Labs - Windows Event Logs & Finding Evil

Skills Assessment

Scenario

- To keep you sharp, your SOC manager has assigned you the task of analyzing older attack logs and providing answers to specific questions.

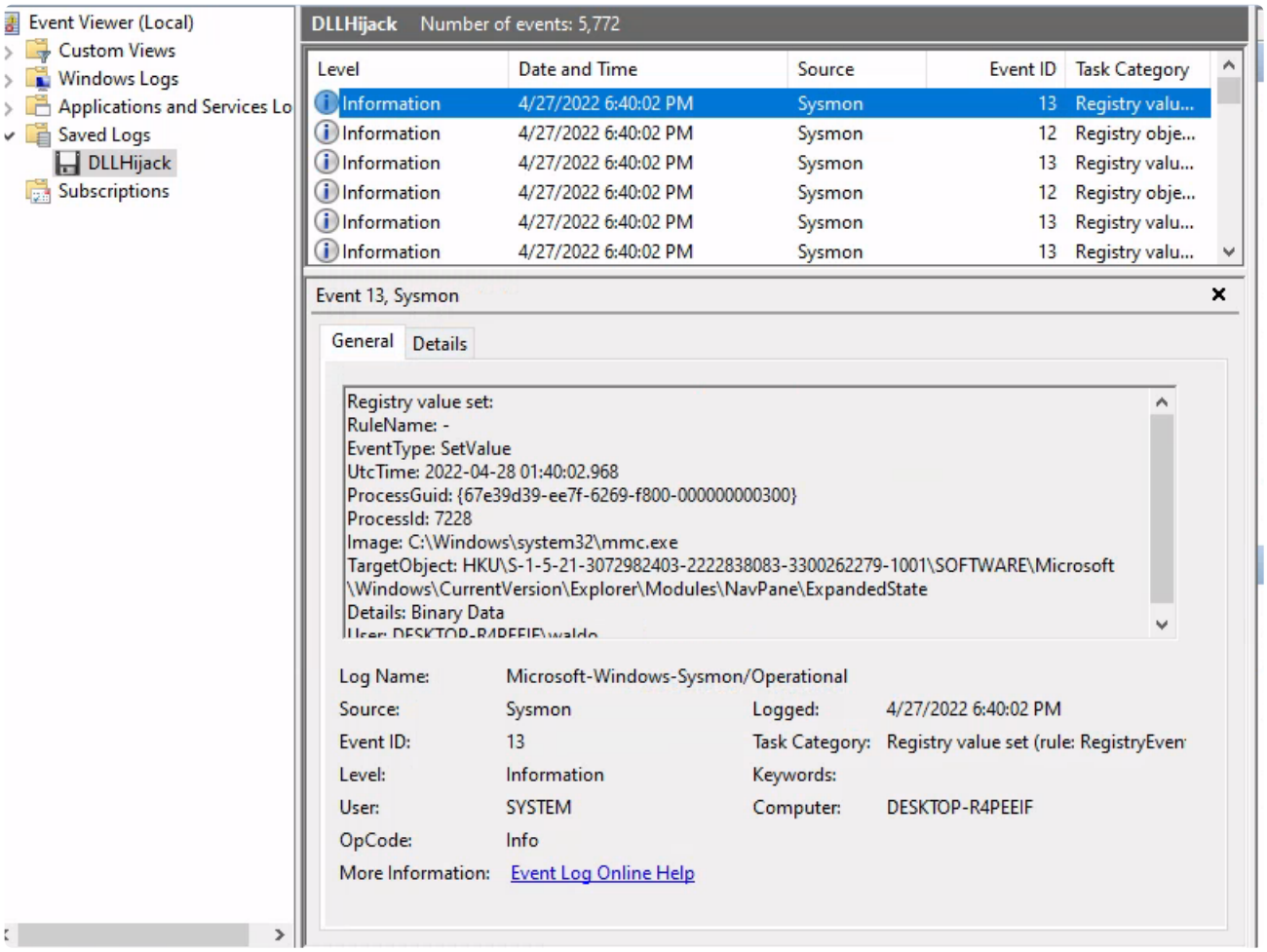
Questions

- By examining the logs located in the "C\Logs\DLLHijack" directory, determine the process responsible for executing a DLL hijacking attack. Enter the process name as your answer. Answer format: _.exe

-> For an DLL hijacking attack, we can look for Event ID 7 in Sysmon.

```
Get-WinEvent -FilterHashtable @{Path='C:\Logs\DLLHijack\EVTX-ATTACK-
SAMPLES\Execution\sysmon_mshta_sharpshooter_stageless_meterpreter.evtx';
ID=1,3}
```

