22:26 PM	Microsoft Window...	4907	Audit Policy Change
Audit Success	3/31/2022 7:22:05 PM	Microsoft Window...	4624	Login

Event 4907, Microsoft Windows security auditing.

General Details

Auditing settings on object were changed.

Subject:

Security ID:	SYSTEM
Account Name:	ARASHPARSA2BB9\$
Account Domain:	WORKGROUP
Logon ID:	0x3E7

Object:

Object Server:	Security
Object Type:	File
Object Name:	C:\\$WINDOWS.~BT\NewOS\Windows\System32\Recovery\47b6b968-2f36-4cef-859e-555019b02fc8\Windows\WinSxS\arm64_microsoft-windows-b!ore-bootmanager-efi_31bf3856ad364e35_10.0.22543.1000_none_c19343c788cc21ea\bootmgfw.efi
Handle ID:	0xb520

Process Information:

Log Name:	Security		
Source:	Microsoft Windows security	Logged:	3/31/2022 7:22:26 PM
Event ID:	4907	Task Category:	Audit Policy Change
Level:	Information	Keywords:	Audit Success
User:	N/A	Computer:	ARASHPARSA2BB9
OpCode:	Info		

More Information: [Event Log Online Help](#)

- Within the [event description](#), we find valuable insights, such as "This event generates when the SACL of an object (for example, a registry key or file) was changed."
- In case unfamiliar with SACL, referring to the provided link (<https://docs.microsoft.com/en-us/windows/win32/secauthz/access-control-lists>) sheds light on access control lists (ACLs).
 - The "S" in SACL denotes a system access control list, which enables administrators to log access attempts to secure objects.
 - Each Access Control Entry (ACE) within a SACL specifies the types of access attempts by a designated trustee that trigger record generation in the security event log.
 - ACEs in a SACL can generate audit records upon failed, successful, or both types of access attempts. For more information about SACLs, see [Audit Generation](#) and [SACL Access Right](#).

- Based on this information, it becomes apparent that the permissions of a file were altered to modify the logging or auditing of access attempts.
- Further exploration of the event details reveals additional intriguing aspects.

Event 4907, Microsoft Windows security auditing.

EventData

SubjectUserSid	S-1-5-18
SubjectUserName	ARASHPARSA2BB9\$
SubjectDomainName	WORKGROUP
SubjectLogonId	0x3e7
ObjectServer	Security
ObjectType	File
ObjectName	C:\\$WINDOWS.~BT\NewOS\Windows\System32\Recovery\47b6b94cef-859e-555019b02fc8\Windows\WinSxS\arm64_microsoft-windows-bootmanager-efi_31bf3856ad364e35_10.0.22543.1000_none_c19343c788cc21ea
HandleId	0xb520
OldSd	
NewSd	S:ARAI(AU;SAFA;DCLCRPCRSWDWO::;WD)
ProcessId	0x1af8
ProcessName	C:\\$WINDOWS.~BT\Sources\SetupHost.exe

- For example, the process responsible for the change is identified as "SetupHost.exe", indicating a potential setup process (although it's worth noting that malware can sometimes masquerade under legitimate names).
 - The object name impacted appears to be the "bootmanager", and we can examine the new and old security descriptors ("NewSd" and "OldSd") to identify the changes.
 - Understanding the meaning of each field in the security descriptor can be accomplished through references such as the article [ACE Strings](#) and [Understanding SDDL Syntax](#).
- From the observed events, we can infer that a setup process occurred, involving the creation of a new file and the initial configuration of security permissions for auditing purposes.
 - Subsequently, we encounter the logon event, followed by a "special logon" event.

4624 Logon

4672 Special Logon

- Analyzing the special logon event, we gain insights into token permissions granted to the user upon a successful logon.

The screenshot shows the Windows Event Viewer interface. At the top, there's a header bar with the title 'Audit Success' and the date/time '3/31/2022 7:23:05 PM'. To the right of the title are fields for 'Microsoft Window...', '4672 Special Logon', and '1 Logon'. Below the header, a message box displays the text 'Event 4672, Microsoft Windows security auditing.' with an 'X' button in the top right corner. The main content area has tabs for 'General' and 'Details'. The 'General' tab is selected, showing the following information:

Security ID:	SYSTEM
Account Name:	SYSTEM
Account Domain:	NT AUTHORITY
Logon ID:	0x3E7
Privileges:	SeAssignPrimaryTokenPrivilege SeTcbPrivilege SeSecurityPrivilege SeTakeOwnershipPrivilege SeLoadDriverPrivilege SeBackupPrivilege SeRestorePrivilege SeDebugPrivilege SeAuditPrivilege SeSystemEnvironmentPrivilege SeImpersonatePrivilege SeDelegateSessionUserImpersonatePrivilege
Log Name:	Security
Source:	Microsoft Windows security
Event ID:	4672
Level:	Information
User:	N/A
OpCode:	Info
More Information:	Event Log Online Help

- A comprehensive list of privileges can be found in the documentation on [privilege constants](#).
 - For instance, the "SeDebugPrivilege" privilege indicates that the user possesses the ability to tamper with memory that does not belong to them.

Useful Windows Event Logs

- Find below an indicative (non-exhaustive) list of useful Windows event logs.

1. Windows System Logs

- [Event ID 1074 \(System Shutdown/Restart\)](#) : This event log indicates when and why the system was shut down or restarted.
 - By monitoring these events, you can determine if there are unexpected shutdowns or restarts, potentially revealing malicious activity such as malware infection or unauthorized user access.

- Event ID 6005 (The Event log service was started) : This event log marks the time when the Event Log Service was started.
 - This is an important record, as it can signify a system boot-up, providing a starting point for investigating system performance or potential security incidents around that period. It can also be used to detect unauthorized system reboots.
- Event ID 6006 (The Event log service was stopped) : This event log signifies the moment when the Event Log Service was stopped. It is typically seen when the system is shutting down.
 - Abnormal or unexpected occurrences of this event could point to intentional service disruption for covering illicit activities.
- Event ID 6013 (Windows uptime) : This event occurs once a day and shows the uptime of the system in seconds.
 - A shorter than expected uptime could mean the system has been rebooted, which could signify a potential intrusion or unauthorized activities on the system.
- Event ID 7040 (Service status change) : This event indicates a change in service startup type, which could be from manual to automatic or vice versa.
 - If a crucial service's startup type is changed, it could be a sign of system tampering.

2. Windows Security Logs

- Event ID 1102 (The audit log was cleared) : Clearing the audit log is often a sign of an attempt to remove evidence of an intrusion or malicious activity.
- Event ID 1116 (Antivirus malware detection) : This event is particularly important because it logs when Defender detects a malware.
 - A surge in these events could indicate a targeted attack or widespread malware infection.
- Event ID 1118 (Antivirus remediation activity has started) : This event signifies that Defender has begun the process of removing or quarantining detected malware.
 - It's important to monitor these events to ensure that remediation activities are successful.
- Event ID 1119 (Antivirus remediation activity has succeeded) : This event signifies that the remediation process for detected malware has been successful.
 - Regular monitoring of these events will help ensure that identified threats are effectively neutralized.
- Event ID 1120 (Antivirus remediation activity has failed) : This event is the counterpart to 1119 and indicates that the remediation process has failed.

- These events should be closely monitored and addressed immediately to ensure threats are effectively neutralized.
- Event ID 4624 (Successful Logon) : This event records successful logon events.
 - This information is vital for establishing normal user behavior. Abnormal behavior, such as logon attempts at odd hours or from different locations, could signify a potential security threat.
- Event ID 4625 (Failed Logon) : This event logs failed logon attempts. Multiple failed logon attempts could signify a brute-force attack in progress.
- Event ID 4648 (A logon was attempted using explicit credentials) : This event is triggered when a user logs on with explicit credentials to run a program.
 - Anomalies in these logon events could indicate lateral movement within a network, which is a common technique used by attackers.
- Event ID 4656 (A handle to an object was requested) : This event is triggered when a handle to an object (like a file, registry key, or process) is requested.
 - This can be a useful event for detecting attempts to access sensitive resources.
- Event ID 4672 (Special Privileges Assigned to a New Logon) : This event is logged whenever an account logs on with super user privileges.
 - Tracking these events helps to ensure that super user privileges are not being abused or used maliciously.
- Event ID 4698 (A scheduled task was created) : This event is triggered when a scheduled task is created.
 - Monitoring this event can help you detect persistence mechanisms, as attackers often use scheduled tasks to maintain access and run malicious code.
- Event ID 4700 & Event ID 4701 (A scheduled task was enabled/disabled) : This records the enabling or disabling of a scheduled task. Scheduled tasks are often manipulated by attackers for persistence or to run malicious code, thus these logs can provide valuable insight into suspicious activities.
- Event ID 4702 (A scheduled task was updated) : Similar to 4698, this event is triggered when a scheduled task is updated.
 - Monitoring these updates can help detect changes that may signify malicious intent.
- Event ID 4719 (System audit policy was changed) : This event records changes to the audit policy on a computer.
 - It could be a sign that someone is trying to cover their tracks by turning off auditing or changing what events get audited.

- Event ID 4738 (A user account was changed) : This event records any changes made to user accounts, including changes to privileges, group memberships, and account settings.
 - Unexpected account changes can be a sign of account takeover or insider threats.
- Event ID 4771 (Kerberos pre-authentication failed) : This event is similar to 4625 (failed logon) but specifically for Kerberos authentication.
 - An unusual amount of these logs could indicate an attacker attempting to brute force your Kerberos service.
- Event ID 4776 (The domain controller attempted to validate the credentials for an account) : This event helps track both successful and failed attempts at credential validation by the domain controller.
 - Multiple failures could suggest a brute-force attack.
- Event ID 5001 (Antivirus real-time protection configuration has changed) : This event indicates that the real-time protection settings of Defender have been modified. Unauthorized changes could indicate an attempt to disable or undermine the functionality of Defender.
- Event ID 5140 (A network share object was accessed) : This event is logged whenever a network share is accessed.
 - This can be critical in identifying unauthorized access to network shares.
- Event ID 5142 (A network share object was added) : This event signifies the creation of a new network share.
 - Unauthorized network shares could be used to exfiltrate data or spread malware across a network.
- Event ID 5145 (A network share object was checked to see whether client can be granted desired access) : This event indicates that someone attempted to access a network share.
 - Frequent checks of this sort might indicate a user or a malware trying to map out the network shares for future exploits.
- Event ID 5157 (The Windows Filtering Platform has blocked a connection) : This is logged when the Windows Filtering Platform blocks a connection attempt.
 - This can be helpful for identifying malicious traffic on your network.
- Event ID 7045 (A service was installed in the system) : A sudden appearance of unknown services might suggest malware installation, as many types of malware install themselves as services.

Analyzing Evil With Sysmon & Event Logs

Overview

- In our pursuit of robust cybersecurity, it is crucial to understand how to identify and analyze malicious events effectively.
 - Building upon our previous exploration of benign events, we will now delve into the realm of malicious activities and discover techniques for detection.

Sysmon Basics

- When investigating malicious events, several event IDs serve as common indicators of compromise.
 - For instance, Event ID 4624 provides insights into new logon events, enabling us to monitor and detect suspicious user access and logon patterns.
 - Similarly, Event ID 4688 furnishes information about newly created processes, aiding the identification of unusual or malicious process launches.
 - To enhance our event log coverage, we can extend the capabilities by incorporating Sysmon, which offers additional event logging capabilities.
- System Monitor (Sysmon) is a Windows system service and device driver that remains resident across system reboots to monitor and log system activity to the Windows event log.
 - Sysmon provides detailed information about process creation, network connections, changes to file creation time, and more.
- Sysmon's primary components include:
 - A Windows service for monitoring system activity.
 - A device driver that assists in capturing the system activity data.
 - An event log to display captured activity data.
- Sysmon's unique capability lies in its ability to log information that typically doesn't appear in the Security Event logs, and this makes it a powerful tool for deep system monitoring and cybersecurity forensic analysis.
- Sysmon categorizes different types of system activity using event IDs, where each ID corresponds to a specific type of event.
 - For example, Event ID 1 corresponds to "Process Creation" events, and Event ID 3 refers to "Network Connection" events.
 - The full list of Sysmon event IDs can be found [here](#).
- For more granular control over what events get logged, Sysmon uses an XML-based configuration file.
 - The configuration file allows you to include or exclude certain types of events based on different attributes like process names, IP addresses, etc.

- We can refer to popular examples of useful Sysmon configuration files:
- For a comprehensive configuration, we can visit: <https://github.com/SwiftOnSecurity/sysmon-config>. <-- We will use this one in this section!
- Another option is: <https://github.com/olafhartong/sysmon-modular>, which provides a modular approach.
- To get started, you can install Sysmon by downloading it from the official Microsoft documentation (<https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon>).
- Once downloaded, open an administrator command prompt and execute the following command to install Sysmon.

```
C:\Tools\Sysmon> sysmon.exe -i -accepteula -h md5,sha256,imphash -l -n
```

```
C:\Users\Waldo\Desktop\Sysmon>sysmon.exe -i -accepteula -h md5,sha256,imphash -l -n

System Monitor v13.33 - System activity monitor
By Mark Russinovich and Thomas Garnier
Copyright (C) 2014-2022 Microsoft Corporation
Using libxml2. libxml2 is Copyright (C) 1998-2012 Daniel Veillard. All Rights Reserved.
Sysinternals - www.sysinternals.com

Sysmon installed.
SysmonDrv installed.
Starting SysmonDrv.
SysmonDrv started.
Starting Sysmon..
Sysmon started.
```

- To utilize a custom Sysmon configuration, execute the following after installing Sysmon.

```
C:\Tools\Sysmon> sysmon.exe -c filename.xml
```

-> **Note:** It should be noted that [Sysmon for Linux](#) also exists.

Detection Example 1: Detecting DLL Hijacking

- In our specific use case, we aim to detect a DLL hijack.
 - The Sysmon event log IDs relevant to DLL hijacks can be found in the Sysmon documentation (<https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon>).

- To detect a DLL hijack, we need to focus on Event Type 7, which corresponds to module load events.
- To achieve this, we need to modify the sysmonconfig-export.xml Sysmon configuration file we downloaded from <https://github.com/SwiftOnSecurity/sysmon-config>.
- By examining the modified configuration, we can observe that the "include" comment signifies events that should be included.

```

<RuleGroup name="" groupRelation="or">
    <ImageLoad onmatch="include">
        <!--NOTE: Using "include" with no rules means nothing in this section will be logged-->
    </ImageLoad>
</RuleGroup>
```

- In the case of detecting DLL hijacks, we change the "include" to "exclude" to ensure that nothing is excluded, allowing us to capture the necessary data.

```

<RuleGroup name="" groupRelation="or">
    <ImageLoad onmatch="exclude">
        <!--NOTE: Using "include" with no rules means nothing in this section will be logged-->
    </ImageLoad>
</RuleGroup>
```

- To utilize the updated Sysmon configuration, execute the following.

```
C:\Tools\Sysmon> sysmon.exe -c sysmonconfig-export.xml
```

```
C:\Users\Waldo\Desktop\Sysmon>Sysmon.exe -c sysmonconfig-export.xml

System Monitor v13.33 - System activity monitor
By Mark Russinovich and Thomas Garnier
Copyright (C) 2014-2022 Microsoft Corporation
Using libxml2. libxml2 is Copyright (C) 1998-2012 Daniel Veillard. All Rights Reserved.
Sysinternals - www.sysinternals.com

Loading configuration file with schema version 4.50
Sysmon schema version: 4.81
Configuration file validated.
Configuration updated.
```

- With the modified Sysmon configuration, we can start observing image load events.
 - To view these events, navigate to the Event Viewer and access "Applications and Services" -> "Microsoft" -> "Windows" -> "Sysmon."
 - A quick check will reveal the presence of the targeted event ID.

- Let's now see how a Sysmon event ID 7 looks like.

```
Image loaded:
RuleName: -
UtcTime: 2022-04-07 22:21:35.372
ProcessGuid:{2c4ac1bf-63dd-624f-640b-00000000d00}
ProcessId: 8080
Image: C:\Windows\System32\mmc.exe
ImageLoaded: C:\Windows\System32\psapi.dll
FileVersion: 10.0.18362.1 (WinBuild.160101.0800)
Description: Process Status Helper
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: PSAPI
Hashes: MD5=89D92079F45D2F2539BCD1EEF73A701E,SHA256=6DAE0B5BAC5B2C34FD313B51AC793B6F0C270DA01474E4D1016B119FC1F9CE8F,IMPHASH=A19426362F5443C7159B76FBEAFD171F
Signed: true
Signature: Microsoft Windows
SignatureStatus: Valid
User: DESKTOP-N33HELB\Waldo
```

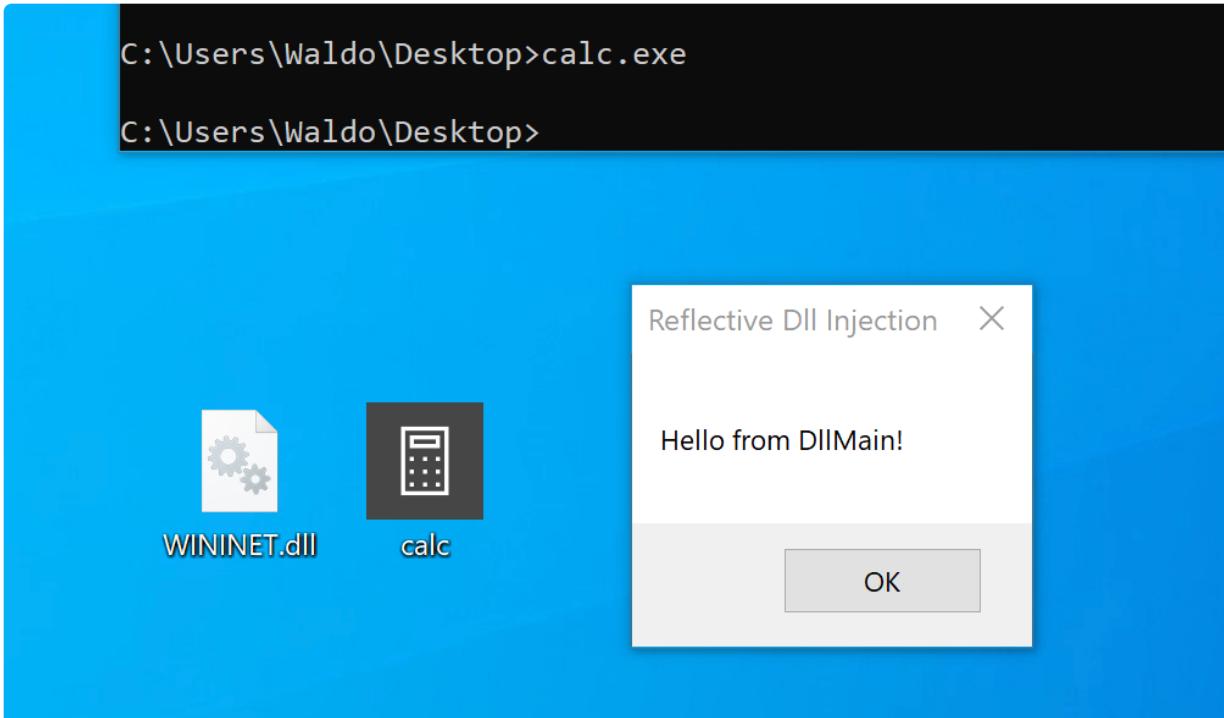
- The event log contains the DLL's signing status (in this case, it is Microsoft-signed), the process or image responsible for loading the DLL, and the specific DLL that was loaded.
 - In our example, we observe that "MMC.exe" loaded "psapi.dll", which is also Microsoft-signed. Both files are located in the System32 directory.
- Now, let's proceed with building a detection mechanism.
 - To gain more insights into DLL hijacks, conducting research is paramount.
 - We stumble upon an informative [blog post](#) that provides an exhaustive list of various DLL hijack techniques.
 - For the purpose of our detection, we will focus on a specific hijack involving the vulnerable executable calc.exe and a list of DLLs that can be hijacked.