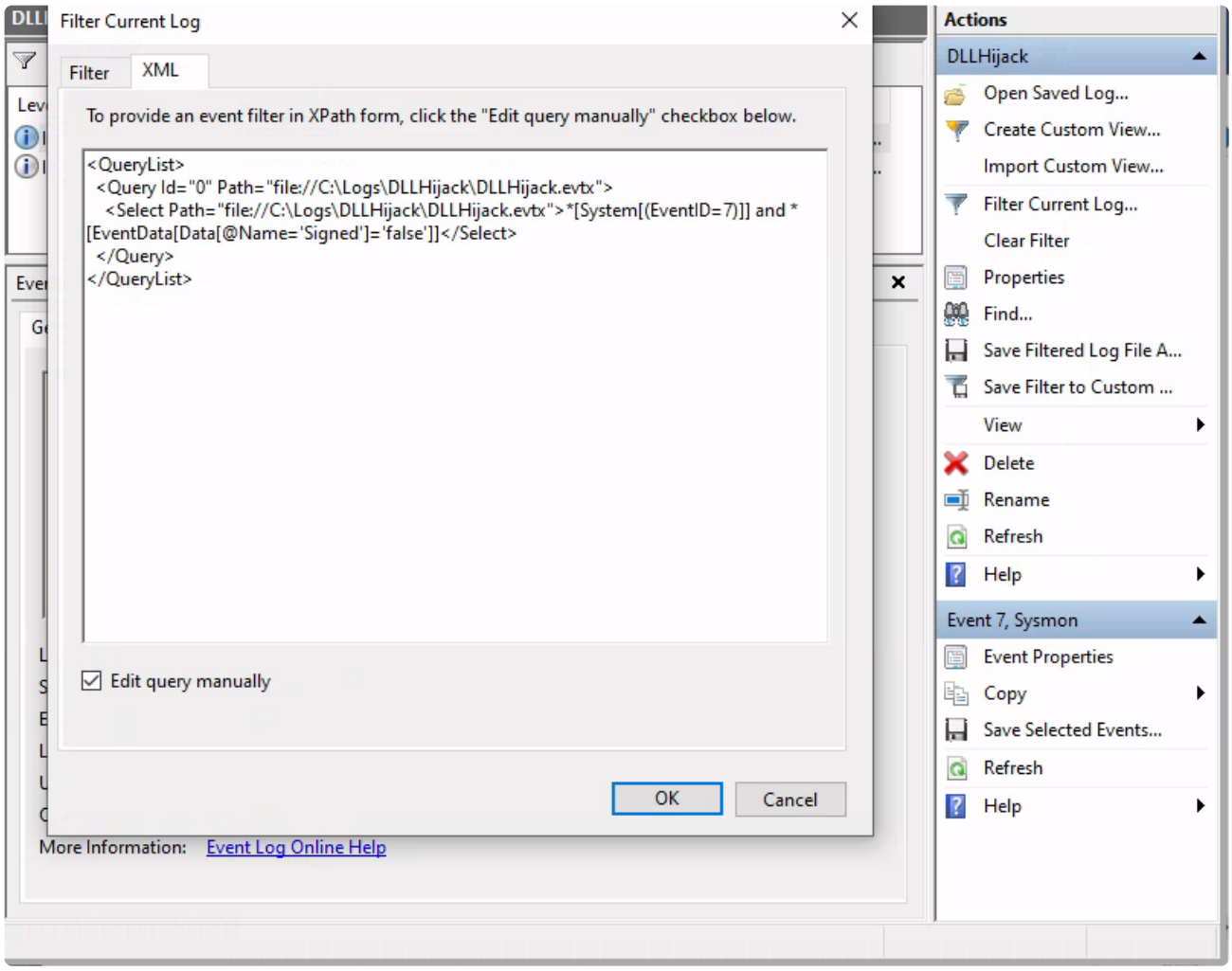
-> We click on the link and examine the log file

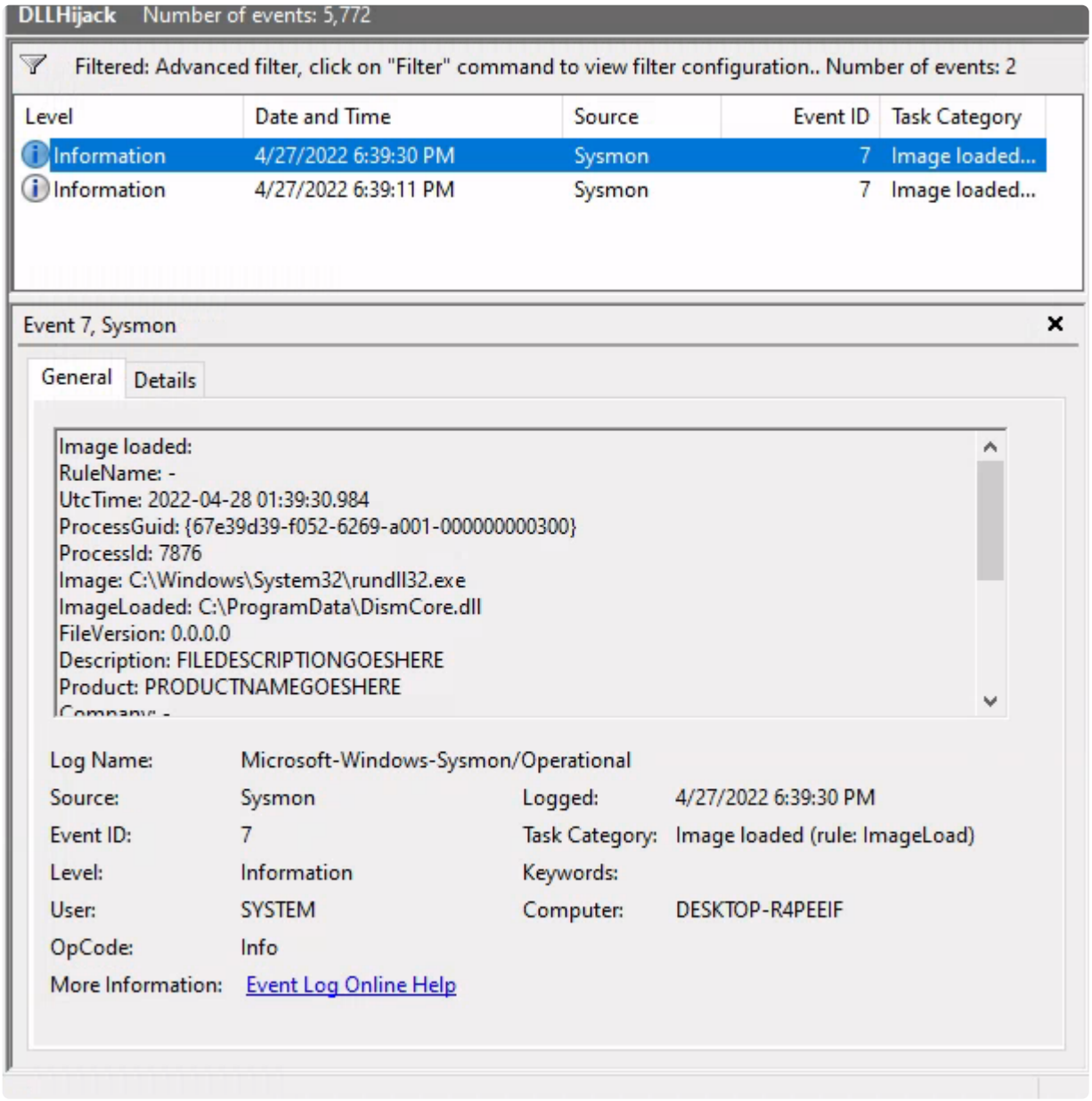


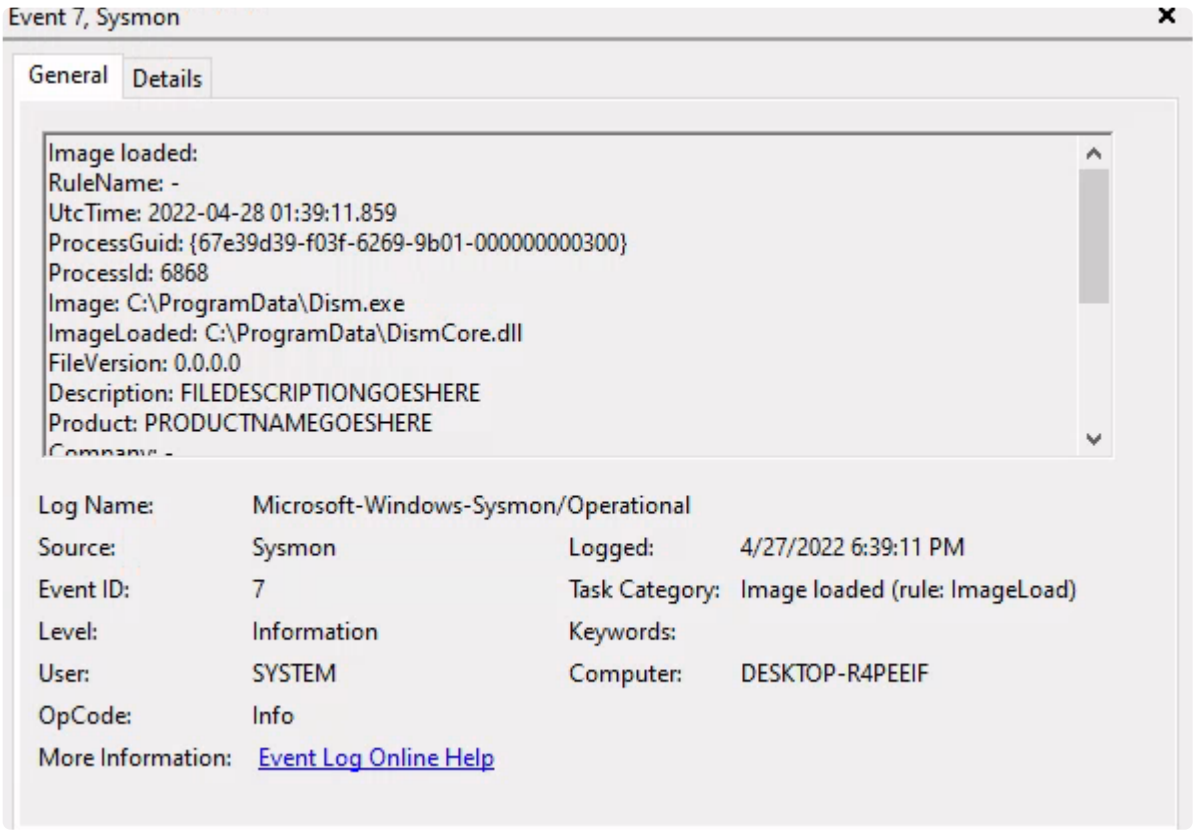
-> We see that there are many log files (5712) to analyze:

-> However, we know if we do search for dll's that are not signed, we can greatly reduced our results. so we filter for the `signed=false`

```
<Select Path="file://C:\Logs\DLLHijack\DLLHijack.evtx">*
[System[(EventID=7)]] and *[EventData[Data[@Name='Signed']='false']]
</Select>
```

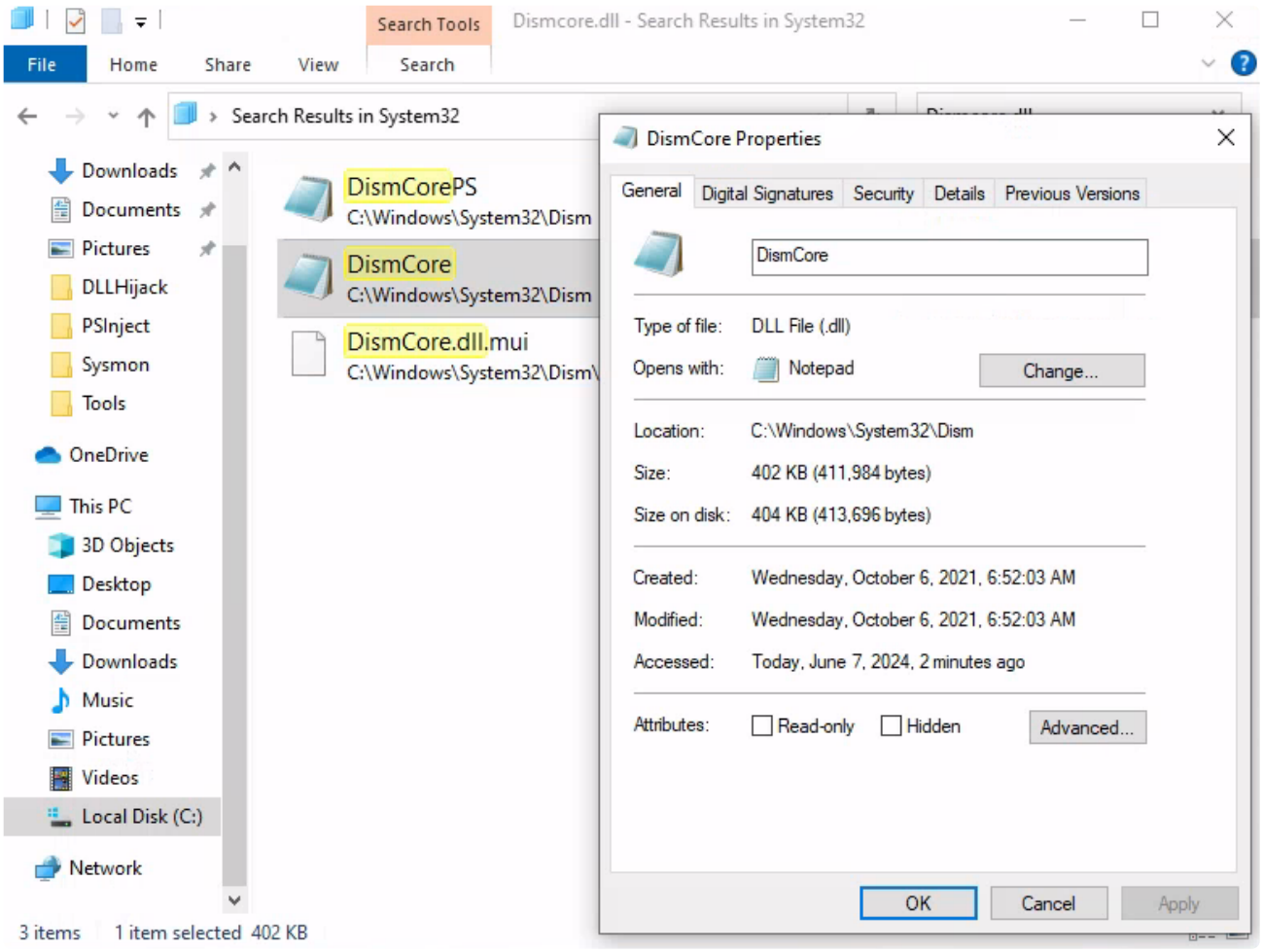

Filter Current Log
Filter
XML
To provide an event filter in XPath form, click the "Edit query manually" checkbox below.
<QueryList>
<Query Id="0" Path="file://C:\Logs\DLLHijack\DLLHijack.evtx">
<Select Path="file://C:\Logs\DLLHijack\DLLHijack.evtx">*[System[(EventID=7)]] and *
[EventData[Data[@Name='Signed']='false']]</Select>
</Query>
</QueryList>
Edit query manually
OK
Cancel
More Information:
Event Log Online Help
Actions
DLLHijack
Open Saved Log...
Create Custom View...
Import Custom View...
Filter Current Log...
Clear Filter
Properties
Find...
Save Filtered Log File A...
Save Filter to Custom ...
View
Delete
Rename
Refresh
Help
Event 7, Sysmon
Event Properties
Copy
Save Selected Events...
Refresh
Help