✗	calc.exe	CRYPTBASE.DLL	DllMain
✗		edutil.dll	DllMain
✗		MLANG.dll	EdpGetIsManaged
✗		PROPSYS.dll	ConvertINetUnicodeToMultiByte
✗		Secur32.dll	DllMain
✗		SSPICLIB.dll	PSCreateMemoryPropertyStore
✗		WININET.dll	PSPropertyBag_WriteDWORD
✗		WININET.dll	DllMain
✗		WININET.dll	GetUserNameExW
✗		WININET.dll	DllMain
✗		WININET.dll	GetUrlCacheEntryBinaryBlob

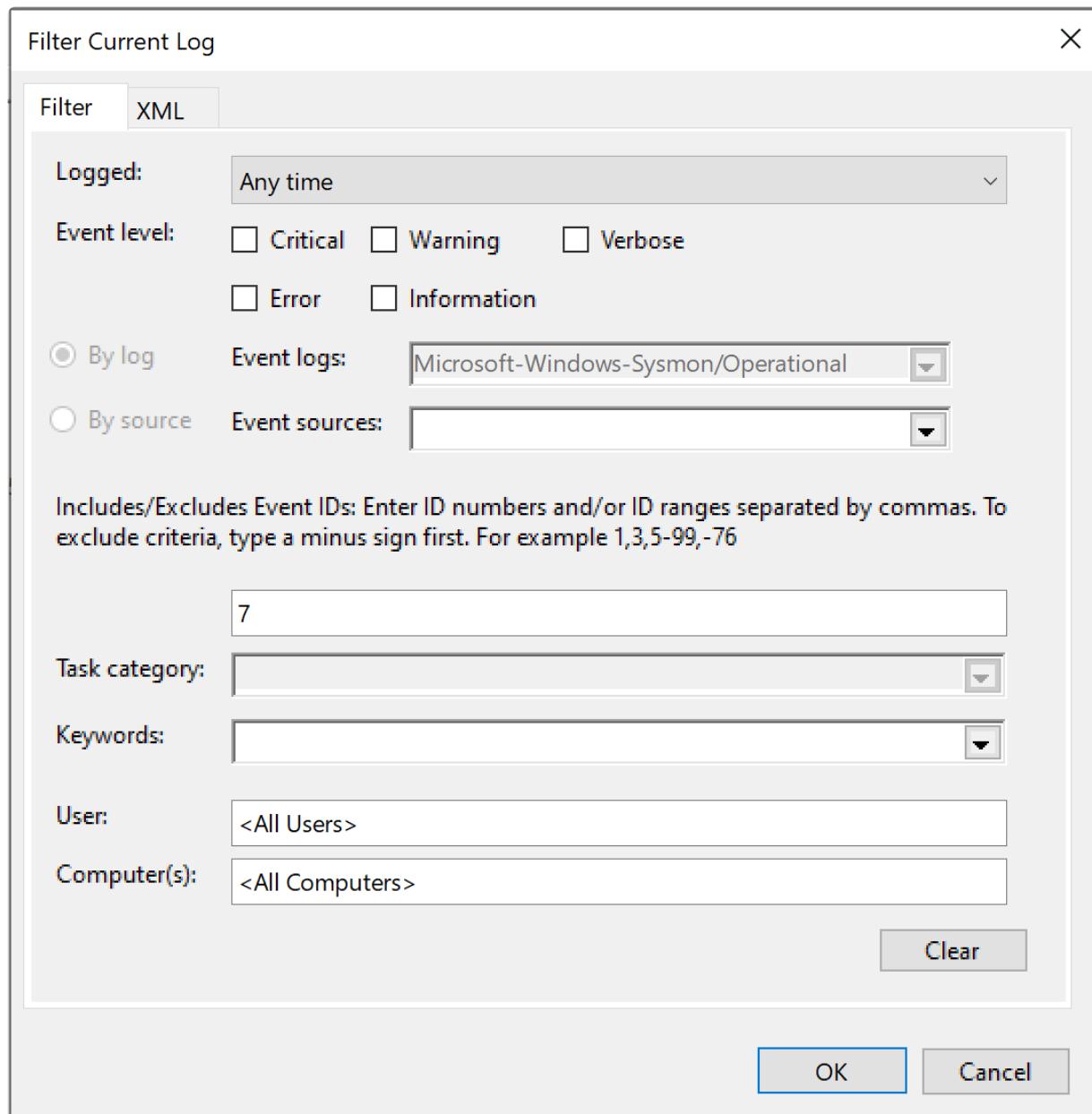
- Let's attempt the hijack using "calc.exe" and "WININET.dll" as an example.
 - To simplify the process, we can utilize Stephen Fewer's "hello world" [reflective DLL](#).
 - It should be noted that DLL hijacking does not require reflective DLLs.
- By following the required steps, which involve renaming `reflective_dll.x64.dll` to `WININET.dll`, moving `calc.exe` from `C:\Windows\System32` along with

`WININET.dll` to a writable directory (such as the `Desktop` folder), and executing `calc.exe`, we achieve success.

- Instead of the Calculator application, a `MessageBox` is displayed.



- Next, we analyze the impact of the hijack. First, we filter the event logs to focus on Event ID 7, which represents module load events, by clicking "Filter Current Log...".



- Subsequently, we search for instances of "calc.exe", by clicking "Find...", to identify the DLL load associated with our hijack.

```

Image loaded:
RuleName: -
UtcTime: 2022-04-07 22:36:42.918
ProcessGuid: (2c4ac1bf-677a-624f-930b-000000005d00)
ProcessId: 6212
Image: C:\Users\Waldo\Desktop\calc.exe
ImageLoaded: C:\Users\Waldo\Desktop\WININET.dll
FileVersion: -
Description: -
Product: -
Company: -
OriginalFileName: -
Hashes: MD5=[REDACTED]35C6,SHA256=[REDACTED]3B13,IMPHASH=[REDACTED]E833
Signed: false
Signature: -
SignatureStatus: Unavailable
User: DESKTOP-N33HELB\Waldo

```

Find

Find what: calc.exe

Find Next

Cancel

- The output from Sysmon provides valuable insights.
 - Now, we can observe several indicators of compromise (IOCs) to create effective

detection rules.

- Before moving forward though, let's compare this to an authenticate load of "wininet.dll" by "calc.exe".

The screenshot shows a debugger interface with two main windows. The top window is titled 'General' and displays a stack dump. The bottom window is a 'Find' dialog box with the text 'calc.exe' entered in the 'Find what:' field. The 'Find Next' button is visible, and the 'Cancel' button is at the bottom right.

Stack Dump (General Tab):

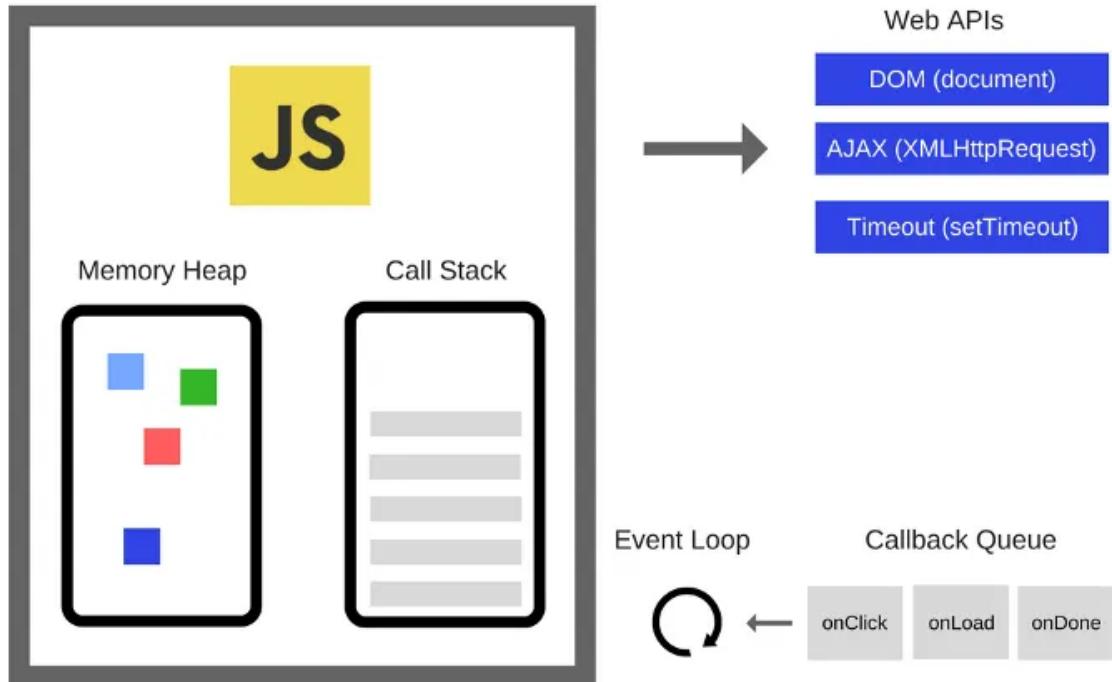
```
Image loaded: 
RuleName: - 
UtcTime: 2022-04-07 22:42:50.839 
ProcessGuid: {2c4ac1bf-68ea-624f-990b-000000005d00} 
ProcessId: 5464 
Image: C:\Windows\System32\calc.exe 
ImageLoaded: C:\Windows\System32\wininet.dll 
FileVersion: 11.00.18362.1766 (WinBuild.160101.0800) 
Description: Internet Extensions for Win32 
Product: Internet Explorer 
Company: Microsoft Corporation 
OriginalFileName: wininet.dll 
Hashes: MD5=464480C74457FB61876F037F0C9CB489,SHA256=C65F1733FDA1BCA1EB5A3B89387D8483626A9A5741B237A242E4967800ED9B0E,IMPHASH=6ADB8F4093077F6949DD162DA29572BB 
Signed: true 
Signature: Microsoft Windows 
SignatureStatus: Valid 
User: DESKTOP-N33HELB\Waldo
```

- Let's explore these IOCs:

1. "calc.exe", originally located in System32, should not be found in a writable directory.
 - Therefore, a copy of "calc.exe" in a writable directory serves as an IOC, as it should always reside in System32 or potentially Syswow64.
 2. "WININET.dll", originally located in System32, should not be loaded outside of System32 by calc.exe.
 - If instances of "WININET.dll" loading occur outside of System32 with "calc.exe" as the parent process, it indicates a DLL hijack within calc.exe.
 - While caution is necessary when alerting on all instances of "WININET.dll" loading outside of System32 (as some applications may package specific DLL versions for stability), in the case of "calc.exe", we can confidently assert a hijack due to the DLL's unchanging name, which attackers cannot modify to evade detection.
 3. The original "WININET.dll" is Microsoft-signed, while our injected DLL remains unsigned.
- These three powerful IOCs provide an effective means of detecting a DLL hijack involving calc.exe.
 - It's important to note that while Sysmon and event logs offer valuable telemetry for hunting and creating alert rules, they are not the sole sources of information.

Detection Example 2: Detecting Unmanaged PowerShell/C-Sharp Injection

- Before delving into detection techniques, let's gain a brief understanding of C# and its runtime environment.
 - C# is considered a "managed" language, meaning it requires a backend runtime to execute its code.
 - For e.g. in javascript, we have the js run time as follows for the browser:



- > It is defined as the environment that allows code to run.
- > Node.js has a different runtime environment, but the concept is nevertheless.
- > Similarly C# has a different runtime, but the concept is similar.
- The [Common Language Runtime \(CLR\)](#) serves as this runtime environment.
- [Managed code](#) does not directly run as assembly; instead, it is compiled into a bytecode format that the runtime processes and executes. Consequently, a managed process relies on the CLR to execute C# code.
- As defenders, we can leverage this knowledge to detect unusual C# injections or executions within our environment. To accomplish this, we can utilize a useful utility

called Process Hacker.

Processes	Services	Network	Disk	Firewall	6.2...	64.03 k...	1.58 GB	User name	Description
Name					PID	CPU	I/O total r...	Private b...	
Memory Compression					2456			116 kB	NT AUTHORITY\SYSTEM
Microsoft.Photos.exe					4816			20.67 MB	DESKTOP-R4PEEIF\waldc
msedge.exe					548			10.1 MB	DESKTOP-R4PEEIF\waldc Microsoft Edge
msedge.exe					1124			81.19 MB	DESKTOP-R4PEEIF\waldc Microsoft Edge
msedge.exe					1876			1.88 MB	DESKTOP-R4PEEIF\waldc Microsoft Edge
msedge.exe					4492			99.34 MB	DESKTOP-R4PEEIF\waldc Microsoft Edge
msedge.exe					5480			12.26 MB	DESKTOP-R4PEEIF\waldc Microsoft Edge
msedge.exe					7236			6.72 MB	DESKTOP-R4PEEIF\waldc Microsoft Edge
msedge.exe					7336	0.03	1.13 kB/s	29.41 MB	DESKTOP-R4PEEIF\waldc Microsoft Edge
MsMpEng.exe					2572			148.1 MB	NT AUTHORITY\SYSTEM Antimalware Service Executable
NisSrv.exe					5560			3.95 MB	NT AUTHORITY\LOCAL SERVICE Microsoft Network Realtime I...
OneDrive.exe					5380			25.84 MB	DESKTOP-R4PEEIF\waldc Microsoft OneDrive
powershell.exe					7124			67.13 MB	DESKTOP-R4PEEIF\waldc Windows PowerShell
ProcessHacker.exe					6584	0.71		23.84 MB	DESKTOP-R4PEEIF\waldc Process Hacker
rdclip.exe					8276			3.43 MB	DESKTOP-R4PEEIF\waldc RDP Clipboard Monitor
Registry					112			2.54 MB	NT AUTHORITY\SYSTEM
RuntimeBroker.exe					3672			7.02 MB	DESKTOP-R4PEEIF\waldc Runtime Broker
RuntimeBroker.exe					4372			25.32 MB	DESKTOP-R4PEEIF\waldc Runtime Broker
RuntimeBroker.exe					6968			5.68 MB	DESKTOP-R4PEEIF\waldc Runtime Broker
RuntimeBroker.exe					8464			6.93 MB	DESKTOP-R4PEEIF\waldc Runtime Broker
RuntimeBroker.exe					8848			2.96 MB	DESKTOP-R4PEEIF\waldc Runtime Broker
RuntimeBroker.exe					8900			5.05 MB	DESKTOP-R4PEEIF\waldc Runtime Broker
SearchApp.exe					1728			129.84 MB	DESKTOP-R4PEEIF\waldc Search application
SearchApp.exe					5256			16.02 MB	DESKTOP-R4PEEIF\waldc Search application
SearchFilterHost.exe					3780			2.3 MB	NT AUTHORITY\SYSTEM Microsoft Windows Search Fil...
SearchIndexer.exe					3936			32.28 MB	NT AUTHORITY\SYSTEM Microsoft Windows Search In...
SearchProtocolHost.exe					5900			2.93 MB	NT AUTHORITY\SYSTEM Microsoft Windows Search Pr...
SecurityHealthService.exe					8952			3.82 MB	NT AUTHORITY\SYSTEM Windows Security Health Serv...
SecurityHealthSystray.exe					8772			1.86 MB	DESKTOP-R4PEEIF\waldc Windows Security notification...
services.exe					656	0.28		5.68 MB	NT AUTHORITY\SYSTEM Services and Controller app
SgrmBroker.exe					552			4.63 MB	NT AUTHORITY\SYSTEM System Guard Runtime Monit...
ShellExperienceHost.exe					1436			17.19 MB	DESKTOP-R4PEEIF\waldc Windows Shell Experience Host
Total					7140			6.02 MB	DESKTOP-R4PEEIF\waldc

CPU usage: 6.24% Physical memory: 2.52 GB (31.56%) Free memory: 5.47 GB (68.44%)

- By using Process Hacker, we can observe a range of processes within our environment.
 - Sorting the processes by name, we can identify interesting color-coded distinctions.
 - Notably, "powershell.exe", a managed process, is highlighted in green compared to other processes.
 - Hovering over powershell.exe reveals the label "Process is managed (.NET),"

confirming its managed status.

powershell.exe	7124	67.08 MB	DESKTOP-F
ProcessHacker.exe			P-F
rdclip.exe			P-F
Registry			HC
RuntimeBroker.exe			P-F
RuntimeBroker.exe	8900	4.91 MB	DESKTOP-F

- Examining the module loads for `powershell.exe`, by right-clicking on `powershell.exe`, clicking "Properties", and navigating to "Modules", we can find relevant information.

clr.dll	0x7ffa0644...	10.76 MB	Microsoft .NET Runtime Common Language Runtime - WorkStation
clrjit.dll	0x7ffa136f...	1.31 MB	Microsoft .NET Runtime Just-In-Time Compiler

- The presence of "Microsoft .NET Runtime...", `clr.dll`, and `clrjit.dll` should attract our attention.
 - These 2 DLLs are used when C# code is ran as part of the runtime to execute the bytecode.
 - If we observe these DLLs loaded in processes that typically do not require them, it suggests a potential `execute-assembly` or unmanaged PowerShell injection attack.
- To showcase unmanaged PowerShell injection, we can inject an unmanaged PowerShell-like DLL into a random process, such as `spoolsv.exe`.
 - We can do that by utilizing the [PSInject project](#) in the following manner.

```
powershell -ep bypass
Import-Module .\Invoke-PSInject.ps1
Invoke-PSInject -ProcId [Process ID of spoolsv.exe] -PoshCode
"V3JpdGutSG9zdCAiSGVsbG8sIEd1cnU50SEi"
-> Process ID can be observed from process Hacker
```

```
Invoke-PSInject -ProcId 3736 -PoshCode
"V3JpdGutSG9zdCAiSGVsbG8sIEd1cnU50SEi"
```

	spoolsv.exe	2792	8.92 MB	N
	StartMenuExperienceHost.exe	5440	22.02 MB	D
	C:\Windows\System32\spoolsv.exe			
	File:			
	C:\Windows\System32\spoolsv.exe			
	Spooler SubSystem App 10.0.19041.1288 (WinBuild.160101.0800)			
	Microsoft Corporation			
	Services:			
	Spooler (Print Spooler)			
	Notes:			
	Signer: Microsoft Windows			
	Console application: services.exe (656)			

- After the injection, we observe that "spoolsv.exe" transitions from an unmanaged to a managed state.

	spoolsv.exe	2792		
	StartMenuExperienceHost.exe	5440		
	C:\Windows\System32\spoolsv.exe			
	File:			
	C:\Windows\System32\spoolsv.exe			
	Spooler SubSystem App 10.0.19041.1288 (WinBuild.160101.0800)			
	Microsoft Corporation			
	Services:			
	Spooler (Print Spooler)			
	Notes:			
	Signer: Microsoft Windows			
	Console application: services.exe (656)			
	Process is managed (.NET).			