DLLHijack Number of events: 5,772
Filtered: Advanced filter, click on "Filter" command to view filter configuration.. Number of events: 2
Level Date and Time Source Event ID Task Category
Information 4/27/2022 6:39:30 PM Sysmon 7 Image loaded...
Information 4/27/2022 6:39:11 PM Sysmon 7 Image loaded...
Event 7, Sysmon
General Details
Image loaded:
RuleName: -
UtcTime: 2022-04-28 01:39:30.984
ProcessGuid: {67e39d39-f052-6269-a001-000000000300}
ProcessId: 7876
Image: C:\Windows\System32\rundll32.exe
ImageLoaded: C:\ProgramData\DismCore.dll
FileVersion: 0.0.0.0
Description: FILEDESCRIPTIONGOESHERE
Product: PRODUCTNAMEGOESHERE
Log Name: Microsoft-Windows-Sysmon/Operational
Source: Sysmon
Logged: 4/27/2022 6:39:30 PM
Event ID: 7
Task Category: Image loaded (rule: ImageLoad)
Level: Information
Keywords:
User: SYSTEM
Computer: DESKTOP-R4PEEIF
OpCode: Info
More Information: Event Log Online Help



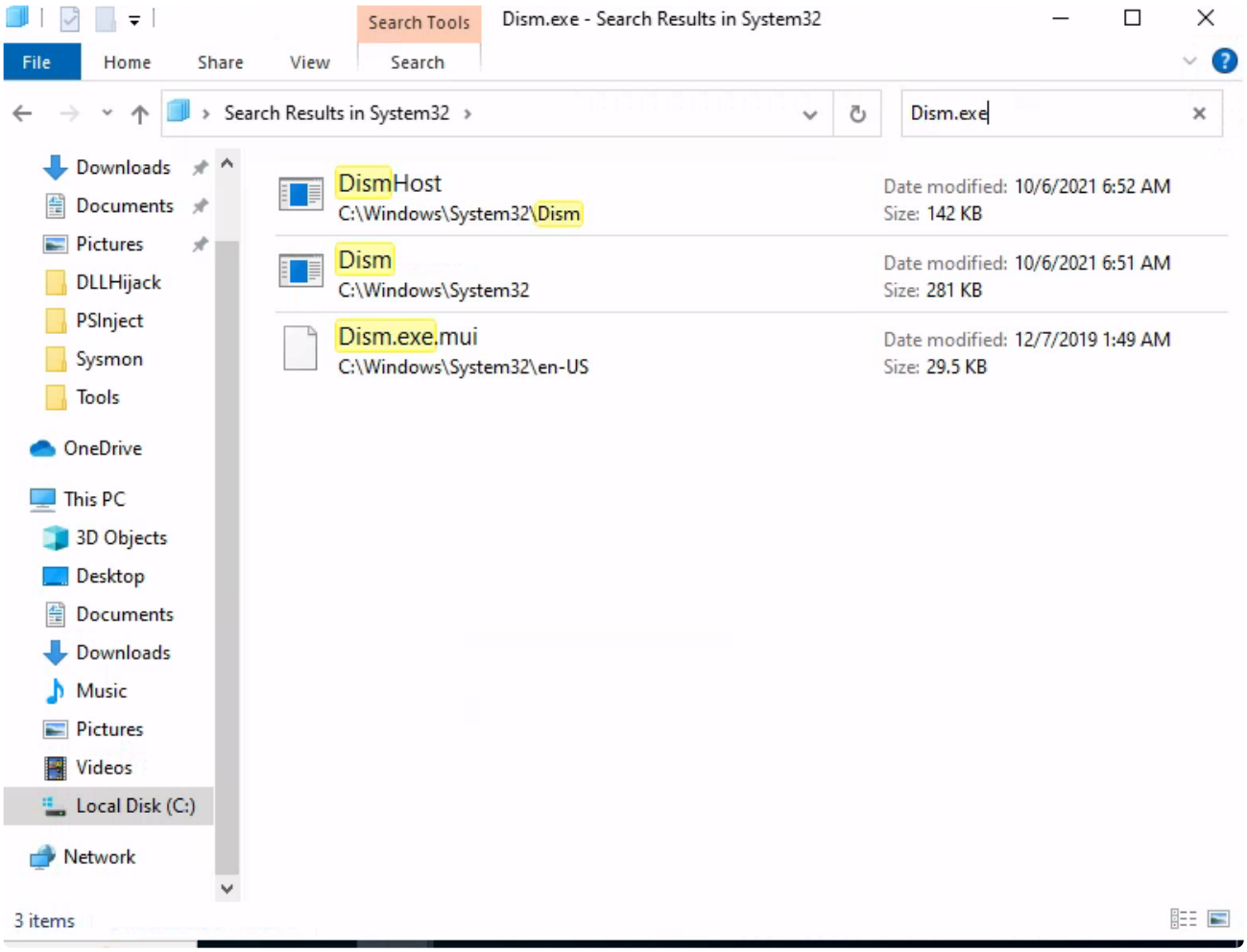
-> We see that we have `rundll32.exe` and `Dism.exe` as candidate binaries for performing DLL hijacking.