- Additionally, by referring to both the related "Modules" tab of Process Hacker and Sysmon Event ID 7, we can examine the DLL load information to validate the presence of the aforementioned DLLs.

spoolsv.exe (2792) Properties

Options			
Name	Base address	Size	Description
5b120a24.BUD	0x530000	16 kB	
advapi32.dll	0x7ffa4894...	688 kB	Advanced Windows 32 Base API
amsi.dll	0x7ffa40c3...	100 kB	Anti-Malware Scan Interface
APMon.dll	0x7ffa279f...	1.44 MB	Adaptive Port Monitor
APMon.dll.mui	0x12f0000	4 kB	Adaptive Port Monitor
apphelp.dll	0x7ffa44be...	576 kB	Application Compatibility Client Library
AppMon.dll	0x7ffa27ce...	136 kB	App Printer
BCP47Langs.dll	0x7ffa3fd3...	368 kB	BCP47 Language Classes
bcrypt.dll	0x7ffa476e...	156 kB	Windows Cryptographic Primitives Library
bcryptprimitives...	0x7ffa4727...	524 kB	Windows Cryptographic Primitives Library
bidispl.dll	0x7ffa2733...	80 kB	Bidispl DLL
cabinet.dll	0x7ffa414f...	164 kB	Microsoft® Cabinet File API
cfgmgr32.dll	0x7ffa4795...	312 kB	Configuration Manager DLL
dbcatalog.dll	0x7ffa47e7...	676 kB	COM+ Configuration Catalog
clr.dll	0x7ffa0644...	10.76 MB	Microsoft .NET Runtime Common Language Runtime - WorkStation
clrjit.dll	0x7ffa136f...	1.31 MB	Microsoft .NET Runtime Just-In-Time Compiler

Event 7, Sysmon

General Details

```
Image loaded:  
RuleName: -  
UtcTime: 2022-04-28 00:38:37.608  
ProcessGuid: {67e39d39-e762-6269-3e00-000000000200}  
ProcessId: 2792  
Image: C:\Windows\System32\spoolsv.exe  
ImageLoaded: C:\Windows\Microsoft.NET\Framework64\v4.0.30319\clr.dll  
FileVersion: 4.8.4400.0 built by: NET48REL1LAST_C  
Description: Microsoft .NET Runtime Common Language Runtime - WorkStation  
Product: Microsoft® .NET Framework  
Company: Microsoft Corporation  
OriginalFileName: clr.dll  
Hashes: SHA1=C8EB4C4F03D801973933D23CF579EFF12A21FAEA,MD5=70384D367DBBF68B996289FE37C19DA2,SHA256=1E39D345F0D7809656656EB2A19533CF42E093F5985A696F45B07F2299DE9507,IMPHASH=6851068577998FF473E5933122867348  
Signed: true  
Signature: Microsoft Corporation  
SignatureStatus: Valid  
User: NT AUTHORITY\SYSTEM
```

Detection Example 3: Detecting Credential Dumping

- Another critical aspect of cybersecurity is detecting credential dumping activities.
 - One widely used tool for credential dumping is [Mimikatz](#), offering various methods for extracting Windows credentials.
 - One specific command, "sekurlsa::logonpasswords", enables the dumping of password hashes or plaintext passwords by accessing the Local Security Authority Subsystem Service (LSASS).
 - LSASS is responsible for managing user credentials and is a primary target for credential-dumping tools like Mimikatz.

- The attack can be executed as follows.

```
C:\Tools\Mimikatz> mimikatz.exe

.#####. mimikatz 2.2.0 (x64) #18362 Feb 29 2020 11:13:36
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##      > http://blog.gentilkiwi.com/mimikatz
## v ##      Vincent LE TOUX          ( vincent.letoux@gmail.com
)
'#####'      > http://pingcastle.com / http://mysmartlogon.com
***/



mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 1128191 (00000000:001136ff)
Session           : RemoteInteractive from 2
User Name         : Administrator
Domain            : DESKTOP-NU10MTO
Logon Server      : DESKTOP-NU10MTO
Logon Time        : 5/31/2023 4:14:41 PM
SID               : S-1-5-21-2712802632-2324259492-1677155984-500

msv :
[00000003] Primary
* Username : Administrator
* Domain   : DESKTOP-NU10MTO
* NTLM     : XXXXXXXXXXXXXXXXXXXXXXXXX
* SHA1     : XXXXXXXXXXXXXXXXXXXXXXXXX0812156b

tspkg :

wdigest :
* Username : Administrator
* Domain   : DESKTOP-NU10MTO
* Password : (null)

kerberos :
* Username : Administrator
* Domain   : DESKTOP-NU10MTO
* Password : (null)

ssp : K0

credman :
```

- As we can see, the output of the "sekurlsa::logonpasswords" command provides powerful insights into compromised credentials.
- To detect this activity, we can rely on a different Sysmon event.
 - Instead of focusing on DLL loads, we shift our attention to process access events.
 - By checking **Sysmon event ID 10**, which represents "ProcessAccess" events, we can identify any suspicious attempts to access LSASS.

› Event ID 10: ProcessAccess

The process accessed event reports when a process opens another process, an operation that's often followed by information queries or reading and writing the address space of the target process. This enables detection of hacking tools that read the memory contents of processes like Local Security Authority (Lsass.exe) in order to steal credentials for use in Pass-the-Hash attacks. Enabling it can generate significant amounts of logging if there are diagnostic utilities active that repeatedly open processes to query their state, so it generally should only be done so with filters that remove expected accesses.

Event 10, Sysmon

General	Details
Process accessed: RuleName: - UtcTime: 2022-04-28 00:49:41.014 SourceProcessGUID: {67e39d39-e132-6269-6003-000000000200} SourceProcessId: 3824 SourceThreadId: 6824 SourceImage: C:\Users\waldo\Downloads\AgentEXE.exe TargetProcessGUID: {67e39d39-e75a-6269-0c00-000000000200} TargetProcessId: 676 TargetImage: C:\Windows\system32\lsass.exe GrantedAccess: 0x1010 CallTrace: C:\Windows\SYSTEM32\ntdll.dll+9d234 C:\Windows\System32\KERNELBASE.dll+2c0fe UNKNOWN(0000000180085CA7) SourceUser: DESKTOP-R4PEEIF\waldo TargetUser: NT AUTHORITY\SYSTEM	

- For instance, if we observe a random file ("AgentEXE" in this case) from a random folder ("Downloads" in this case) attempting to access LSASS, it indicates unusual behavior.
 - Additionally, the **SourceUser** being different from the **TargetUser** (e.g., "waldo" as the **SourceUser** and "SYSTEM" as the **TargetUser**) further emphasizes the abnormality.
 - It's also worth noting that as part of the mimikatz-based credential dumping process, the user must request **SeDebugPrivileges**.
 - As the name suggests, it's primarily used for debugging.
 - This can be another Indicator of Compromise (IOC).
- Please note that some legitimate processes may access LSASS, such as authentication-related processes or security tools like AV or EDR.

Additional Telemetry Sources

Event Tracing for Windows

Overview

- In the realm of effective threat detection and incident response, we often find ourselves relying on the limited log data at our disposal.
- However, this approach falls short of fully harnessing the immense wealth of information that can be derived from the powerful resource known as `Event Tracing for Windows (ETW)`.
- Unfortunately, this oversight can be attributed to a lack of awareness and appreciation for the comprehensive and intricate insights that ETW can offer.

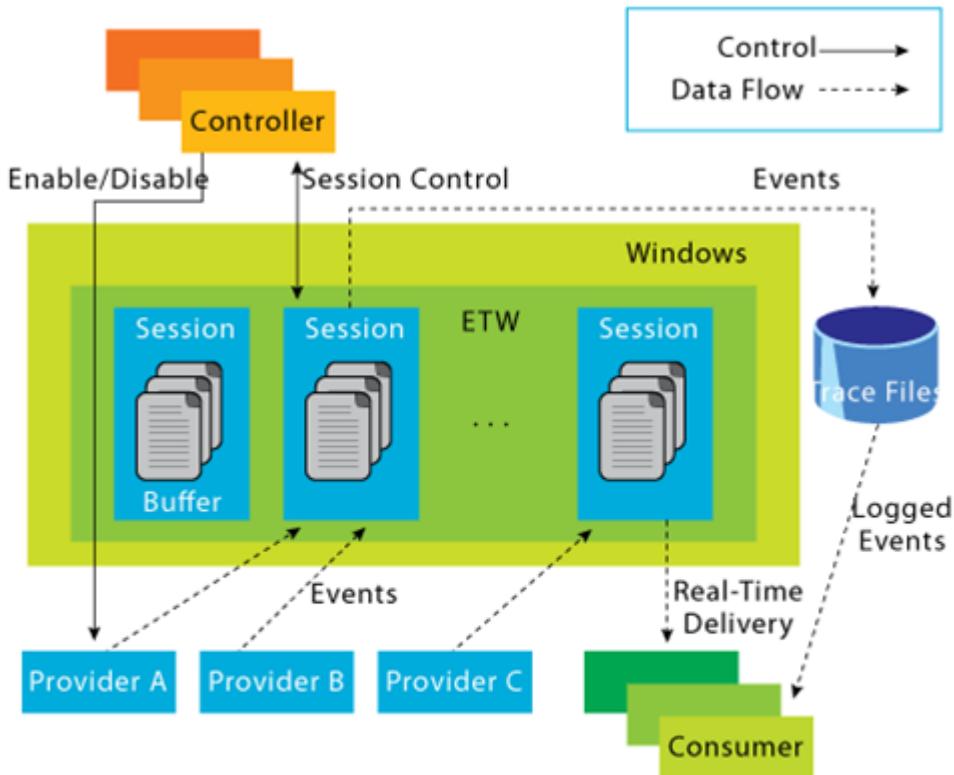
What is ETW?

- According to Microsoft, `Event Tracing For Windows (ETW)` is a general-purpose, high-speed tracing facility provided by the operating system.
- Using a buffering and logging mechanism implemented in the kernel, ETW provides a tracing mechanism for events raised by both user-mode applications and kernel-mode device drivers.
- ETW, functioning as a high-performance event tracing mechanism deeply embedded within the Windows operating system, presents an unparalleled opportunity to bolster our defense capabilities.
 - Its architecture facilitates the dynamic generation, collection, and analysis of various events occurring within the system, resulting in the creation of intricate, real-time logs that encompass a wide spectrum of activities.
- By effectively leveraging ETW, we can tap into an expansive array of telemetry sources that surpass the limitations imposed by traditional log data.
 - ETW captures a diverse set of events, spanning system calls, process creation and termination, network activity, file and registry modifications, and numerous other dimensions.
 - These events collectively weave a detailed tapestry of system activity, furnishing invaluable context for the identification of anomalous behavior, discovery of potential security incidents, and facilitation of forensic investigations.
- ETW's versatility and extensibility are further accentuated by its seamless integration with Event Providers.
 - These specialized components generate specific types of events and can be seamlessly incorporated into applications, operating system components, or third-party software.
 - Consequently, this integration ensures a broad coverage of potential event sources.

- Furthermore, ETW's extensibility enables the creation of custom providers tailored to address specific organizational requirements, thereby fostering a targeted and focused approach to logging and monitoring.
- Notably, ETW's lightweight nature and minimal performance impact render it an optimal telemetry solution for real-time monitoring and continuous security assessment.
 - By selectively enabling and configuring relevant event providers, we can finely adjust the scope of data collection to align with our specific security objectives, striking a harmonious balance between the richness of information and system performance considerations.
- Moreover, the existence of robust tooling and utilities, including Microsoft's Message Analyzer and PowerShell's `Get-WinEvent` cmdlet, greatly simplifies the retrieval, parsing, and analysis of ETW logs.
 - These tools offer advanced filtering capabilities, event correlation mechanisms, and real-time monitoring features, empowering members of the blue team to extract actionable insights from the vast pool of information captured by ETW.

ETW Architecture & Components

- The underlying architecture and the key components of Event Tracing for Windows (ETW) are illustrated in the following diagram from [Microsoft](#).



Controllers: The Controllers component, as its name implies, assumes control over all aspects related to ETW operations.

- It encompasses functionalities such as initiating and terminating trace sessions, as well as enabling or disabling providers within a particular trace.
- Trace sessions can establish subscriptions to one or multiple providers, thereby granting the providers the ability to commence logging operations.
- An example of a widely used controller is the built-in utility "logman.exe," which facilitates the management of ETW activities.
- At the core of ETW's architecture is the publish-subscribe model. This model involves two primary components:
 - **Providers** : Providers play a pivotal role in generating events and writing them to the designated ETW sessions.
 - Applications have the ability to register ETW providers, enabling them to generate and transmit numerous events.
 - There are four distinct types of providers utilized within ETW.
 - **MOF Providers** : These providers are based on Managed Object Format (MOF) and are capable of generating events according to predefined MOF schemas.
 - They offer a flexible approach to event generation and are widely used in various scenarios.

- **WPP Providers** : Standing for "Windows Software Trace Preprocessor," WPP providers leverage specialized macros and annotations within the application's source code to generate events.
 - This type of provider is often utilized for low-level kernel-mode tracing and debugging purposes.
- **Manifest-based Providers** : Manifest-based providers represent a more contemporary form of providers within ETW.
 - They rely on XML manifest files that define the structure and characteristics of events.
 - This approach offers enhanced flexibility and ease of management, allowing for dynamic event generation and customization.
- **TraceLogging Providers** : TraceLogging providers offer a simplified and efficient approach to event generation.
 - They leverage the TraceLogging API, introduced in recent Windows versions, which streamlines the process of event generation with minimal code overhead.
- **Consumers** : Consumers subscribe to specific events of interest and receive those events for further processing or analysis.
 - By default, the events are typically directed to an .ETL (Event Trace Log) file for handling.
 - However, an alternative consumer scenario involves leveraging the capabilities of the Windows API to process and consume the events.
- **Channels** : To facilitate efficient event collection and consumption, ETW relies on event channels.
 - Event channels act as logical containers for organizing and filtering events based on their characteristics and importance.
 - ETW supports multiple channels, each with its own defined purpose and audience.
 - Event consumers can selectively subscribe to specific channels to receive relevant events for their respective use cases.
- **ETL files** : ETW provides specialized support for writing events to disk through the use of event trace log files, commonly referred to as "ETL files."
 - These files serve as durable storage for events, enabling offline analysis, long-term archiving, and forensic investigations.
 - ETW allows for seamless rotation and management of ETL files to ensure efficient storage utilization.

Notes:

- ETW supports event providers in both kernel mode and user mode.
- Some event providers generate a significant volume of events, which can potentially overwhelm the system resources if they are constantly active.
 - As a result, to prevent unnecessary resource consumption, these providers are typically disabled by default and are only enabled when a tracing session specifically requests their activation.
- In addition to its inherent capabilities, ETW can be extended through custom event providers.
- Only ETW provider events that have a Channel property applied to them can be consumed by the event log

Interacting With ETW

- `Logman` is a pre-installed utility for managing Event Tracing for Windows (ETW) and Event Tracing Sessions.
 - This tool is invaluable for creating, initiating, halting, and investigating tracing sessions.
 - This is particularly useful when determining which sessions are set for data collection or when initiating your own data collection.
- Employing the `-ets` parameter will allow for a direct investigation of the event tracing sessions, providing insights into system-wide tracing sessions.
 - As an example, the Sysmon Event Tracing Sessions can be found towards the end of the displayed information.

```
C:\Tools> logman.exe query -ets
```

Data Collector Set	Type
Status	

Circular Kernel Context Logger	Trace
Running	
Eventlog-Security	Trace
Running	
DiagLog	Trace
Running	
Diagtrack-Listener	Trace
Running	
EventLog-Application	Trace
Running	
EventLog-Microsoft-Windows-Sysmon-Operational	Trace

Running
EventLog-System Trace
Running
LwtNetLog Trace
Running
Microsoft-Windows-Rdp-Graphics-RdpIdd-Trace Trace
Running
NetCore Trace
Running
NtfsLog Trace
Running
RadioMgr Trace
Running
UBPM Trace
Running
WdiContextLog Trace
Running
WiFiSession Trace
Running
SHS-06012023-115154-7-7f Trace
Running
UserNotPresentTraceSession Trace
Running
8696EAC4-1288-4288-A4EE-49EE431B0AD9 Trace
Running
ScreenOnPowerStudyTraceSession Trace
Running
SYSMON TRACE Trace
Running
MSDTC_TRACE_SESSION Trace
Running
SysmonDnsEtwSession Trace
Running
MpWppTracing-20230601-115025-00000003-ffffffff Trace
Running
WindowsUpdate_trace_log Trace
Running
Admin_PS_Provider Trace
Running
Terminal-Services-LSM-ApplicationLag-3764 Trace
Running
Microsoft.Windows.Remediation Trace
Running
SgrmEtwSession Trace
Running

The command completed successfully.

- When we examine an Event Tracing Session directly, we uncover specific session details including the Name, Max Log Size, Log Location, and the subscribed providers.
 - This information is invaluable for incident responders. Discovering a session that records providers relevant to your interests may provide crucial logs for an investigation.
- Please note that the “-ets” parameter is vital to the command. Without it, Logman will not identify the Event Tracing Session.
- For each provider subscribed to the session, we can acquire critical data:
 - **Name / Provider GUID** : This is the exclusive identifier for the provider.
 - **Level** : This describes the event level, indicating if it's filtering for warning, informational, critical, or all events.
 - **Keywords Any** : Keywords create a filter based on the kind of event generated by the provider.

```
C:\Tools> logman.exe query "EventLog-System" -ets
```

```
Name: EventLog-System
Status: Running
Root Path: %systemdrive%\PerfLogs\Admin
Segment: Off
Schedules: On
Segment Max Size: 100 MB
```

```
Name: EventLog-System\EventLog-System
Type: Trace
Append: Off
Circular: Off
Overwrite: Off
Buffer Size: 64
Buffers Lost: 0
Buffers Written: 47
Buffer Flush Timer: 1
Clock Type: System
File Mode: Real-time
```

```
Provider:
Name: Microsoft-Windows-FunctionDiscoveryHost
Provider Guid: {538CBBAD-4877-4EB2-B26E-7CAEE8F0F8CB}
Level: 255
```

```
KeywordsAll:          0x0
KeywordsAny:          0x8000000000000000 (System)
Properties:           65
Filter Type:          0

Provider:
Name:                 Microsoft-Windows-Subsys-SMSS
Provider Guid:        {43E63DA5-41D1-4FBF-ADED-1BBED98FDD1D}
Level:                255
KeywordsAll:          0x0
KeywordsAny:          0x4000000000000000 (System)
Properties:           65
Filter Type:          0

Provider:
Name:                 Microsoft-Windows-Kernel-General
Provider Guid:        {A68CA8B7-004F-D7B6-A698-07E2DE0F1F5D}
Level:                255
KeywordsAll:          0x0
KeywordsAny:          0x8000000000000000 (System)
Properties:           65
Filter Type:          0

Provider:
Name:                 Microsoft-Windows-FilterManager
Provider Guid:        {F3C5E28E-63F6-49C7-A204-E48A1BC4B09D}
Level:                255
KeywordsAll:          0x0
KeywordsAny:          0x8000000000000000 (System)
Properties:           65
Filter Type:          0
```

--- SNIP ---

The command completed successfully.

- By using the `logman query providers` command, we can generate a list of all available providers on the system, including their respective GUIDs.

```
C:\Tools> logman.exe query providers
```

Provider	GUID
-----	-----

ACPI Driver Trace Provider	{DAB01D4D-2D48-477D-B1C3-
DAAD0CE6F06B}	
Active Directory Domain Services: SAM	{8E598056-8993-11D2-819E-
0000F875A064}	
Active Directory: Kerberos Client	{BBA3ADD2-C229-4CDB-AE2B-
57EB6966B0C4}	
Active Directory: NetLogon	{F33959B4-DBEC-11D2-895B-
00C04F79AB69}	
ADODB.1	{04C8A86F-3369-12F8-4769-
24E484A9E725}	
ADOMD.1	{7EA56435-3F2F-3F63-A829-
F0B35B5CAD41}	
Application Popup	{47BFA2B7-BD54-4FAC-B70B-
29021084CA8F}	
Application-Addon-Event-Provider	{A83FA99F-C356-4DED-9FD6-
5A5EB8546D68}	
ATA Port Driver Tracing Provider	{D08BD885-501E-489A-BAC6-
B7D24BFE6BBF}	
AuthFw NetShell Plugin	{935F4AE6-845D-41C6-97FA-
380DAD429B72}	
BCP.1	{24722B88-DF97-4FF6-E395-
DB533AC42A1E}	
BFE Trace Provider	{106B464A-8043-46B1-8CB8-
E92A0CD7A560}	
BITS Service Trace	{4A8AAA94-CFC4-46A7-8E4E-
17BC45608F0A}	
Certificate Services Client CredentialRoaming Trace	{EF4109DC-68FC-45AF-
B329-CA2825437209}	
Certificate Services Client Trace	{F01B7774-7ED7-401E-8088-
B576793D7841}	
Circular Kernel Session Provider	{54DEA73A-ED1F-42A4-AF71-
3E63D056F174}	
Classpnp Driver Tracing Provider	{FA8DE7C4-ACDE-4443-9994-
C4E2359A9EDB}	
Critical Section Trace Provider	{3AC66736-CC59-4CFF-8115-
8DF50E39816B}	
DBNETLIB.1	{BD568F20-FCCD-B948-054E-
DB3421115D61}	
Deduplication Tracing Provider	{5EBB59D1-4739-4E45-872D-
B8703956D84B}	
Disk Class Driver Tracing Provider	{945186BF-3DD6-4F3F-9C8E-
9EDD3FC9D558}	
Downlevel IPsec API	{94335EB3-79EA-44D5-8EA9-
306F49B3A041}	
Downlevel IPsec NetShell Plugin	{E4FF10D8-8A88-4FC6-82C8-
8C23E9462FE5}	

Downlevel IPsec Policy Store	{94335EB3-79EA-44D5-8EA9-
306F49B3A070}	
Downlevel IPsec Service	{94335EB3-79EA-44D5-8EA9-
306F49B3A040}	
EA IME API	{E2A24A32-00DC-4025-9689-
C108C01991C5}	
Error Instrument	{CD7CF0D0-02CC-4872-9B65-
0DBA0A90EFE8}	
FD Core Trace	{480217A9-F824-4BD4-BBE8-
F371CAAF9A0D}	
FD Publication Trace	{649E3596-2620-4D58-A01F-
17AEFE8185DB}	
FD SSDP Trace	{DB1D0418-105A-4C77-9A25-
8F96A19716A4}	
FD WNet Trace	{8B20D3E4-581F-4A27-8109-
DF01643A7A93}	
FD WSDAPI Trace	{7E2DBFC7-41E8-4987-BCA7-
76CADFAD765F}	
FDPHost Service Trace	{F1C521CA-DA82-4D79-9EE4-
D7A375723B68}	
File Kernel Trace; Operation Set 1	{D75D8303-6C21-4BDE-9C98-
ECC6320F9291}	
File Kernel Trace; Operation Set 2	{058DD951-7604-414D-A5D6-
A56D35367A46}	
File Kernel Trace; Optional Data	{7DA1385C-F8F5-414D-B9D0-
02FCA090F1EC}	
File Kernel Trace; Volume To Log	{127D46AF-4AD3-489F-9165-
F00BA64D5467}	
FWPKCLNT Trace Provider	{AD33FA19-F2D2-46D1-8F4C-
E3C3087E45AD}	
FPUCLNT Trace Provider	{5A1600D2-68E5-4DE7-BCF4-
1C2D215FE0FE}	
Heap Trace Provider	{222962AB-6180-4B88-A825-
346B75F2A24A}	
IKEEXT Trace Provider	{106B464D-8043-46B1-8CB8-
E92A0CD7A560}	
IMAPI1 Shim	{1FF10429-99AE-45BB-8A67-
C9E945B9FB6C}	
IMAPI2 Concatenate Stream	{0E85A5A5-4D5C-44B7-8BDA-
5B7AB54F7E9D}	
IMAPI2 Disc Master	{0E85A5A5-4D5C-44B7-8BDA-
5B7AB54F7E91}	
IMAPI2 Disc Recorder	{0E85A5A5-4D5C-44B7-8BDA-
5B7AB54F7E93}	
IMAPI2 Disc Recorder Enumerator	{0E85A5A5-4D5C-44B7-8BDA-
5B7AB54F7E92}	

IMAPI2 dll	{0E85A5A5-4D5C-44B7-8BDA-
5B7AB54F7E90}	
IMAPI2 Interleave Stream	{0E85A5A5-4D5C-44B7-8BDA-
5B7AB54F7E9E}	
IMAPI2 Media Eraser	{0E85A5A5-4D5C-44B7-8BDA-
5B7AB54F7E97}	
IMAPI2 MSF	{0E85A5A5-4D5C-44B7-8BDA-
5B7AB54F7E9F}	
IMAPI2 Multisession Sequential	{0E85A5A5-4D5C-44B7-8BDA-
5B7AB54F7EA0}	
IMAPI2 Pseudo-Random Stream	{0E85A5A5-4D5C-44B7-8BDA-
5B7AB54F7E9C}	
IMAPI2 Raw CD Writer	{0E85A5A5-4D5C-44B7-8BDA-
5B7AB54F7E9A}	
IMAPI2 Raw Image Writer	{07E397EC-C240-4ED7-8A2A-
B9FF0FE5D581}	
IMAPI2 Standard Data Writer	{0E85A5A5-4D5C-44B7-8BDA-
5B7AB54F7E98}	
IMAPI2 Track-at-Once CD Writer	{0E85A5A5-4D5C-44B7-8BDA-
5B7AB54F7E99}	
IMAPI2 Utilities	{0E85A5A5-4D5C-44B7-8BDA-
5B7AB54F7E94}	
IMAPI2 Write Engine	{0E85A5A5-4D5C-44B7-8BDA-
5B7AB54F7E96}	
IMAPI2 Zero Stream	{0E85A5A5-4D5C-44B7-8BDA-
5B7AB54F7E9B}	
IMAPI2FS Tracing	{F8036571-42D9-480A-BABB-
DE7833CB059C}	
Intel-iaLPSS-GPI0	{D386CC7A-620A-41C1-ABF5-
55018C6C699A}	
Intel-iaLPSS-I2C	{D4AEAC44-AD44-456E-9C90-
33F8CDCED6AF}	
Intel-iaLPSS2-GPI02	{63848CFF-3EC7-4DDF-8072-
5F95E8C8EB98}	
Intel-iaLPSS2-I2C	{C2F86198-03CA-4771-8D4C-
CE6E15CBCA56}	
IPMI Driver Trace	{D5C6A3E9-FA9C-434E-9653-
165B4FC869E4}	
IPMI Provider Trace	{651D672B-E11F-41B7-ADD3-
C2F6A4023672}	
KMDFv1 Trace Provider	{544D4C9D-942C-46D5-BF50-
DF5CD9524A50}	
Layer2 Security HC Diagnostics Trace	{2E8D9EC5-A712-48C4-8CE0-
631EB0C1CD65}	
Local Security Authority (LSA)	{CC85922F-DB41-11D2-9244-
006008269001}	

LsaSrv	{199FE037-2B82-40A9-82AC-
E1D46C792B99}	
Microsoft-Antimalware-AMFilter	{CFEB0608-330E-4410-B00D-
56D8DA9986E6}	
Microsoft-Antimalware-Engine	{0A002690-3839-4E3A-B3B6-
96D8DF868D99}	
Microsoft-Antimalware-Engine-Instrumentation	{68621C25-DF8D-4A6B-AABC-
19A22E296A7C}	
Microsoft-Antimalware-NIS	{102AAB0A-9D9C-4887-A860-
55DE33B96595}	
Microsoft-Antimalware-Protection	{E4B70372-261F-4C54-8FA6-
A5A7914D73DA}	
Microsoft-Antimalware-RTP	{8E92DEEF-5E17-413B-B927-
59B2F06A3CFC}	
Microsoft-Antimalware-Scan-Interface	{2A576B87-09A7-520E-C21A-
4942F0271D67}	
Microsoft-Antimalware-Service	{751EF305-6C6E-4FED-B847-
02EF79D26AEF}	
Microsoft-Antimalware-ShieldProvider	{928F7D29-0577-5BE5-3BD3-
B6BDAB9AB307}	
Microsoft-Antimalware-UacScan	{D37E7910-79C8-57C4-DA77-
52BB646364CD}	
Microsoft-AppV-Client	{E4F68870-5AE8-4E5B-9CE7-
CA9ED75B0245}	
Microsoft-AppV-Client-StreamingUX	{28CB46C7-4003-4E50-8BD9-
442086762D12}	
Microsoft-AppV-ServiceLog	{9CC69D1C-7917-4ACD-8066-
6BF8B63E551B}	
Microsoft-AppV-SharedPerformance	{FB4A19EE-EB5A-47A4-BC52-
E71AAC6D0859}	
Microsoft-Client-Licensing-Platform	{B6CC0D55-9ECC-49A8-B929-
2B9022426F2A}	
Microsoft-Gaming-Services	{BC1BDB57-71A2-581A-147B-
E0B49474A2D4}	
Microsoft-IE	{9E3B3947-CA5D-4614-91A2-
7B624E0E7244}	
Microsoft-IE-JSDumpHeap	{7F8E35CA-68E8-41B9-86FE-
D6ADC5B327E7}	
Microsoft-IEFRAME	{5C8BB950-959E-4309-8908-
67961A1205D5}	
Microsoft-JScript	{57277741-3638-4A4B-BDBA-
0AC6E45DA56C}	
Microsoft-OneCore-OnlineSetup	{41862974-DA3B-4F0B-97D5-
BB29FBB9B71E}	
Microsoft-PerfTrack-IEFRAME	{B2A40F1F-A05A-4DFD-886A-
4C4F18C4334C}	

```
Microsoft-PerfTrack-MSHTML {FFDB9886-80F3-4540-AA8B-B85192217DDF}
Microsoft-User Experience Virtualization-Admin {61BC445E-7A8D-420E-AB36-9C7143881B98}
Microsoft-User Experience Virtualization-Agent Driver {DE29CF61-5EE6-43FF-9AAC-959C4E13CC6C}
Microsoft-User Experience Virtualization-App Agent {1ED6976A-4171-4764-B415-7EA08BC46C51}
Microsoft-User Experience Virtualization-IPC {21D79DB0-8E03-41CD-9589-F3EF7001A92A}
Microsoft-User Experience Virtualization-SQM Uploader {57003E21-269B-4BDC-8434-B3BF8D57D2D5}
Microsoft-Windows Networking VPN Plugin Platform {E5FC4A0F-7198-492F-9B0F-88FDCBFDED48}
Microsoft-Windows-AAD {4DE9BC9C-B27A-43C9-8994-0915F1A5E24F}
Microsoft-Windows-ACL-UI {EA4CC8B8-A150-47A3-AFB9-C8D194B19452}
```

The command completed successfully.

- Windows 10 includes more than 1,000 built-in providers.
 - Moreover, Third-Party Software often incorporates its own ETW providers, especially those operating in Kernel mode.
- Due to the high number of providers, it's usually advantageous to filter them using `findstr`.
 - For instance, you will see multiple results for "Winlogon" in the given example.

```
C:\Tools> logman.exe query providers | findstr "Winlogon"
Microsoft-Windows-Winlogon {DBE9B383-7CF3-4331-91CC-A3CB16A3B538}
Windows Winlogon Trace {D451642C-63A6-11D7-9720-00B0D03E0347}
```

- By specifying a provider with Logman, we gain a deeper understanding of the provider's function.
 - This will inform us about the Keywords we can filter on, the available event levels, and which processes are currently utilizing the provider.

```
C:\Tools> logman.exe query providers Microsoft-Windows-Winlogon
```

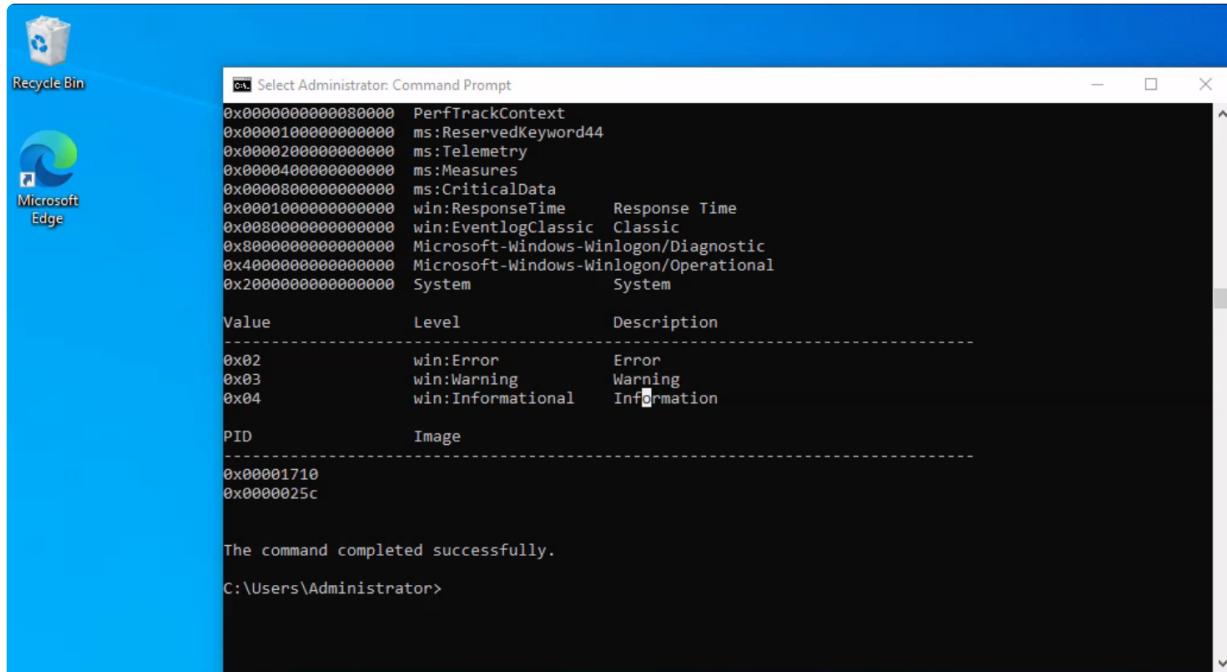
Provider	GUID	
Microsoft-Windows-Winlogon A3CB16A3B538}	{DBE9B383-7CF3-4331-91CC-	
Value	Keyword	Description
0x00000000000010000	PerfInstrumentation	
0x00000000000020000	PerfDiagnostics	
0x00000000000040000	NotificationEvents	
0x00000000000080000	PerfTrackContext	
0x00001000000000000	ms:ReservedKeyword44	
0x00002000000000000	ms:Telemetry	
0x00004000000000000	ms:Measures	
0x00008000000000000	ms:CriticalData	
0x00010000000000000	win:ResponseTime	Response Time
0x00800000000000000	win:EventlogClassic	Classic
0x80000000000000000	Microsoft-Windows-Winlogon/Diagnostic	
0x40000000000000000	Microsoft-Windows-Winlogon/Operational	
0x20000000000000000	System	System
Value	Level	Description
0x02	win:Error	Error
0x03	win:Warning	Warning
0x04	win:Informational	Information
PID	Image	
0x00001710		
0x0000025c		

The command completed successfully.

- The Microsoft-Windows-Winlogon/Diagnostic and Microsoft-Windows-Winlogon/Operational keywords reference the event logs generated from this provider.
- GUI-based alternatives also exist. These are:
 1. Using the graphical interface of the Performance Monitor tool, we can visualize

various running trace sessions.

- A detailed overview of a specific trace can be accessed simply by double-clicking on it.
- This reveals all pertinent data related to the trace, from the engaged providers and their activated features to the nature of the trace itself.
- Additionally, these sessions can be modified to suit our needs by incorporating or eliminating providers.
- Lastly, we can devise new sessions by opting for the "User Defined" category.



```
Administrator: Command Prompt
0x0000000000080000  PerfTrackContext
0x0000100000000000  ms:ReservedKeyword44
0x0002000000000000  ms:Telemetry
0x0004000000000000  ms:Measures
0x0008000000000000  ms:CriticalData
0x0010000000000000  win:ResponseTime      Response Time
0x0080000000000000  win:EventlogClassic  Classic
0x8000000000000000  Microsoft-Windows-Winlogon/Diagnostic
0x4000000000000000  Microsoft-Windows-Winlogon/Operational
0x2000000000000000  System                System

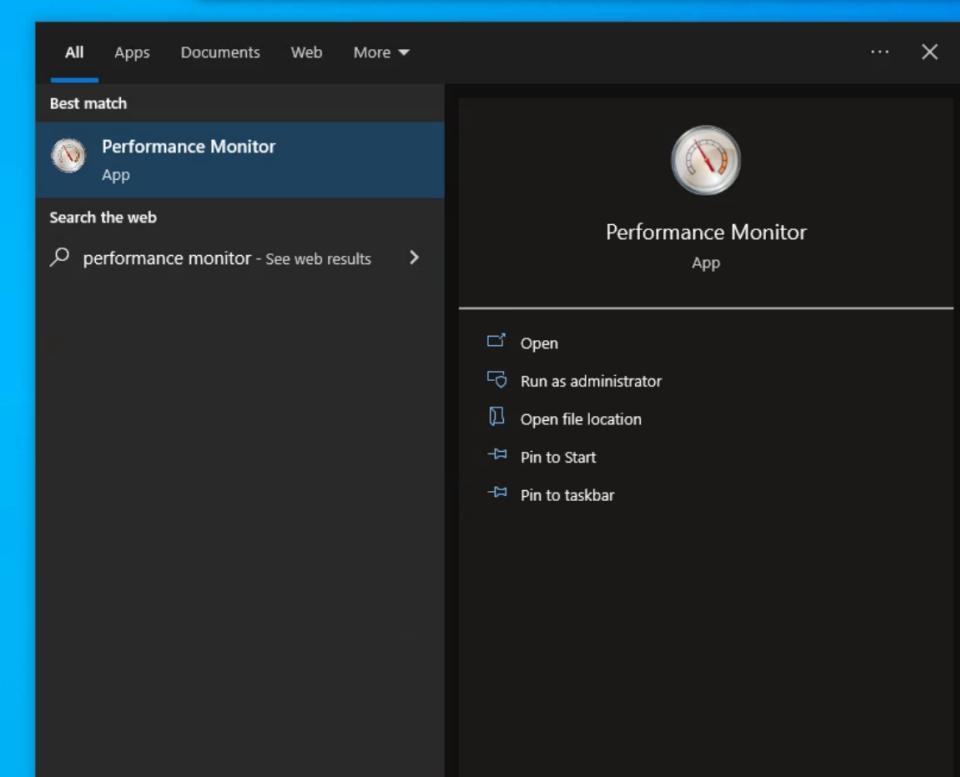
Value          Level       Description
-----
0x02           win:Error    Error
0x03           win:Warning  Warning
0x04           win:Informational  Information

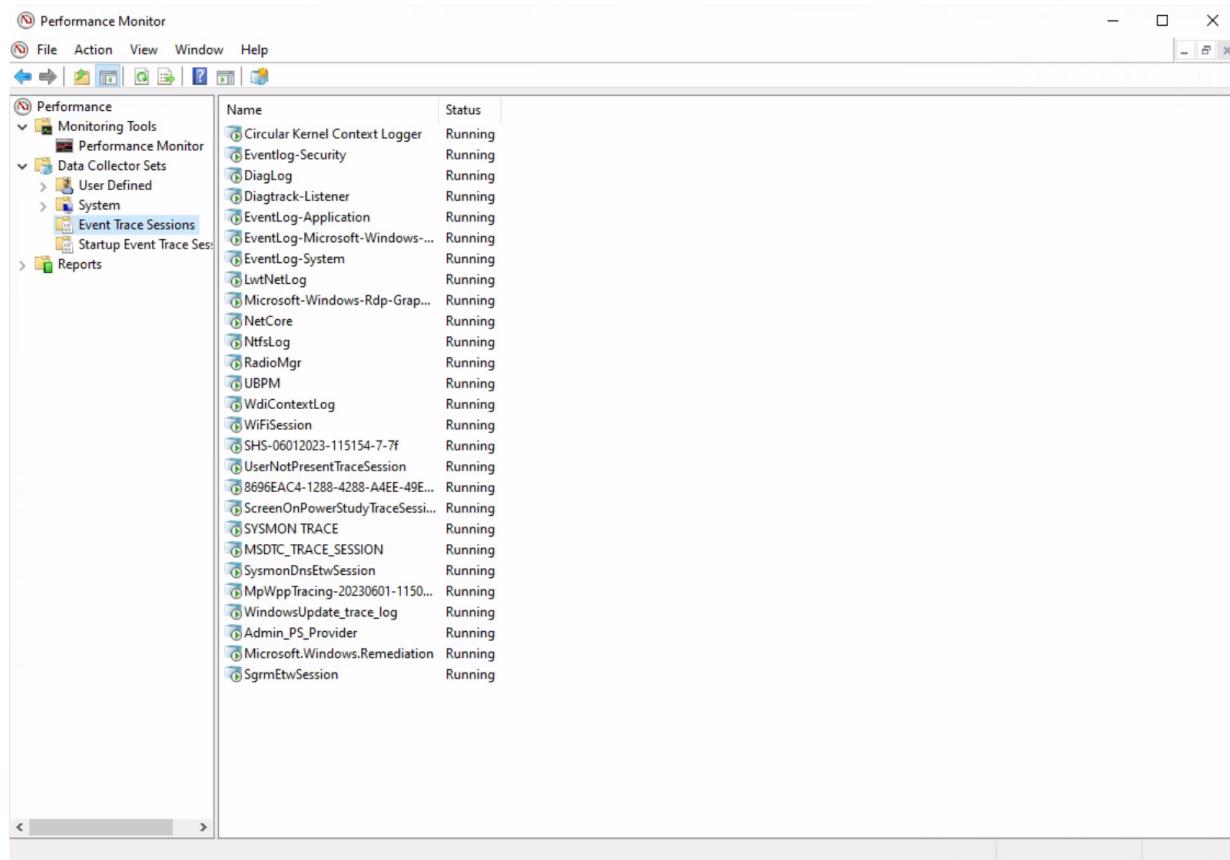
PID            Image

-----
0x0001710
0x0000025c

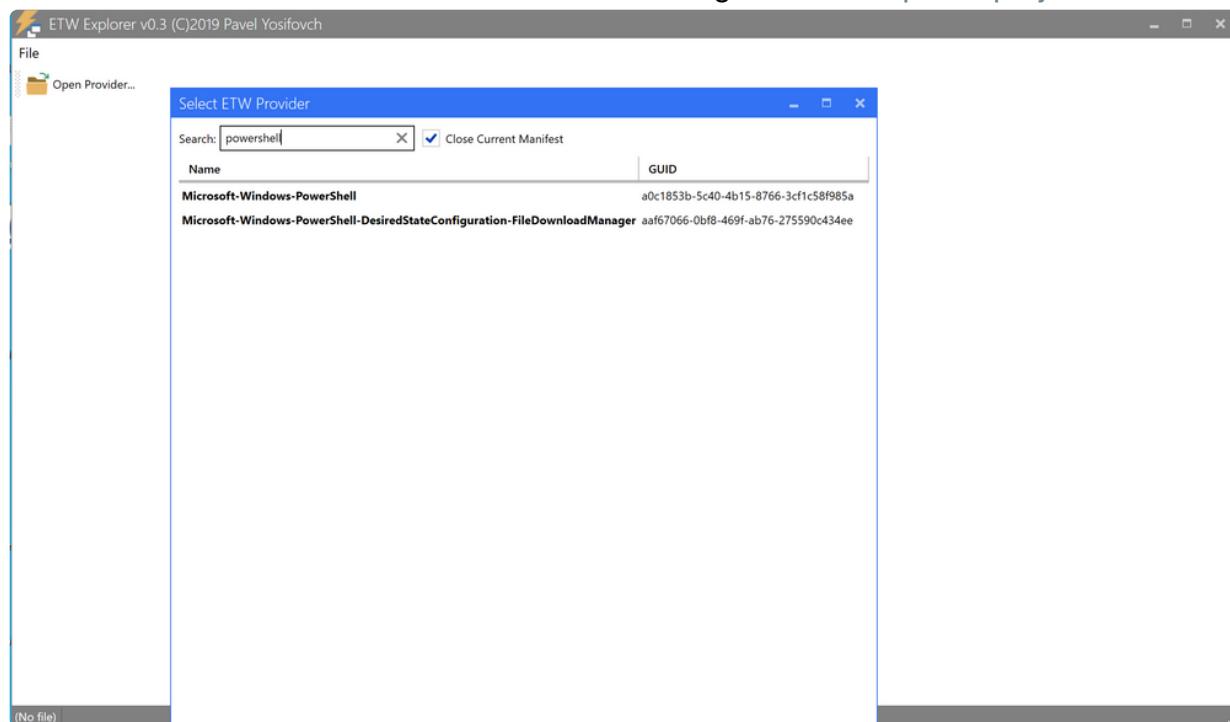
The command completed successfully.

C:\Users\Administrator>
```