-> However, looking at the the system32 folder, we see the following:



-> We see that `DismCore.dll` is the hijacked DLL.

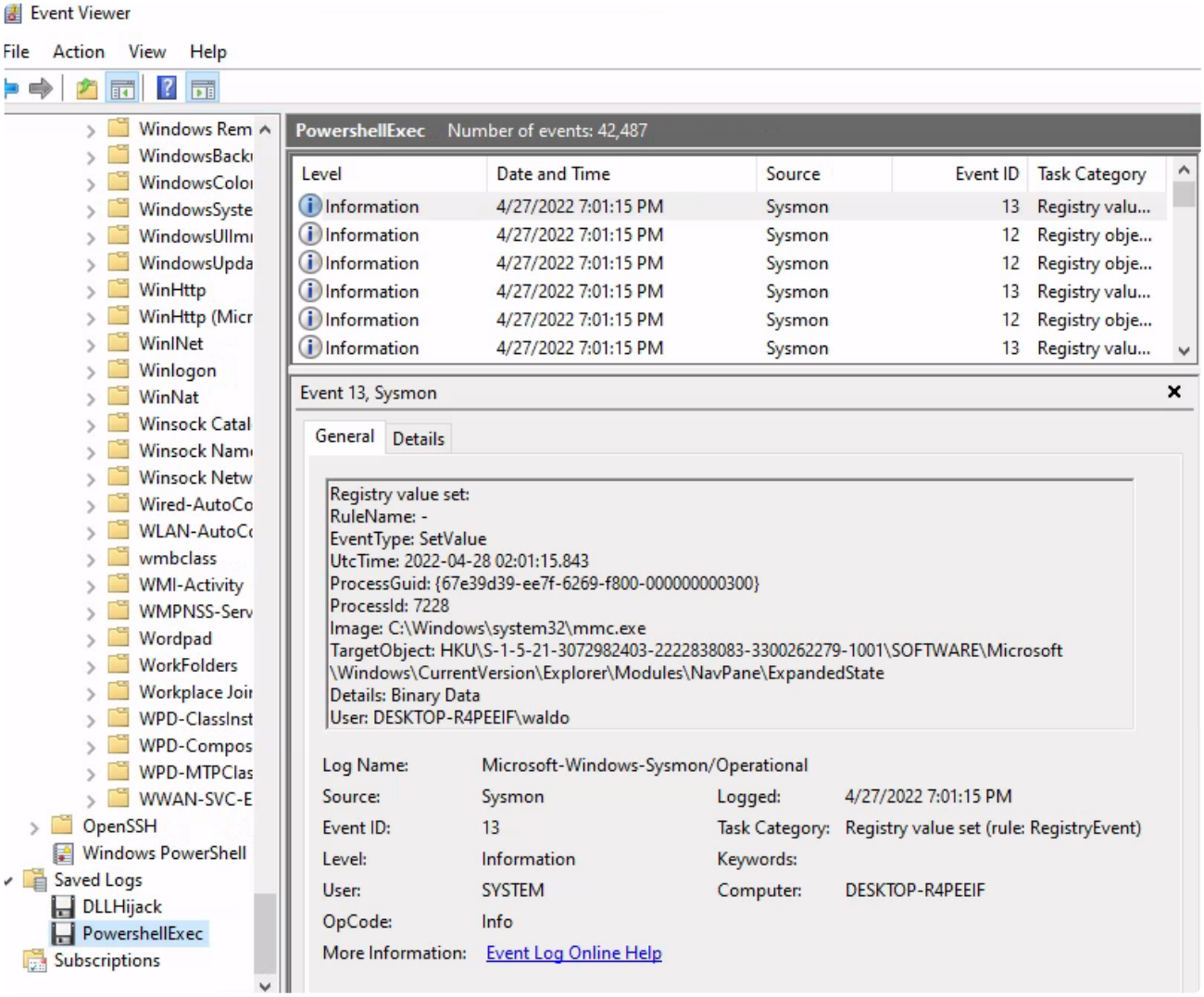
-> We also see Dism being an exe as follows:



-> Thus, we can say that it's `Dism.exe` being pulled from `System32` to `ProgramData` along with it's DLL.

- By examining the logs located in the "C:\Logs\PowershellExec" directory, determine the process that executed unmanaged PowerShell code. Enter the process name as your answer. Answer format: _.exe