2. ETW Provider metadata can also be viewed through the EtwExplorer project.



Useful Providers

- **Microsoft-Windows-Kernel-Process** : This ETW provider is instrumental in monitoring process-related activity within the Windows kernel.
 - It can aid in detecting unusual process behaviors such as process injection, process hollowing, and other tactics commonly used by malware and advanced persistent threats (APTs).
- **Microsoft-Windows-Kernel-File** : As the name suggests, this provider focuses on file-related operations.
 - It can be employed for detection scenarios involving unauthorized file access, changes to critical system files, or suspicious file operations indicative of exfiltration or ransomware activity.
- **Microsoft-Windows-Kernel-Network** : This ETW provider offers visibility into network-related activity at the kernel level.
 - It's especially useful in detecting network-based attacks such as data exfiltration, unauthorized network connections, and potential signs of command and control (C2) communication.
- **Microsoft-Windows-SMBClient/SMBServer** : These providers monitor Server Message Block (SMB) client and server activity, providing insights into file sharing and network communication.
 - They can be used to detect unusual SMB traffic patterns, potentially indicating lateral movement or data exfiltration.
- **Microsoft-Windows-DotNETRuntime** : This provider focuses on .NET runtime events, making it ideal for identifying anomalies in .NET application execution, potential exploitation of .NET vulnerabilities, or malicious .NET assembly loading.
- **OpenSSH** : Monitoring the OpenSSH ETW provider can provide important insights into Secure Shell (SSH) connection attempts, successful and failed authentications, and potential brute force attacks.
- **Microsoft-Windows-VPN-Client** : This provider enables tracking of Virtual Private Network (VPN) client events.
 - It can be useful for identifying unauthorized or suspicious VPN connections.
- **Microsoft-Windows-PowerShell** : This ETW provider tracks PowerShell execution and command activity, making it invaluable for detecting suspicious PowerShell usage, script block logging, and potential misuse or exploitation.
- **Microsoft-Windows-Kernel-Registry** : This provider monitors registry operations, making it useful for detection scenarios related to changes in registry keys, often associated with persistence mechanisms, malware installation, or system configuration changes.
- **Microsoft-Windows-CodeIntegrity** : This provider monitors code and driver integrity checks, which can be key in identifying attempts to load unsigned or malicious drivers or code.

- **Microsoft-Antimalware-Service** : This ETW provider can be employed to detect potential issues with the antimalware service, including disabled services, configuration changes, or potential evasion techniques employed by malware.
- **WinRM** : Monitoring the Windows Remote Management (WinRM) provider can reveal unauthorized or suspicious remote management activity, often indicative of lateral movement or remote command execution.
- **Microsoft-Windows-TerminalServices-LocalSessionManager** : This provider tracks local Terminal Services sessions, making it useful for detecting unauthorized or suspicious remote desktop activity.
- **Microsoft-Windows-Security-Mitigations** : This provider keeps tabs on the effectiveness and operations of security mitigations in place. It's essential for identifying potential bypass attempts of these security controls.
- **Microsoft-Windows-DNS-Client** : This ETW provider gives visibility into DNS client activity, which is crucial for detecting DNS-based attacks, including DNS tunneling or unusual DNS requests that may indicate C2 communication.
- **Microsoft-Antimalware-Protection** : This provider monitors the operations of antimalware protection mechanisms. It can be used to detect any issues with these mechanisms, such as disabled protection features, configuration changes, or signs of evasion techniques employed by malicious actors.

Restricted Providers

- In the realm of Windows operating system security, certain ETW providers are considered "restricted."
 - These providers offer valuable telemetry but are only accessible to processes that carry the requisite permissions.
 - This exclusivity is designed to ensure that sensitive system data remains shielded from potential threats.
- One of these high-value, restricted providers is **Microsoft-Windows-Threat-Intelligence**.
 - This provider offers crucial insights into potential security threats and is often leveraged in Digital Forensics and Incident Response (DFIR) operations.
 - However, to access this provider, processes must be privileged with a specific right, known as Protected Process Light (PPL).
- According to Elastic: *To be able to run as a PPL, an anti-malware vendor must apply to Microsoft, prove their identity, sign binding legal documents, implement an Early Launch Anti-Malware (ELAM) driver, run it through a test suite, and submit it to Microsoft for a special Authenticode signature. It is not a trivial process. Once this process is complete, the vendor can use this ELAM driver to have Windows protect*

their anti-malware service by running it as a PPL. With that said, workarounds to access the Microsoft-Windows-Threat-Intelligence provider exist.

- In the context of Microsoft-Windows-Threat-Intelligence, the benefits of this privileged access are manifold.
 - This provider can record highly granular data about potential threats, enabling security professionals to detect and analyze sophisticated attacks that may have eluded other defenses.
 - Its telemetry can serve as vital evidence in forensic investigations, revealing details about the origin of a threat, the systems and data it interacted with, and the alterations it made.
 - Moreover, by monitoring this provider in real-time, security teams can potentially identify ongoing threats and intervene to mitigate damage.
- In the next section, we will utilize ETW to investigate attacks that may evade detection if we rely solely on Sysmon for monitoring and analysis, due to its inherent limitations in capturing certain events.

Tapping Into ETW

Detection Example 1: Detecting Strange Parent-Child Relationships

- Abnormal parent-child relationships among processes can be indicative of malicious activities.
 - In standard Windows environments, certain processes never call or spawn others.
 - For example, it is highly unlikely to see "calc.exe" spawning "cmd.exe" in a normal Windows environment.
 - Understanding these typical parent-child relationships can assist in detecting anomalies.
 - Samir Bousseaden has shared an insightful mind map introducing common parent-child relationships, which can be referenced [here](#).
- By utilizing Process Hacker, we can explore parent-child relationships within Windows.
 - Sorting the processes by dropdowns in the Processes view reveals a hierarchical

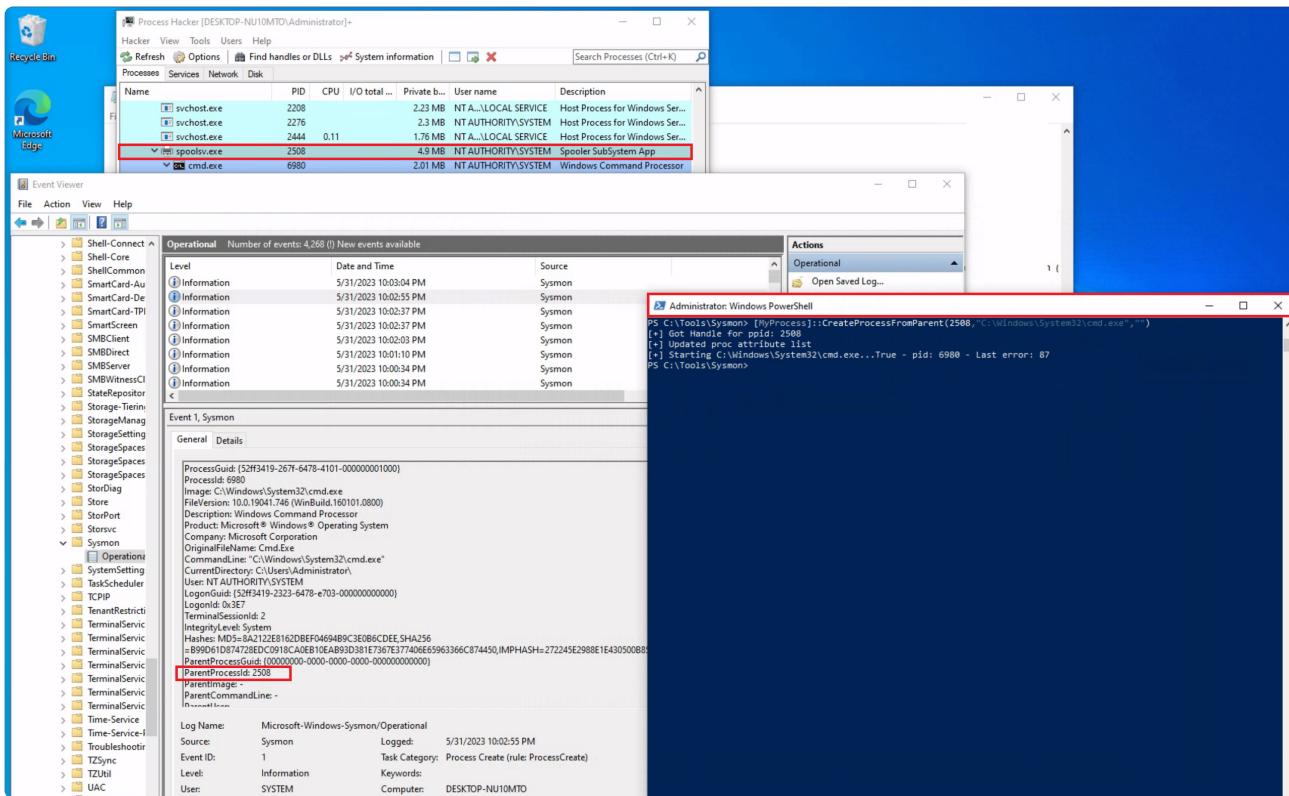
representation of the relationships.

Name	PID	1.7...	32.64 k...	1.91 GB	Private b...	User name	Description
System Idle Process	98.24			60 kB	NT AUTHORITY\SYSTEM		
System	4	0.15		196 kB	NT AUTHORITY\SYSTEM	NT Kernel & System	
Registry	112			9.55 MB	NT AUTHORITY\SYSTEM		
smss.exe	340			1.04 MB	NT AUTHORITY\SYSTEM	Windows Session Manager	
Memory Compression	2456			336 kB	NT AUTHORITY\SYSTEM		
Interrupts	0.14			0		Interrupts and DPCs	
csrss.exe	436			1.66 MB	NT AUTHORITY\SYSTEM	Client Server Runtime Process	
wininit.exe	512			1.32 MB	NT AUTHORITY\SYSTEM	Windows Start-Up Application	
services.exe	656	0.26		5.61 MB	NT AUTHORITY\SYSTEM	Services and Controller app	
svchost.exe	812			11.16 MB	NT AUTHORITY\SYSTEM	Host Process for Windows Ser...	
dllhost.exe	7000			6.42 MB	DESKTOP-R4PEEIF\waldo	COM Surrogate	
StartMenuExperie...	5440			24.54 MB	DESKTOP-R4PEEIF\waldo		
RuntimeBroker.exe	3672			7 MB	DESKTOP-R4PEEIF\waldo	Runtime Broker	
SearchApp.exe	1728			137.36 MB	DESKTOP-R4PEEIF\waldo	Search application	
RuntimeBroker.exe	4372			24.95 MB	DESKTOP-R4PEEIF\waldo	Runtime Broker	
SearchApp.exe	5256			16.02 MB	DESKTOP-R4PEEIF\waldo	Search application	
ShellExperienceH...	1436			16.19 MB	DESKTOP-R4PEEIF\waldo	Windows Shell Experience Host	
RuntimeBroker.exe	6968			5.9 MB	DESKTOP-R4PEEIF\waldo	Runtime Broker	
RuntimeBroker.exe	8848			2.86 MB	DESKTOP-R4PEEIF\waldo	Runtime Broker	
TextInputHost.exe	3720			9.51 MB	DESKTOP-R4PEEIF\waldo		
dllhost.exe	3080			4.74 MB	DESKTOP-R4PEEIF\waldo	COM Surrogate	
ApplicationFrame...	7844			5.91 MB	DESKTOP-R4PEEIF\waldo	Application Frame Host	
unsecapp.exe	1908			1.27 MB	NT AUTHORITY\SYSTEM	Sink to receive asynchronous ...	
Microsoft.Photos....	244			22.83 MB	DESKTOP-R4PEEIF\waldo		
RuntimeBroker.exe	8372			2.34 MB	DESKTOP-R4PEEIF\waldo	Runtime Broker	
WmiPrvSE.exe	6316			9.72 MB	NT AUTHORITY\SYSTEM	WMI Provider Host	
UserOOBEBroker....	2052			1.82 MB	NT AUTHORITY\SYSTEM	User OOBE Broker	
TiWorker.exe	5124			66.48 MB	NT AUTHORITY\SYSTEM	Windows Modules Installer W...	
svchost.exe	936			8.73 MB	N...\\NETWORK SERVICE	Host Process for Windows Ser...	
svchost.exe	980			2.92 MB	NT AUTHORITY\SYSTEM	Host Process for Windows Ser...	
svchost.exe	524			996 kB	NT AUTHORITY\SYSTEM	Host Process for Windows Ser...	
svchost.exe	392			1.55 MB	NT A...\\LOCAL SERVICE	Host Process for Windows Ser...	
...	1002			1.26 MB	NT A...\\LOCAL SERVICE		

- Analyzing these relationships in standard and custom environments enables us to identify deviations from normal patterns.
 - For example, if we observe the "spoolsv.exe" process creating "whoami.exe" instead of its expected behavior of creating a "conhost", it raises suspicion.
- To showcase a strange parent-child relationship, where "cmd.exe" appears to be created by "spoolsv.exe" with no accompanying arguments, we will utilize an attacking technique called Parent PID Spoofing.
 - Parent PID Spoofing can be executed through the psgetsystem project in the following manner.

```
PS C:\Tools\psgetsystem> powershell -ep bypass
PS C:\Tools\psgetsystem> Import-Module .\psgetsys.ps1
PS C:\Tools\psgetsystem> [MyProcess]::CreateProcessFromParent([Process ID of spoolsv.exe],"C:\Windows\System32\cmd.exe","")
```

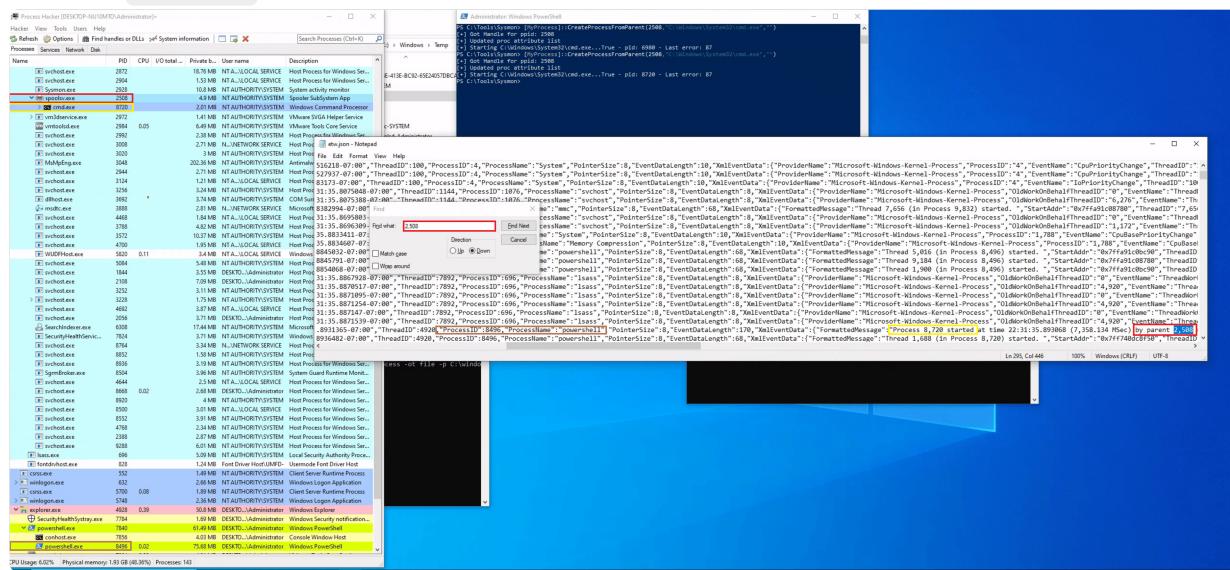
```
[MyProcess]::CreateProcessFromParent(2388, "C:\Windows\System32\cmd.exe",
""")
```



- Due to the parent PID spoofing technique we employed, Sysmon Event 1 incorrectly displays `spoolsv.exe` as the parent of `cmd.exe`.
 - However, it was actually `powershell.exe` that created `cmd.exe`.
- As we have previously discussed, although Sysmon and event logs provide valuable telemetry for hunting and creating alert rules, they are not the only sources of information.
 - Let's begin by collecting data from the `Microsoft-Windows-Kernel-Process` provider using `SilkETW` (the provider can be identified using `logman` as we described previously, `logman.exe query providers | findstr "Process"`).
 - After that, we can proceed to simulate the attack again to assess whether ETW can provide us with more accurate information regarding the execution of `cmd.exe`.

```
c:\Tools\SilkETW_SilkService_v8\v8\SilkETW>SilkETW.exe -t user -pn Microsoft-Windows-Kernel-Process -ot file -p C:\windows\temp\etw.json
```

- The `etw.json` file (that includes data from the Microsoft-Windows-Kernel-Process provider) seems to contain information about `powershell.exe` being the one who created `cmd.exe`.



- It should be noted that SilkETW event logs can be ingested and viewed by Windows Event Viewer through `SilkService` to provide us with deeper and more extensive visibility into the actions performed on a system.

Detection Example 2: Detecting Malicious .NET Assembly Loading

- Traditionally, adversaries employed a strategy known as "Living off the Land" (LotL), exploiting legitimate system tools, such as PowerShell, to carry out their malicious operations.
 - This approach reduces the risk of detection since it involves the use of tools that are native to the system, and therefore less likely to raise suspicion.
- However, the cybersecurity community has adapted and developed countermeasures against this strategy.
- Responding to these defensive advancements, attackers have developed a new approach that Mandiant labels as "Bring Your Own Land" (BYOL).
 - Instead of relying on the tools already present on a victim's system, threat actors and penetration testers emulating these tactics now employ .NET assemblies executed entirely in memory.
 - This involves creating custom-built tools using languages like C#, rendering them independent of the pre-existing tools on the target system. The "Bring Your Own Land" lands is quite effective for the following reasons:
 - Each Windows system comes equipped with a certain version of .NET pre-installed by default.

- A salient feature of .NET is its managed nature, alleviating the need for programmers to manually handle memory management.
 - This attribute is part of the framework's managed code execution process, where the Common Language Runtime (CLR) takes responsibility for key system-level operations such as garbage collection, eliminating memory leaks and ensuring more efficient resource utilization.
- One of the intriguing advantages of using .NET assemblies is their ability to be loaded directly into memory.
 - This means that an executable or DLL does not need to be written physically to the disk - instead, it is executed directly in memory.
 - This behavior minimizes the artifacts left behind on the system and can help bypass some forms of detection that rely on inspecting files written to disk.
- Microsoft has integrated a wide range of libraries into the .NET framework to address numerous common programming challenges.
 - These libraries include functionalities for establishing HTTP connections, implementing cryptographic operations, and enabling inter-process communication (IPC), such as named pipes.
 - These pre-built tools streamline the development process, reduce the likelihood of errors, and make it easier to build robust and efficient applications.
 - Furthermore, for a threat actor, these rich features provide a toolkit for creating more sophisticated and covert attack methods.
- A powerful illustration of this BYOL strategy is the "execute-assembly" command implemented in CobaltStrike, a widely-used software platform for Adversary Simulations and Red Team Operations.
 - CobaltStrike's 'execute-assembly' command allows the user to execute .NET assemblies directly from memory, making it an ideal tool for implementing a BYOL strategy.
- In a manner akin to how we detected the execution of unmanaged PowerShell scripts through the observation of anomalous `clr.dll` and `clrjit.dll` loading activity in processes that ordinarily wouldn't require them, we can employ a similar approach to identify malicious .NET assembly loading.
 - This is achieved by scrutinizing the activity related to the loading of .NET-associated DLLs, specifically `clr.dll` and `mscoree.dll`
- Monitoring the loading of such libraries can help reveal attempts to execute .NET assemblies in unusual or unexpected contexts, which can be a sign of malicious activity.

- This type of DLL loading behavior can often be detected by leveraging Sysmon's Event ID 7, which corresponds to "Image Loaded" events.
 - For demonstrative purposes, let's emulate a malicious .NET assembly load by executing a precompiled version of **Seatbelt** that resides on disk.
 - **Seatbelt** is a well-known .NET assembly, often employed by adversaries who load and execute it in memory to gain situational awareness on a compromised system.

```
PS C:\Tools\GhostPack Compiled Binaries>.\Seatbelt.exe TokenPrivileges
```

===== TokenPrivileges =====

Current Token's Privileges

```
SeIncreaseQuotaPrivilege: DISABLED  
SeSecurityPrivilege: DISABLED  
SeTakeOwnershipPrivilege: DISABLED  
SeLoadDriverPrivilege: DTSAFE
```

```
        SeSystemProfilePrivilege: DISABLED
        SeSystemtimePrivilege: DISABLED
SeProfileSingleProcessPrivilege: DISABLED
SeIncreaseBasePriorityPrivilege: DISABLED
        SeCreatePagefilePrivilege: DISABLED
        SeBackupPrivilege: DISABLED
        SeRestorePrivilege: DISABLED
        SeShutdownPrivilege: DISABLED
        SeDebugPrivilege: SE_PRIVILEGE_ENABLED
SeSystemEnvironmentPrivilege: DISABLED
        SeChangeNotifyPrivilege:
SE_PRIVILEGE_ENABLED_BY_DEFAULT, SE_PRIVILEGE_ENABLED
        SeRemoteShutdownPrivilege: DISABLED
        SeUndockPrivilege: DISABLED
        SeManageVolumePrivilege: DISABLED
        SeImpersonatePrivilege:
SE_PRIVILEGE_ENABLED_BY_DEFAULT, SE_PRIVILEGE_ENABLED
        SeCreateGlobalPrivilege:
SE_PRIVILEGE_ENABLED_BY_DEFAULT, SE_PRIVILEGE_ENABLED
        SeIncreaseWorkingSetPrivilege: DISABLED
        SeTimeZonePrivilege: DISABLED
        SeCreateSymbolicLinkPrivilege: DISABLED
SeDelegateSessionUserImpersonatePrivilege: DISABLED
```

- Assuming we have Sysmon configured appropriately to log image loading events (Event ID 7), executing 'Seatbelt.exe' would trigger the loading of key .NET-related DLLs such as 'clr.dll' and 'mscoree.dll'.
 - Sysmon, keenly observing system activities, will log these DLL load operations as Event ID 7 records.

Event 7, Sysmon

General Details

```
Image loaded:  
RuleName: -  
UtcTime: 2023-06-05 20:57:05.530  
ProcessGuid: {52ff3419-4c21-647e-5f01-000000001000}  
ProcessId: 6080  
Image: C:\Tools\GhostPack Compiled Binaries\Seatbelt.exe  
ImageLoaded: C:\Windows\System32\mscoree.dll  
FileVersion: 10.0.19041.1 (WinBuild.160101.0800)  
Description: Microsoft .NET Runtime Execution Engine  
Product: Microsoft® Windows® Operating System  
Company: Microsoft Corporation  
OriginalFileName: mscoree.dll  
Hashes: MD5=D5971EF71DE1BDD46D537203ABFCC756,SHA256=8828DE042D008783BA5B31C82935A3ED38D5996927C3399B3E1FC6FE723FC84E,IMPHASH=65F23EFA1EB51A5DAA8399BFAA840074  
Signed: true  
Signature: Microsoft Windows  
SignatureStatus: Valid
```

Log Name: Microsoft-Windows-Sysmon/Operational
Source: Sysmon Logged: 6/5/2023 1:57:05 PM
Event ID: 7 Task Category: Image loaded (rule: ImageLoad)
Level: Information Keywords:
User: SYSTEM Computer: DESKTOP-NU10MTO
OpCode: Info
More Information: [Event Log Online Help](#)

Event 7, Sysmon

General Details

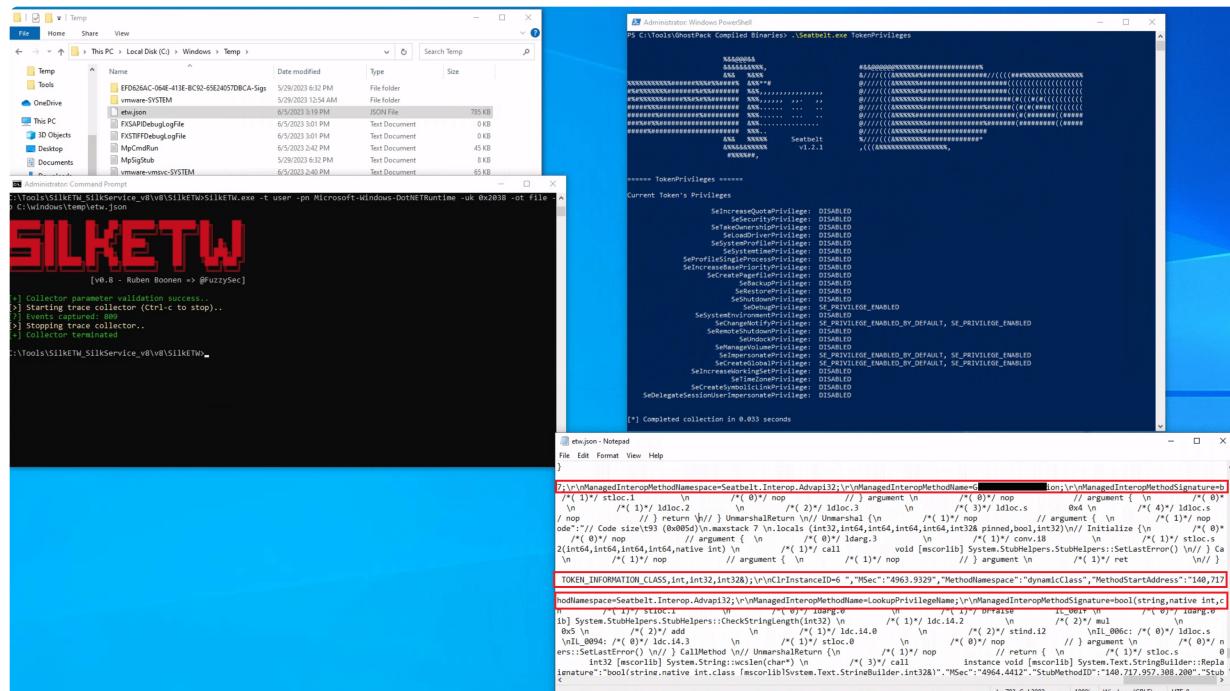
```
Image loaded:  
RuleName: -  
UtcTime: 2023-06-05 20:57:05.544  
ProcessGuid: {52ff3419-4c21-647e-5f01-000000001000}  
ProcessId: 6080  
Image: C:\Tools\GhostPack Compiled Binaries\Seatbelt.exe  
ImageLoaded: C:\Windows\Microsoft.NET\Framework64\v4.0.30319\clr.dll  
FileVersion: 4.8.4515.0 built by: NET48REL1LAST_C  
Description: Microsoft .NET Runtime Common Language Runtime - WorkStation  
Product: Microsoft® .NET Framework  
Company: Microsoft Corporation  
OriginalFileName: clr.dll  
Hashes: MD5=2B0E5597FF51A3A4D5BB2DDAB0214531,SHA256=8D09CE35C987EADCF01686BB559920951B0116985FE4FEB5A488A6A8F7C4BDB9,IMPHASH=259C196C67C4E02F941CAD54D9D9BB8A  
Signed: true  
Signature: Microsoft Corporation  
SignatureStatus: Valid
```

Log Name: Microsoft-Windows-Sysmon/Operational
Source: Sysmon Logged: 6/5/2023 1:57:05 PM
Event ID: 7 Task Category: Image loaded (rule: ImageLoad)
Level: Information Keywords:
User: SYSTEM Computer: DESKTOP-NU10MTO
OpCode: Info
More Information: [Event Log Online Help](#)

- As already mentioned, relying solely on Sysmon Event ID 7 for detecting attacks can be challenging due to the large volume of events it generates (especially if not configured properly).
 - Additionally, while it informs us about the DLLs being loaded, it doesn't provide granular details about the actual content of the loaded .NET assembly.
 - To augment our visibility and gain deeper insights into the actual assembly being loaded, we can again leverage Event Tracing for Windows (ETW) and specifically the `Microsoft-Windows-DotNETRuntime` provider.
 - Let's use SilkETW to collect data from the `Microsoft-Windows-DotNETRuntime` provider.
 - After that, we can proceed to simulate the attack again to evaluate whether ETW can furnish us with more detailed and actionable intelligence regarding the loading and execution of the 'Seatbelt' .NET assembly.

```
c:\Tools\SilkETW_SilkService_v8\v8\SilkETW>SilkETW.exe -t user -pn Microsoft-Windows-DotNETRuntime -uk 0x2038 -ot file -p C:\windows\temp\etw.json
```

- The `etw.json` file (that includes data from the `Microsoft-Windows-DotNETRuntime` provider) seems to contain a wealth of information about the loaded assembly, including method names.



- It's worth noting that in our current SilkETW configuration, we're not capturing the entirety of events from the "Microsoft-Windows-DotNETRuntime" provider.

- Instead, we're selectively targeting a specific subset (indicated by `0x2038`), which includes: `JitKeyword`, `InteropKeyword`, `LoaderKeyword`, and `NGenKeyword`.
 - The `JitKeyword` relates to the Just-In-Time (JIT) compilation events, providing information on the methods being compiled at runtime.
 - This could be particularly useful for understanding the execution flow of the .NET assembly.
 - The `InteropKeyword` refers to Interoperability events, which come into play when managed code interacts with unmanaged code.
 - These events could provide insights into potential interactions with native APIs or other unmanaged components.
 - `LoaderKeyword` events provide details on the assembly loading process within the .NET runtime, which can be vital for understanding what .NET assemblies are being loaded and potentially executed.
 - Lastly, the `NGenKeyword` corresponds to Native Image Generator (NGen) events, which are concerned with the creation and usage of precompiled .NET assemblies.
 - Monitoring these could help detect scenarios where attackers use precompiled .NET assemblies to evade JIT-related detections.
- This [blog post](#) provides valuable perspectives on SilkETW as well as the identification of malware based on .NET.

Analyzing Windows Event Logs En Masse

Get-WinEvent

Overview

- Understanding the importance of mass analysis of Windows Event Logs and Sysmon logs is pivotal in the realm of cybersecurity, especially in Incident Response (IR) and threat hunting scenarios.
 - These logs hold invaluable information about the state of your systems, user activities, potential threats, system changes, and troubleshooting information.
 - However, these logs can also be voluminous and unwieldy. For large-scale organizations, it's not uncommon to generate millions of logs each day. Hence, to distill useful information from these logs, we require efficient tools and techniques to analyze these logs en masse.
- One of these tools is the the [Get-WinEvent cmdlet](#) in PowerShell.

Using Get-WinEvent

- The `Get-WinEvent` cmdlet is an indispensable tool in PowerShell for querying Windows Event logs en masse.
 - The cmdlet provides us with the capability to retrieve different types of event logs, including classic Windows event logs like System and Application logs, logs generated by Windows Event Log technology, and Event Tracing for Windows (ETW) logs.
- To quickly identify the available logs, we can leverage the `-ListLog` parameter in conjunction with the `Get-WinEvent` cmdlet.
 - By specifying `*` as the parameter value, we retrieve all logs without applying any filtering criteria.
 - This allows us to obtain a comprehensive list of logs and their associated properties.
 - By executing the following command, we can retrieve the list of logs and display essential properties such as `LogName`, `RecordCount`, `IsClassicLog`, `.IsEnabled`, `LogMode`, and `LogType`.
 - The `|` character is a pipe operator. It is used to pass the output of one command (in this case, the `Get-WinEvent` command) to another command (in this case, the `Select-Object` command).

```
PS C:\Users\Administrator> Get-WinEvent -ListLog * | Select-Object
LogName, RecordCount, IsClassicLog, isEnabled, LogMode, LogType |
Format-Table -AutoSize
```

LogName	RecordCount	IsClassicLog	isEnabled	LogMode	LogType
Windows PowerShell					
2916	True	True	Circular	Administrative	
System					
1786	True	True	Circular	Administrative	
Security					
8968	True	True	Circular	Administrative	
Key Management Service					
0	True	True	Circular	Administrative	
Internet Explorer					
0	True	True	Circular	Administrative	
HardwareEvents					
0	True	True	Circular	Administrative	
Application					
2079	True	True	Circular	Administrative	
Windows Networking	Vpn	Plugin	Platform/Operational	Verbose	

False False Circular Operational
Windows Networking Vpn Plugin Platform/Operational
False False Circular Operational
SMSApi
0 False True Circular Operational
Setup
16 False True Circular Operational
OpenSSH/Operational
0 False True Circular Operational
OpenSSH/Admin
0 False True Circular Administrative
Network Isolation Operational
False False Circular Operational
Microsoft-WindowsPhone-Connectivity-WiFiConnSvc-Channel
0 False True Circular Operational
Microsoft-Windows-WWAN-SVC-Events/Operational
0 False True Circular Operational
Microsoft-Windows-WPD-MTPClassDriver/Operational
0 False True Circular Operational
Microsoft-Windows-WPD-CompositeClassDriver/Operational
0 False True Circular Operational
Microsoft-Windows-WPD-ClassInstaller/Operational
0 False True Circular Operational
Microsoft-Windows-Workplace Join/Admin
0 False True Circular Administrative
Microsoft-Windows-WorkFolders/WHC
0 False True Circular Operational
Microsoft-Windows-WorkFolders/Operational
0 False True Circular Operational
Microsoft-Windows-Wordpad/Admin
False False Circular Operational
Microsoft-Windows-WMPNSS-Service/Operational
0 False True Circular Operational
Microsoft-Windows-WMI-Activity/Operational
895 False True Circular Operational
Microsoft-Windows-wmbclass/Trace
False False Circular Operational
Microsoft-Windows-WLAN-AutoConfig/Operational
0 False True Circular Operational
Microsoft-Windows-Wired-AutoConfig/Operational
0 False True Circular Operational
Microsoft-Windows-Winsock-WS2HELP/Operational
0 False True Circular Operational
Microsoft-Windows-Winsock-NameResolution/Operational
False False Circular Operational
Microsoft-Windows-Winsock-AFD/Operational

```
False      False Circular    Operational
Microsoft-Windows-WinRM/Operational
230       False      True Circular    Operational
Microsoft-Windows-WinNat/Oper
False      False Circular    Operational
Microsoft-Windows-Winlogon/Operational
648       False      True Circular    Operational
Microsoft-Windows-WinINet-Config/ProxyConfigChanged
2         False      True Circular    Operational
--- SNIP ---
```

- This command provides us with valuable information about each log, including the name of the log, the number of records present, whether the log is in the classic `.evt` format or the newer `.evtx` format, its enabled status, the log mode (Circular, Retain, or AutoBackup), and the log type (Administrative, Analytical, Debug, or Operational).
- Additionally, we can explore the event log providers associated with each log using the `-ListProvider` parameter.
 - Event log providers serve as the sources of events within the logs. Executing the following command allows us to retrieve the list of providers and their respective linked logs.
- This command provides us with an overview of the available providers and their associations with specific logs.
 - It enables us to identify providers of interest for filtering purposes.
- Now, let's focus on retrieving specific event logs using the `Get-WinEvent` cmdlet.
 - At its most basic, `Get-WinEvent` retrieves event logs from local or remote computers.
 - The examples below demonstrate how to retrieve events from various logs.

1. Retrieving events from the System log

```
PS C:\Users\Administrator> Get-WinEvent -LogName 'System' -MaxEvents 50
| Select-Object TimeCreated, ID, ProviderName, LevelDisplayName, Message
| Format-Table -AutoSize
```

```
Get-WinEvent -LogName 'System' -MaxEvents 50 | Select-Object
TimeCreated, ID, ProviderName, LevelDisplayName, Message | Format-Table
-AutoSize
```

TimeCreated	ID	ProviderName
LevelDisplayName	Message	
-----	-----	-----

6/2/2023 9:41:42 AM 16 Microsoft-Windows-Kernel-General
Information The access history in hive \??
\C:\Users\Administrator\AppData\Local\Packages\MicrosoftWindows.Client.C
BS_cw5...

6/2/2023 9:38:32 AM 16 Microsoft-Windows-Kernel-General
Information The access history in hive \??
\C:\Users\Administrator\AppData\Local\Packages\Microsoft.Windows.ShellEx
perien...

6/2/2023 9:38:32 AM 10016 Microsoft-Windows-DistributedCOM
Warning The machine-default permission settings do not grant
Local Activation permission for the COM Server applicat...