-> We look at the log as files

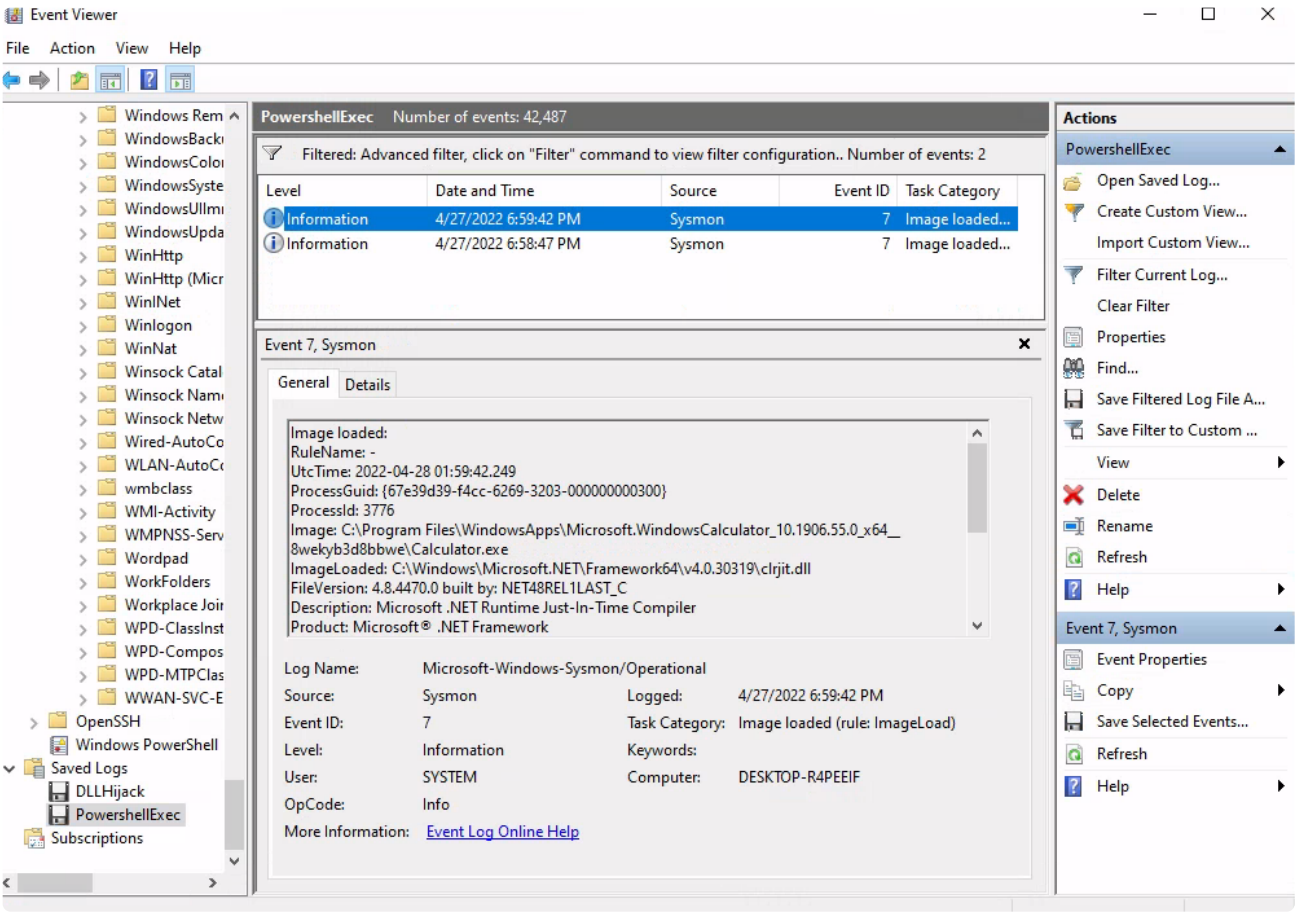


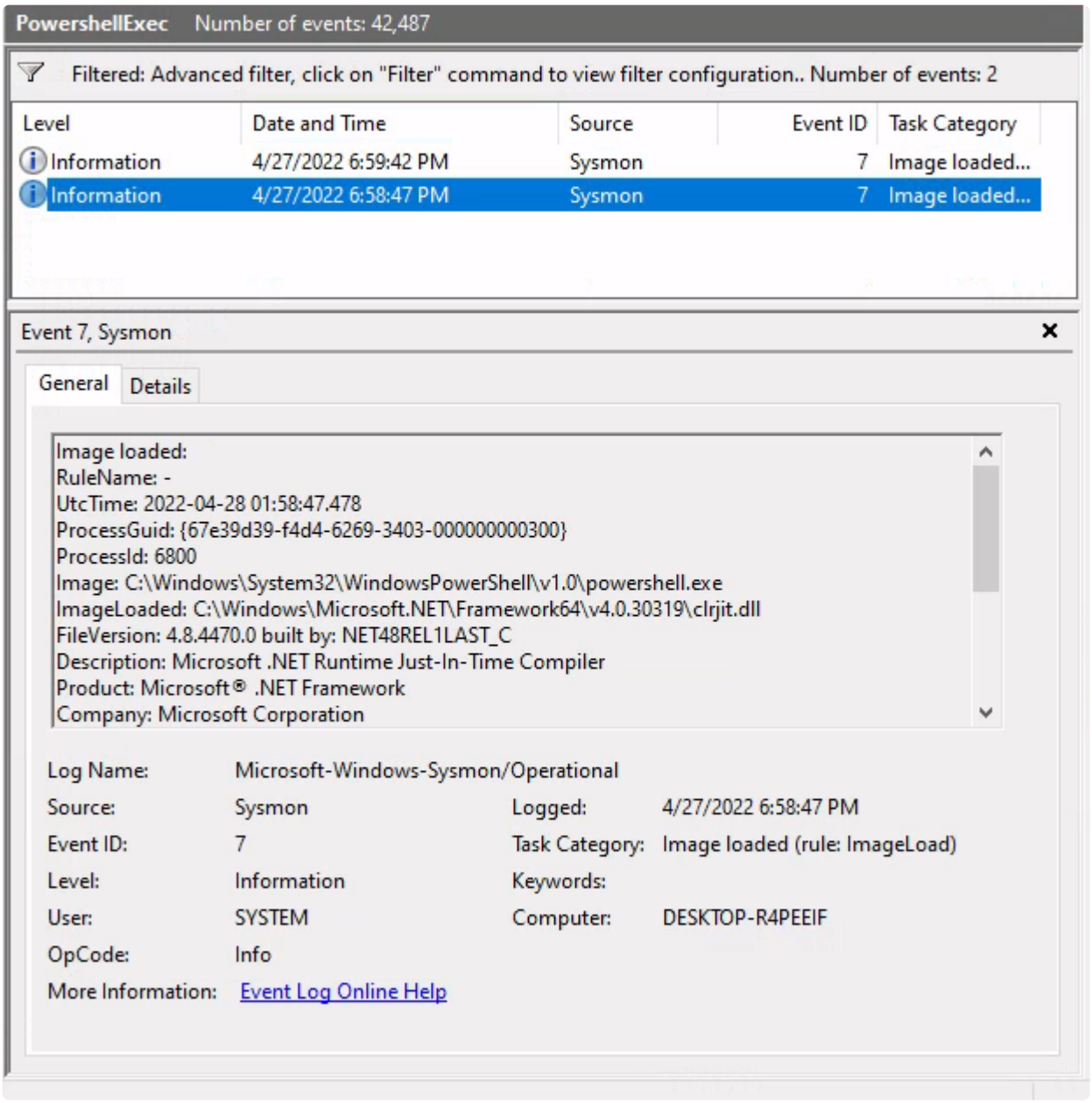
-> once again, there are many log files, so we filter them.