6/2/2023 9:37:31 AM 16 Microsoft-Windows-Kernel-General
Information The access history in hive \??
\C:\Users\Administrator\AppData\Local\Packages\Microsoft.WindowsAlarms_8
wekyb3...

6/2/2023 9:37:31 AM 16 Microsoft-Windows-Kernel-General
Information The access history in hive \??
\C:\Users\Administrator\AppData\Local\Packages\microsoft.windowscommunic
ations...

6/2/2023 9:37:31 AM 16 Microsoft-Windows-Kernel-General
Information The access history in hive \??
\C:\Users\Administrator\AppData\Local\Packages\Microsoft.Windows.Content
Delive...

6/2/2023 9:36:35 AM 16 Microsoft-Windows-Kernel-General
Information The access history in hive \??
\C:\Users\Administrator\AppData\Local\Packages\Microsoft.YourPhone_8weky
b3d8bb...

6/2/2023 9:36:32 AM 16 Microsoft-Windows-Kernel-General
Information The access history in hive \??
\C:\Users\Administrator\AppData\Local\Packages\Microsoft.AAD.BrokerPlugi
n_cw5n...

6/2/2023 9:36:30 AM 16 Microsoft-Windows-Kernel-General
Information The access history in hive \??
\C:\Users\Administrator\AppData\Local\Packages\Microsoft.Windows.Search_
cw5n1h...

6/2/2023 9:36:29 AM 16 Microsoft-Windows-Kernel-General
Information The access history in hive \??
\C:\Users\Administrator\AppData\Local\Packages\Microsoft.Windows.StartMe
nuExpe...

6/2/2023 9:36:14 AM 16 Microsoft-Windows-Kernel-General
Information The access history in hive \??
\C:\Users\Administrator\AppData\Local\Microsoft\Windows\UsrClass.dat was
clear...

6/2/2023 9:36:14 AM 16 Microsoft-Windows-Kernel-General
Information The access history in hive \??

```
\C:\Users\Administrator\ntuser.dat was cleared updating 2366 keys and
creating...
6/2/2023 9:36:14 AM  7001 Microsoft-Windows-Winlogon
Information      User Logon Notification for Customer Experience
Improvement Program
6/2/2023 9:33:04 AM   16 Microsoft-Windows-Kernel-General
Information      The access history in hive \??
\C:\Windows\AppCompat\Programs\Amcache.hve was cleared updating 920 keys
and c...
6/2/2023 9:31:54 AM   16 Microsoft-Windows-Kernel-General
Information      The access history in hive \??
\C:\Windows\ServiceProfiles\NetworkService\AppData\Local\Microsoft\Windo
ws\Del...
6/2/2023 9:30:23 AM   16 Microsoft-Windows-Kernel-General
Information      The access history in hive \??
\C:\Windows\System32\config\COMPONENTS was cleared updating 54860 keys
and cre...
6/2/2023 9:30:16 AM   15 Microsoft-Windows-Kernel-General
Information      Hive \SystemRoot\System32\config\DRIVERS was
reorganized with a starting size of 3956736 bytes and an ending...
6/2/2023 9:30:10 AM  1014 Microsoft-Windows-DNS-Client
Warning          Name resolution for the name settings-
win.data.microsoft.com timed out after none of the configured DNS
serv...
6/2/2023 9:29:54 AM  7026 Service Control Manager
Information      The following boot-start or system-start driver(s) did
not load: ...
6/2/2023 9:29:54 AM 10148 Microsoft-Windows-WinRM
Information      The WinRM service is listening for WS-Management
requests. ...
6/2/2023 9:29:51 AM 51046 Microsoft-Windows-DHCPv6-Client
Information      DHCPv6 client service is started
```

- This example retrieves the first 50 events from the System log. It selects specific properties, including the event's creation time, ID, provider name, level display name, and message.
 - This facilitates easier analysis and troubleshooting.

2. Retrieving events from Microsoft-Windows-WinRM/Operational

```
PS C:\Users\Administrator> Get-WinEvent -LogName 'Microsoft-Windows-
WinRM/Operational' -MaxEvents 30 | Select-Object TimeCreated, ID,
ProviderName, LevelDisplayName, Message | Format-Table -AutoSize
```

```
Get-WinEvent -LogName 'Microsoft-Windows-WinRM/Operational' -MaxEvents 30 | Select-Object TimeCreated, ID, ProviderName, LevelDisplayName, Message | Format-Table -AutoSize
```

TimeCreated	ID	ProviderName	LevelDisplayName
Message			
--	--	--	--
6/2/2023 9:30:15 AM	132	Microsoft-Windows-WinRM Information	WSMan
operation Enumeration completed successfully			
6/2/2023 9:30:15 AM	145	Microsoft-Windows-WinRM Information	WSMan
operation Enumeration started with resourceUri...			
6/2/2023 9:30:15 AM	132	Microsoft-Windows-WinRM Information	WSMan
operation Enumeration completed successfully			
6/2/2023 9:30:15 AM	145	Microsoft-Windows-WinRM Information	WSMan
operation Enumeration started with resourceUri...			
6/2/2023 9:29:54 AM	209	Microsoft-Windows-WinRM Information	The Winrm service started successfully
-- SNIP --			

- In this example, events are retrieved from the Microsoft-Windows-WinRM/Operational log.
 - The command retrieves the first 30 events and selects relevant properties for display, including the event's creation time, ID, provider name, level display name, and message.
- To retrieve the oldest events, instead of manually sorting the results, we can utilize the `-Oldest` parameter with the Get-WinEvent cmdlet.
 - This parameter allows us to retrieve the first events based on their chronological order.
 - The following command demonstrates how to retrieve the oldest 30 events from the 'Microsoft-Windows-WinRM/Operational' log.

```
PS C:\Users\Administrator> Get-WinEvent -LogName 'Microsoft-Windows-WinRM/Operational' -Oldest -MaxEvents 30 | Select-Object TimeCreated, ID, ProviderName, LevelDisplayName, Message | Format-Table -AutoSize
```

```
Get-WinEvent -LogName 'Microsoft-Windows-WinRM/Operational' -Oldest -MaxEvents 30 | Select-Object TimeCreated, ID, ProviderName, LevelDisplayName, Message | Format-Table -AutoSize
```

TimeCreated	ID	ProviderName	LevelDisplayName
-------------	----	--------------	------------------

Message

```
--  
8/3/2022 4:41:38 PM 145 Microsoft-Windows-WinRM Information WSMAN  
operation Enumeration started with resourceUri ...  
8/3/2022 4:41:42 PM 254 Microsoft-Windows-WinRM Information Activity Transfer  
8/3/2022 4:41:42 PM 161 Microsoft-Windows-WinRM Error The  
client cannot connect to the destination specifie...  
8/3/2022 4:41:42 PM 142 Microsoft-Windows-WinRM Error WSMAN  
operation Enumeration failed, error code 215085...  
8/3/2022 9:51:03 AM 145 Microsoft-Windows-WinRM Information WSMAN  
operation Enumeration started with resourceUri ...  
8/3/2022 9:51:07 AM 254 Microsoft-Windows-WinRM Information Activity Transfer
```

3. Retrieving events from .evtx Files

- If you have an exported `.evtx` file from another computer or you have backed up an existing log, you can utilize the `Get-WinEvent` cmdlet to read and query those logs.
 - This capability is particularly useful for auditing purposes or when you need to analyze logs within scripts.
- To retrieve log entries from a `.evtx` file, you need to provide the log file's path using the `-Path` parameter.
 - The example below demonstrates how to read events from the 'C:\Tools\chainsaw\EVTX-ATTACK-SAMPLES\Execution\exec_sysmon_1_lolbin_pcalua.evtx' file, which represents an exported Windows PowerShell log.

```
PS C:\Users\Administrator> Get-WinEvent -Path 'C:\Tools\chainsaw\EVTX-  
ATTACK-SAMPLES\Execution\exec_sysmon_1_lolbin_pcalua.evtx' -MaxEvents 5  
| Select-Object TimeCreated, ID, ProviderName, LevelDisplayName, Message  
| Format-Table -AutoSize
```

TimeCreated	Id	ProviderName	LevelDisplayName
Message			
--			
5/12/2019 10:01:51 AM	1	Microsoft-Windows-Sysmon	Information
Process Create:...			
5/12/2019 10:01:50 AM	1	Microsoft-Windows-Sysmon	Information
Process Create:...			

```
5/12/2019 10:01:43 AM 1 Microsoft-Windows-Sysmon Information
```

```
Process Create:...
```

- By specifying the path of the log file using the `-Path` parameter, we can retrieve events from that specific file.
 - The command selects relevant properties and formats the output for easier analysis, displaying the event's creation time, ID, provider name, level display name, and message.

4. Filtering events with `FilterHashtable`

- To filter Windows event logs, we can use the `-FilterHashtable` parameter, which enables us to define specific conditions for the logs we want to retrieve.

```
PS C:\Users\Administrator> Get-WinEvent -FilterHashtable  
@{LogName='Microsoft-Windows-Sysmon/Operational'; ID=1,3} | Select-  
Object TimeCreated, ID, ProviderName, LevelDisplayName, Message |  
Format-Table -AutoSize
```

TimeCreated	ID	ProviderName	LevelDisplayName
Message			
6/2/2023 10:40:09 AM	1	Microsoft-Windows-Sysmon	Information
Process Create:...			
6/2/2023 10:39:01 AM	1	Microsoft-Windows-Sysmon	Information
Process Create:...			
6/2/2023 10:34:12 AM	1	Microsoft-Windows-Sysmon	Information
Process Create:...			
6/2/2023 10:33:26 AM	1	Microsoft-Windows-Sysmon	Information
Process Create:...			
6/2/2023 10:33:16 AM	1	Microsoft-Windows-Sysmon	Information
Process Create:...			
6/2/2023 9:36:10 AM	3	Microsoft-Windows-Sysmon	Information
Network connection detected:...			
5/29/2023 6:30:26 PM	1	Microsoft-Windows-Sysmon	Information
Process Create:...			
5/29/2023 6:30:24 PM	3	Microsoft-Windows-Sysmon	Information
Network connection detected:...			

- The command above retrieves events with IDs 1 and 3 from the `Microsoft-Windows-Sysmon/Operational` event log, selects specific properties from those events, and

displays them in a table format.

- **Note:** If we observe Sysmon event IDs 1 and 3 (related to "dangerous" or uncommon binaries) occurring within a short time frame, it could potentially indicate the presence of a process communicating with a Command and Control (C2) server.
- For exported events the equivalent command is the following.

```
PS C:\Users\Administrator> Get-WinEvent -FilterHashtable
@{Path='C:\Tools\chainsaw\EVTX-ATTACK-
SAMPLES\Execution\sysmon_mshta_sharpshooter_stageless_meterpreter.evtx';
ID=1,3} | Select-Object TimeCreated, ID, ProviderName, LevelDisplayName,
Message | Format-Table -AutoSize

TimeCreated           Id ProviderName      LevelDisplayName
Message
-----
--  

6/15/2019 12:14:32 AM 1 Microsoft-Windows-Sysmon Information
Process Create:...
6/15/2019 12:13:44 AM 3 Microsoft-Windows-Sysmon Information
Network connection detected:...
6/15/2019 12:13:42 AM 1 Microsoft-Windows-Sysmon Information
Process Create:...
```

Note: These logs are related to a process communicating with a Command and Control (C2) server right after it was created.

- If we want the get event logs based on a date range (5/28/23 - 6/2/2023), this can be done as follows.

```
PS C:\Users\Administrator> $startDate = (Get-Date -Year 2023 -Month 5 -
Day 28).Date
PS C:\Users\Administrator> $endDate     = (Get-Date -Year 2023 -Month 6 -
Day 3).Date
PS C:\Users\Administrator> Get-WinEvent -FilterHashtable
@{LogName='Microsoft-Windows-Sysmon/Operational'; ID=1,3;
StartTime=$startDate; EndTime=$endDate} | Select-Object TimeCreated, ID,
ProviderName, LevelDisplayName, Message | Format-Table -AutoSize

TimeCreated           Id ProviderName      LevelDisplayName
Message
```

```
--  
6/2/2023 3:26:56 PM      1 Microsoft-Windows-Sysmon Information  
Process Create:...  
6/2/2023 3:25:20 PM      1 Microsoft-Windows-Sysmon Information  
Process Create:...  
6/2/2023 3:25:20 PM      1 Microsoft-Windows-Sysmon Information  
Process Create:...  
6/2/2023 3:24:13 PM      1 Microsoft-Windows-Sysmon Information  
Process Create:...  
6/2/2023 3:24:13 PM      1 Microsoft-Windows-Sysmon Information  
Process Create:...  
6/2/2023 3:23:41 PM      1 Microsoft-Windows-Sysmon Information  
Process Create:...  
6/2/2023 3:20:27 PM      1 Microsoft-Windows-Sysmon Information  
Process Create:...  
6/2/2023 3:20:26 PM      1 Microsoft-Windows-Sysmon Information  
Process Create:...  
--- SNIP ---
```

Note: The above will filter between the start date inclusive and the end date exclusive. That's why we specified June 3rd and not 2nd.

5. Filtering events with FilterHashtable & XML

- Consider an intrusion detection scenario where a suspicious network connection to a particular IP (52.113.194.132) has been identified.
 - With Sysmon installed, you can use Event ID 3 (Network Connection) logs to investigate the potential threat.

```
PS C:\Users\Administrator> Get-WinEvent -FilterHashtable  
@{LogName='Microsoft-Windows-Sysmon/Operational'; ID=3} |  
`ForEach-Object {  
$xml = [xml]$_.ToXml()  
$EventData = $xml.Event.EventData.Data  
New-Object PSObject -Property @{  
    SourceIP = $EventData | Where-Object {$_.Name -eq "SourceIp"} |  
Select-Object -ExpandProperty '#text'  
    DestinationIP = $EventData | Where-Object {$_.Name -eq  
"DestinationIp"} | Select-Object -ExpandProperty '#text'  
    ProcessGuid = $EventData | Where-Object {$_.Name -eq "ProcessGuid"} |  
Select-Object -ExpandProperty '#text'  
    ProcessId = $EventData | Where-Object {$_.Name -eq "ProcessId"} |  
Select-Object -ExpandProperty '#text'
```

```

}
} | Where-Object {$_.DestinationIP -eq "52.113.194.132"}
```

DestinationIP	ProcessId	SourceIP	ProcessGuid
52.113.194.132	9196	10.129.205.123	{52ff3419-51ad-6475-1201-00000000e00}
52.113.194.132	5996	10.129.203.180	{52ff3419-54f3-6474-3d03-00000000c00}

- This script will retrieve all Sysmon network connection events (ID 3), parse the XML data for each event to retrieve specific details (source IP, destination IP, Process GUID, and Process ID), and filter the results to include only events where the destination IP matches the suspected IP.
- Further, we can use the `ProcessGuid` to trace back the original process that made the connection, enabling us to understand the process tree and identify any malicious executables or scripts.
- You might wonder how we could have been aware of `Event.EventData.Data`.
 - The Windows XML EventLog (EVTX) format can be found [here](#)
- In the "Tapping Into ETW" section we were looking for anomalous `clr.dll` and `mscoree.dll` loading activity in processes that ordinarily wouldn't require them.
 - The command below is leveraging Sysmon's Event ID 7 to detect the loading of abovementioned DLLs.

```

PS C:\Users\Administrator> $Query = @"
<QueryList>
    <Query Id="0">
        <Select Path="Microsoft-Windows-
Sysmon/Operational">*[System[(EventID=7)]] and *
[EventData[Data='mscoree.dll']] or *[EventData[Data='clr.dll']]
        </Select>
    </Query>
</QueryList>
"@
```

```

PS C:\Users\Administrator> Get-WinEvent -FilterXml $Query | ForEach-Object {Write-Host $_.Message `n}
```

Image loaded:
RuleName: -

```
UtcTime: 2023-06-05 22:23:16.560
ProcessGuid: {52ff3419-6054-647e-aa02-000000001000}
ProcessId: 2936
Image: C:\Tools\GhostPack Compiled Binaries\Seatbelt.exe
ImageLoaded: C:\Windows\Microsoft.NET\Framework64\v4.0.30319\clr.dll
FileVersion: 4.8.4515.0 built by: NET48REL1LAST_C
Description: Microsoft .NET Runtime Common Language Runtime - WorkStation
Product: Microsoft® .NET Framework
Company: Microsoft Corporation
OriginalFileName: clr.dll
Hashes:
MD5=2B0E5597FF51A3A4D5BB2DDAB0214531, SHA256=8D09CE35C987EADCF01686BB5599
20951B0116985FE4FEB5A488A6A8F7C4BDB9, IMPHASH=259C196C67C4E02F941CAD54D9D
9BB8A
Signed: true
Signature: Microsoft Corporation
SignatureStatus: Valid
User: DESKTOP-NU10MT0\Administrator

Image loaded:
RuleName: -
UtcTime: 2023-06-05 22:23:16.544
ProcessGuid: {52ff3419-6054-647e-aa02-000000001000}
ProcessId: 2936
Image: C:\Tools\GhostPack Compiled Binaries\Seatbelt.exe
ImageLoaded: C:\Windows\System32\mscoree.dll
FileVersion: 10.0.19041.1 (WinBuild.160101.0800)
Description: Microsoft .NET Runtime Execution Engine
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: mscoree.dll
Hashes:
MD5=D5971EF71DE1BDD46D537203ABFCC756, SHA256=8828DE042D008783BA5B31C82935
A3ED38D5996927C3399B3E1FC6FE723FC84E, IMPHASH=65F23EFA1EB51A5DAAB399BFAA8
40074
Signed: true
Signature: Microsoft Windows
SignatureStatus: Valid
User: DESKTOP-NU10MT0\Administrator
--- SNIP ---
```

6. Filtering events with FilterXPath

- To use XPath queries with Get-WinEvent, we need to use the `-FilterXPath` parameter. This allows us to craft an XPath query to filter the event logs.
- For instance, if we want to get Process Creation ([Sysmon Event ID 1](#)) events in the Sysmon log to identify installation of any [Sysinternals](#) tool we can use the command below.
 - **Note:** During the installation of a Sysinternals tool the user must accept the presented EULA.
 - The acceptance action involves the registry key included in the command below.

```
PS C:\Users\Administrator> Get-WinEvent -LogName 'Microsoft-Windows-Sysmon/Operational' -FilterXPath "*[EventData[Data[@Name='Image']]='C:\Windows\System32\reg.exe'] and *[EventData[Data[@Name='CommandLine']]='`"C:\Windows\system32\reg.exe`" ADD HKCU\Software\Sysinternals /v EulaAccepted /t REG_DWORD /d 1 /f']]" | Select-Object TimeCreated, ID, ProviderName, LevelDisplayName, Message | Format-Table -AutoSize
```

TimeCreated	ID	ProviderName	LevelDisplayName
Message			
5/29/2023 12:44:46 AM	1	Microsoft-Windows-Sysmon	Information
		Process Create:...	
5/29/2023 12:29:53 AM	1	Microsoft-Windows-Sysmon	Information
		Process Create:...	

Note: `Image` and `CommandLine` can be identified by browsing the XML representation of any Sysmon event with ID 1 through, for example, Event Viewer.

The screenshot shows the Windows Event Viewer interface. On the left, there's a navigation pane with various logs like Security, System, and Application. The main pane is titled "Operational" and shows "Number of events: 4,223". A specific event is selected, labeled "Event 1, Sysmon". The event details are shown in an XML View, which includes the following data:

```

<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
  <System>
    <Provider Name="Sysmon" />
    <EventID>1</EventID>
    <Version>1</Version>
    <Level>Information</Level>
    <TaskCategory>Process Create</TaskCategory>
    <TimeCreated>2023-06-03T15:21:08Z</TimeCreated>
    <Source>Sysmon</Source>
  </System>
  <EventData>
    <Data Name="RuleName">Process Create</Data>
    <Data Name="UtcTime">2023-06-03 20:51:08.538</Data>
    <Data Name="ProcessGuid">{52ff3419-a7bc-647b-1901-000000000100}</Data>
    <Data Name="ProcessId">2490</Data>
    <Data Name="Image">C:\Windows\System32\SecurityHealthHost.exe</Data>
    <Data Name="FileVersion">4.18.1907.16384 (WinBuild.160101.0800)</Data>
    <Data Name="Description">Windows Security Health Host</Data>
    <Data Name="Product">Microsoft® Windows® Operating System</Data>
    <Data Name="Company">Microsoft Corporation</Data>
    <Data Name="OriginalFileName">SecurityHealthHost.exe</Data>
    <Data Name="CommandLine">C:\Windows\System32\SecurityHealthHost.exe {08728914-3F57-4D52-9E31-49DAECA5A80A} -Embedding</Data>
    <Data Name="CurrentDirectory">C:\Windows\system32</Data>
    <Data Name="User">DESKTOP-MU10MTO\Administrator</Data>
    <Data Name="LogonGuid">{52ff3419-a733-647b-b024-110000000000}</Data>
    <Data Name="LogonId">0x1124</Data>
    <Data Name="TerminalSessionId">2</Data>
    <Data Name="IntegrityLevel">High</Data>
  </EventData>

```

- Lastly, suppose we want to investigate any network connections to a particular suspicious IP address (52.113.194.132) that Sysmon has logged.
 - To do that we could use the following command.\

```
PS C:\Users\Administrator> Get-WinEvent -LogName 'Microsoft-Windows-Sysmon/Operational' -FilterXPath "*[System[EventID=3] and EventData[Data[@Name='DestinationIp']='52.113.194.132']]"
```

ProviderName: Microsoft-Windows-Sysmon

TimeCreated	Id	LevelDisplayName	Message
5/29/2023 6:30:24 PM detected:...	3	Information	Network connection
5/29/2023 12:32:05 AM detected:...	3	Information	Network connection

7. Filtering events based on property values

- The `-Property *` parameter, when used with `Select-Object`, instructs the command to select all properties of the objects passed to it.
 - In the context of the Get-WinEvent command, these properties will include all available information about the event.
 - Let's see an example that will present us with all properties of Sysmon event ID 1 logs.

```
PS C:\Users\Administrator> Get-WinEvent -FilterHashtable @{'LogName='Microsoft-Windows-Sysmon/Operational'; ID=1'} -MaxEvents 1 | Select-Object -Property *
```

Message	: Process Create:
	RuleName: -
	UtcTime: 2023-06-03 01:24:25.104
	ProcessGuid: {52ff3419-9649-647a-1902-000000001000}
	ProcessId: 1036
	Image: C:\Windows\System32\taskhostw.exe
	FileVersion: 10.0.19041.1806 (WinBuild.160101.0800)
	Description: Host Process for Windows Tasks
	Product: Microsoft® Windows® Operating System
	Company: Microsoft Corporation

```
        OriginalFileName: taskhostw.exe
        CommandLine: taskhostw.exe -RegisterDevice -
ProtectionStateChanged -FreeNetworkOnly
        CurrentDirectory: C:\Windows\system32\
        User: NT AUTHORITY\SYSTEM
        LogonGuid: {52ff3419-85d0-647a-e703-000000000000}
        LogonId: 0x3E7
        TerminalSessionId: 0
        IntegrityLevel: System
        Hashes:
MD5=C7B722B96F3969EACAE9FA205FAF7EF0 , SHA256=76D3D02B265FA5768294549C938D
3D9543CC9FEF6927

4728E0A72E3FCC335366 , IMPHASH=3A0C6863CDE566AF997DB2DEFFF9D924
        ParentProcessGuid: {00000000-0000-0000-0000-
000000000000}
        ParentProcessId: 1664
        ParentImage: -
        ParentCommandLine: -
        ParentUser: -
Id : 1
Version : 5
Qualifiers :
Level : 4
Task : 1
Opcode : 0
Keywords : -9223372036854775808
RecordId : 32836
ProviderName : Microsoft-Windows-Sysmon
ProviderId : 5770385f-c22a-43e0-bf4c-06f5698ffbd9
LogName : Microsoft-Windows-Sysmon/Operational
ProcessId : 2900
ThreadId : 2436
MachineName : DESKTOP-NU10MT0
UserId : S-1-5-18
TimeCreated : 6/2/2023 6:24:25 PM
ActivityId :
RelatedActivityId :
ContainerLog : Microsoft-Windows-Sysmon/Operational
MatchedQueryIds : {}
Bookmark :
System.Diagnostics.Eventing.Reader.EventBookmark
LevelDisplayName : Information
OpcodeDisplayName : Info
TaskDisplayName : Process Create (rule: ProcessCreate)
KeywordsDisplayNames : {}
```

```
Properties      :  
{System.Diagnostics.Eventing.Reader.EventProperty,  
 System.Diagnostics.Eventing.Reader.EventProperty,  
 System.Diagnostics.Eventing.Reader.EventProperty,  
 System.Diagnostics.Eventing.Reader.EventProperty...}
```

- Let's now see an example of a command that retrieves `Process Create` events from the `Microsoft-Windows-Sysmon/Operational` log, checks the parent command line of each event for the string `-enc`, and then displays all properties of any matching events as a list.

```
PS C:\Users\Administrator> Get-WinEvent -FilterHashtable  
@{LogName='Microsoft-Windows-Sysmon/Operational'; ID=1} | Where-Object  
{$_.Properties[21].Value -like "*-enc*"} | Format-List  
  
TimeCreated    : 5/29/2023 12:44:58 AM  
ProviderName   : Microsoft-Windows-Sysmon  
Id             : 1  
Message        : Process Create:  
    RuleName: -  
    UtcTime: 2023-05-29 07:44:58.467  
    ProcessGuid: {52ff3419-57fa-6474-7005-000000000c00}  
    ProcessId: 2660  
    Image:  
    C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe  
        FileVersion: 4.8.4084.0 built by: NET48REL1  
        Description: Visual C# Command Line Compiler  
        Product: Microsoft® .NET Framework  
        Company: Microsoft Corporation  
        OriginalFileName: csc.exe  
        CommandLine:  
        "C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe" /noconfig  
        /fullpaths  
        @"C:\Users\ADMINI~1\AppData\Local\Temp\z5erlc11.cmdline"  
        CurrentDirectory: C:\Users\Administrator\  
        User: DESKTOP-NU10MT0\Administrator  
        LogonGuid: {52ff3419-57f9-6474-8071-510000000000}  
        LogonId: 0x517180  
        TerminalSessionId: 0  
        IntegrityLevel: High  
        Hashes:  
        MD5=F65B029562077B648A6A5F6A1AA76A66 , SHA256=4A6D0864E19C0368A47217C129B0  
        75DDDF61A6A262388F9D2104  
        5D82F3423ED7 , IMPHASH=EE1E569AD02AA1F7AECA80AC0601D80D
```

ParentProcessGuid: {52ff3419-57f9-6474-6e05-000000000c00}
ParentProcessId: 5840
ParentImage:
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
ParentCommandLine:
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile
-NonInteractive -ExecutionPolicy Unrestricted -EncodedCommand
JgBjAGGyAYwBwAC4AYwBvAG0AIAA2ADUAMAAwADEAIAA+ACAAJABuAHUAbABsAAoAaQBmACAA
KAAkAFAAUwBWAGUAcgBzAGkAbwBuAFQAYQBiAGwAZQAUAFAAUwBWAGUAcgBzAGkAbwBuACAA
LQBzAHQAIABbAFYAZQByAHMAaQBvAG4AXQAiADMALgAwACIAKQAgAHsACgAnAHsAIgBmAGEA
aQBzAGUAZAAiADoAdAByAHUAZQAsACIAbQBzAGcAIgA6ACIAQQBuAHMAaQBiAGwAZQAgAHIA
ZQBxAHUAAQByAGUAcwAgAFAAbwB3AGUAcgBTAGgAZQBsAGwAIAB2ADMALgAwACAAAbwByACAA
bgBlAHcAZQByACIAfQAnAAoAZQB4AGkAdAAgADEACgB9AAoAJABlAHgAZQBjAF8AdwByAGEA
cABwAGUAcgBfAHMAdAByACAAPQAgACQAaQBuAHAAAdQb0ACAAfAAgAE8AdQb0AC0AUwB0AHIA
aQBuAGcACgAkAHMAcABsAGkAdABfAHAAyQByAHQAcwAgAD0AIAAkAGUAcBLAGMAXwB3AHIA
YQBwAHAAZQByAF8AcwB0AHIALgBTAHAAbABpAHQAKABAACgAIgBgADAAYAAwAGAAMABgADAA
IgApACwAIAAyACwAIABbAFMAdAByAGkAbgBnAFMAcABsAGkAdABPAHAAdABpAG8AbgBzAF0A
OgA6AFIAZQBtAG8AdgBlAEUAbQBwAHQAeQBFBAG4AdAByAGkAZQBzACKACgBJAGYAIAAoAC0A
bgBvAHQAIAAkAHMAcABsAGkAdABfAHAAyQByAHQAcwAuAEwAZQBuAGcAdABoACAALQB1AHEA
IAAyACKAIAB7ACAAdABoAHIAbwB3ACAAIgBpAG4AdgBhAGwAaQBkACAAcABhAHkAbABvAGEA
ZAAiACAAfQAKAFMAZQB0AC0AVgBhAHIAaQBhAGIAbABLACAALQB0AGEAbQB1ACAagBzAG8A
bgBfAHIAyQb3ACAALQBWAGEAbAB1AGUAIAAkAHMAcABsAGkAdABfAHAAyQByAHQAcwBbADEA
XQAKACQAZQB4AGUAYwBfAHcAcgBhAHAAcABLAIHIAIA9ACAAwBTAGMAcgBpAHAAdABCAGwA
bwBjAGsAXQA6ADoAQwByAGUAYQB0AGUAKAAkAHMAcABsAGkAdABfAHAAyQByAHQAcwBbADAA
XQApAAoAJgAkAGUAeABLAGMAXwB3AHIAyQbWAHAAZQByAA==

ParentUser: DESKTOP-NU10MT0\Administrator

TimeCreated : 5/29/2023 12:44:57 AM
ProviderName : Microsoft-Windows-Sysmon
Id : 1
Message : Process Create:
RuleName: -
UtcTime: 2023-05-29 07:44:57.919
ProcessGuid: {52ff3419-57f9-6474-6f05-000000000c00}
ProcessId: 3060
Image: C:\Windows\System32\chcp.com
FileVersion: 10.0.19041.1806 (WinBuild.160101.0800)
Description: Change CodePage Utility
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: CHCP.COM
CommandLine: "C:\Windows\system32\chcp.com" 65001
CurrentDirectory: C:\Users\Administrator\
User: DESKTOP-NU10MT0\Administrator
LogonGuid: {52ff3419-57f9-6474-8071-510000000000}
LogonId: 0x517180

```

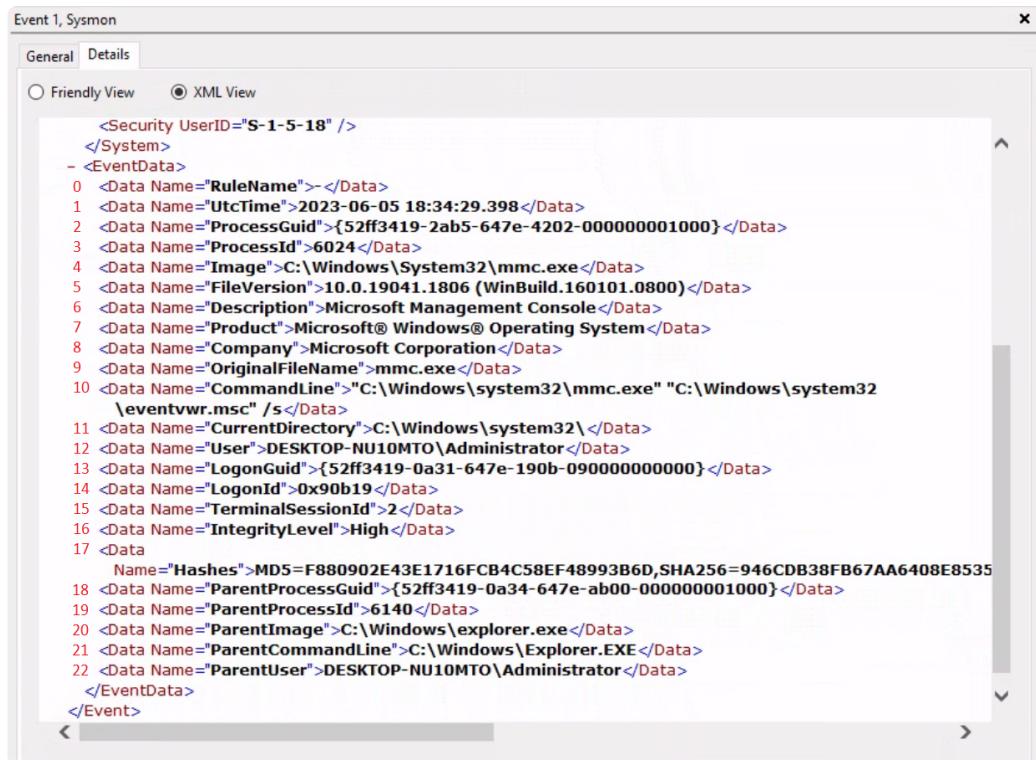
TerminalSessionId: 0
IntegrityLevel: High
Hashes:
MD5=33395C4732A49065EA72590B14B64F32 , SHA256=025622772AFB1486F4F7000B70CC
51A20A640474D6E4DBE95A70
BEB3FD53AD40 , IMPHASH=75FA51C548B19C4AD5051FAB7D57EB56
ParentProcessGuid: {52ff3419-57f9-6474-6e05-000000000c00}
ParentProcessId: 5840
ParentImage:
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
ParentCommandLine:
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile
    -NonInteractive -ExecutionPolicy Unrestricted -EncodedCommand
JgBjAGgAYwBwAC4AYwBvAG0AIAA2ADUAMAAwADEAIAA+ACAAJABuAHUAbABsAAoAaQBmACAA
KAAkAFAAUwBWAGUAcgBzAGkAbwBuAFQAYQBiAGwAZQAuAFAAUwBWAGUAcgBzAGkAbwBuACAA
LQBxAHQAIABbAFYAZQByAHMAaQBvAG4AXQAiADMALgAwACIAKQAgAHsACgAnAHsAIgBmAGEA
aQBxAGUAZAAiADoAdAByAHUAZQAsACIAbQBzAGcAIgA6ACIAQQBuAHMAaQBxAGwAZQAgAHIA
ZQBxAHUAAqByAGUAcwAgAFAAbwB3AGUAcgBTAGgAZQBsAGwAIAB2ADMALgAwACAAAbwByACAA
bgBlAHcAZQByACIAfQAnAAoAZQB4AGkAdAAgADEACgB9AAoAJABLHgAZQByAF8AdwByAGEA
cABwAGUAcgBfAHMAdAByACAAPQAgACQAAQBuAHAAdQB0ACAAfAAgAE8AdQB0AC0AUwB0AHIA
aQBxAGcACgAkAHMAcABsAGkAdABfAHAAAYQByAHQAcwAgAD0AIAAkAGUAeABLAGMAXwB3AHIA
YQBwAHAAZQByAF8AcwB0AHIALgBTAAAbABpAHQAKABAACgAIgBgADAAYAAwAGAAMABgADAA
IgApAcwAIAAyAcwAIABbAFMAdAByAGkAbgBnAFMacABsAGkAdABPAHAAdABpAG8AbgBzAF0A
0gA6AFIAZQbtAG8AdgBlAEUAbQBwAHQAeQBFBAG4AdAByAGkAZQBzACKACgBJAGYAIAAoAC0A
bgBvAHQAIAAkAHMAcABsAGkAdABfAHAAAYQByAHQAcwAuAEwAZQBxAGcAdABoACAALQB1AHEA
IAAyACKAIAB7ACAAAdABoAHIAbwB3ACAAIgBpAG4AdgBhAGwAaQBkACAAcABhAHkAbABvAGEA
ZAAiACAAfQAKAFMAZQB0AC0AVgBhAHIAaQBhAGIAbABlACAALQB0AGEAbQB1ACAagBzAG8A
bgBfAHIAYQB3ACAALQBWAGEAbAB1AGUAIAAkAHMAcABsAGkAdABfAHAAAYQByAHQAcwBbADEA
XQAKACQAZQB4AGUAYwBfAHcAcgBhAHAAcABlAHIAIA9ACAAwBTAGMACgBpAHAAdABCAGwA
bwBjAGsAXQA6ADoAQwByAGUAYQB0AGUAKAAkAHMAcABsAGkAdABfAHAAAYQByAHQAcwBbADAA
XQApAAoAJgAkAGUAeABLAGMAXwB3AHIAZQByAA==

ParentUser: DESKTOP-NU10MT0\Administrator
--- SNIP ---

```

- | Where-Object {\$_.Properties[21].Value -like "*-enc*"} : This portion of the command further filters the retrieved events.
 - The '|' character (pipe operator) passes the output of the previous command (i.e., the filtered events) to the 'Where-Object' cmdlet.
 - The 'Where-Object' cmdlet filters the output based on the script block that follows it.
 - `$_.Properties[21].Value -like "*-enc*"`: In the script block, `$_.Properties[21].Value` refers to the current object in the pipeline, i.e., each individual event that was retrieved and passed from the previous command.

- `.Properties[21].Value`: The `Properties` property of a "Process Create" Sysmon event is an array containing various data about the event.
 - The specific index `21` corresponds to the `ParentCommandLine` property of the event, which holds the exact command line used to start the process.



The screenshot shows the Windows Event Viewer interface for a Sysmon event. The event details are displayed in XML format under the 'Details' tab. The XML content is as follows:

```

<Event 1, Sysmon>
  General Details
  Friendly View XML View
  <Security UserID="S-1-5-18" />
  </System>
  - <EventData>
    0 <Data Name="RuleName">-</Data>
    1 <Data Name="UtcTime">2023-06-05 18:34:29.398</Data>
    2 <Data Name="ProcessGuid">{52ff3419-2ab5-647e-4202-000000001000}</Data>
    3 <Data Name="ProcessId">6024</Data>
    4 <Data Name="Image">C:\Windows\System32\mmc.exe</Data>
    5 <Data Name="FileVersion">10.0.19041.1806 (WinBuild.160101.0800)</Data>
    6 <Data Name="Description">Microsoft Management Console</Data>
    7 <Data Name="Product">Microsoft® Windows® Operating System</Data>
    8 <Data Name="Company">Microsoft Corporation</Data>
    9 <Data Name="OriginalFileName">mmc.exe</Data>
    10 <Data Name="CommandLine">"C:\Windows\system32\mmc.exe" "C:\Windows\system32\eventvwr.msc" /s</Data>
    11 <Data Name="CurrentDirectory">C:\Windows\system32\</Data>
    12 <Data Name="User">DESKTOP-NU10MTO\Administrator</Data>
    13 <Data Name="LogonGuid">{52ff3419-0a31-647e-190b-090000000000}</Data>
    14 <Data Name="LogonId">0x90b19</Data>
    15 <Data Name="TerminalSessionId">2</Data>
    16 <Data Name="IntegrityLevel">High</Data>
    17 <Data Name="Hashes">MD5=F880902E43E1716FCB4C58EF48993B6D,SHA256=946CDB3FB67AA6408E8535
    18 <Data Name="ParentProcessGuid">{52ff3419-0a34-647e-ab00-000000001000}</Data>
    19 <Data Name="ParentProcessId">6140</Data>
    20 <Data Name="ParentImage">C:\Windows\explorer.exe</Data>
    21 <Data Name="ParentCommandLine">"C:\Windows\Explorer.EXE"</Data>
    22 <Data Name="ParentUser">DESKTOP-NU10MTO\Administrator</Data>
  </EventData>
</Event>

```

- `-like "*-enc*"`: This is a comparison operator that matches strings based on a wildcard string, where `*` represents any sequence of characters.
 - In this case, it's looking for any command lines that contain `-enc` anywhere within them.
 - The `-enc` string might be part of suspicious commands, for example, it's a common parameter in PowerShell commands to denote an encoded command which could be used to obfuscate malicious scripts.
- `| Format-List` : Finally, the output of the previous command (the events that meet the specified condition) is passed to the `Format-List` cmdlet.
 - This cmdlet displays the properties of the input objects as a list, making it easier to read and analyze.