-> To filter for suspicious unmanaged code execution, we filter for logs with the loading of DLL's of `clr.dll` and `clrjit.dll`, which are required when C# code is ran as part to execute the bytecode, where the C# code may contain malicious code (in a nutshell, C# code running in runtime environment with the bytecode)

-> We filter using the following xml query

```
<Select Path="file://C:\Logs\PowershellExec\PowershellExec.evtx">*
[System[(EventID=1)]] and *
[EventData[Data[@Name='ImageLoaded']='C:\Windows\Microsoft.NET\Framework
64\v4.0.30319\clrjit.dll']]</Select>
```


Event Viewer
File Action View Help
Windows Rem
WindowsBacku
WindowsColor
WindowsSyste
WindowsUIImi
WindowsUpda
WinHttp
WinHttp (Micr
WinINet
Winlogon
WinNat
Winsock Catal
Winsock Nam
Winsock Netw
Wired-AutoCo
WLAN-AutoC
wmbclass
WMI-Activity
WMPNSS-Serv
Wordpad
WorkFolders
Workplace Joi
WPD-ClassInst
WPD-Compos
WPD-MTPClas
WWAN-SVC-E
OpenSSH
Windows PowerShell
Saved Logs
DLLHijack
PowershellExec
Subscriptions
PowershellExec Number of events: 42,487
Filtered: Advanced filter, click on "Filter" command to view filter configuration.. Number of events: 2
Level Date and Time Source Event ID Task Category
Information 4/27/2022 6:59:42 PM Sysmon 7 Image loaded...
Information 4/27/2022 6:58:47 PM Sysmon 7 Image loaded...
Event 7, Sysmon
General Details
Image loaded:
RuleName: -
UtcTime: 2022-04-28 01:59:42.249
ProcessGuid: {67e39d39-f4cc-6269-3203-000000000300}
ProcessId: 3776
Image: C:\Program Files\WindowsApps\Microsoft.WindowsCalculator_10.1906.55.0_x64__
8wekyb3d8bbwe\Calculator.exe
ImageLoaded: C:\Windows\Microsoft.NET\Framework64\v4.0.30319\clrjit.dll
FileVersion: 4.8.4470.0 built by: NET48REL1LAST_C
Description: Microsoft .NET Runtime Just-In-Time Compiler
Product: Microsoft® .NET Framework
Log Name: Microsoft-Windows-Sysmon/Operational
Source: Sysmon
Logged: 4/27/2022 6:59:42 PM
Event ID: 7
Task Category: Image loaded (rule: ImageLoad)
Level: Information
Keywords:
User: SYSTEM
Computer: DESKTOP-R4PEEIF
OpCode: Info
More Information: Event Log Online Help
Actions
PowershellExec
Open Saved Log...
Create Custom View...
Import Custom View...
Filter Current Log...
Clear Filter
Properties
Find...
Save Filtered Log File A...
Save Filter to Custom ...
View
Delete
Rename
Refresh
Help
Event 7, Sysmon
Event Properties
Copy
Save Selected Events...
Refresh
Help



-> We see that we are left with 2 log results left.
-> It's reasonable to see an Powershell executing c# code, but it is uncommon to see a calculator doing that, especially in real-time.
-> Hence, calculator.exe is the process executing unmanaged PowerShell code.

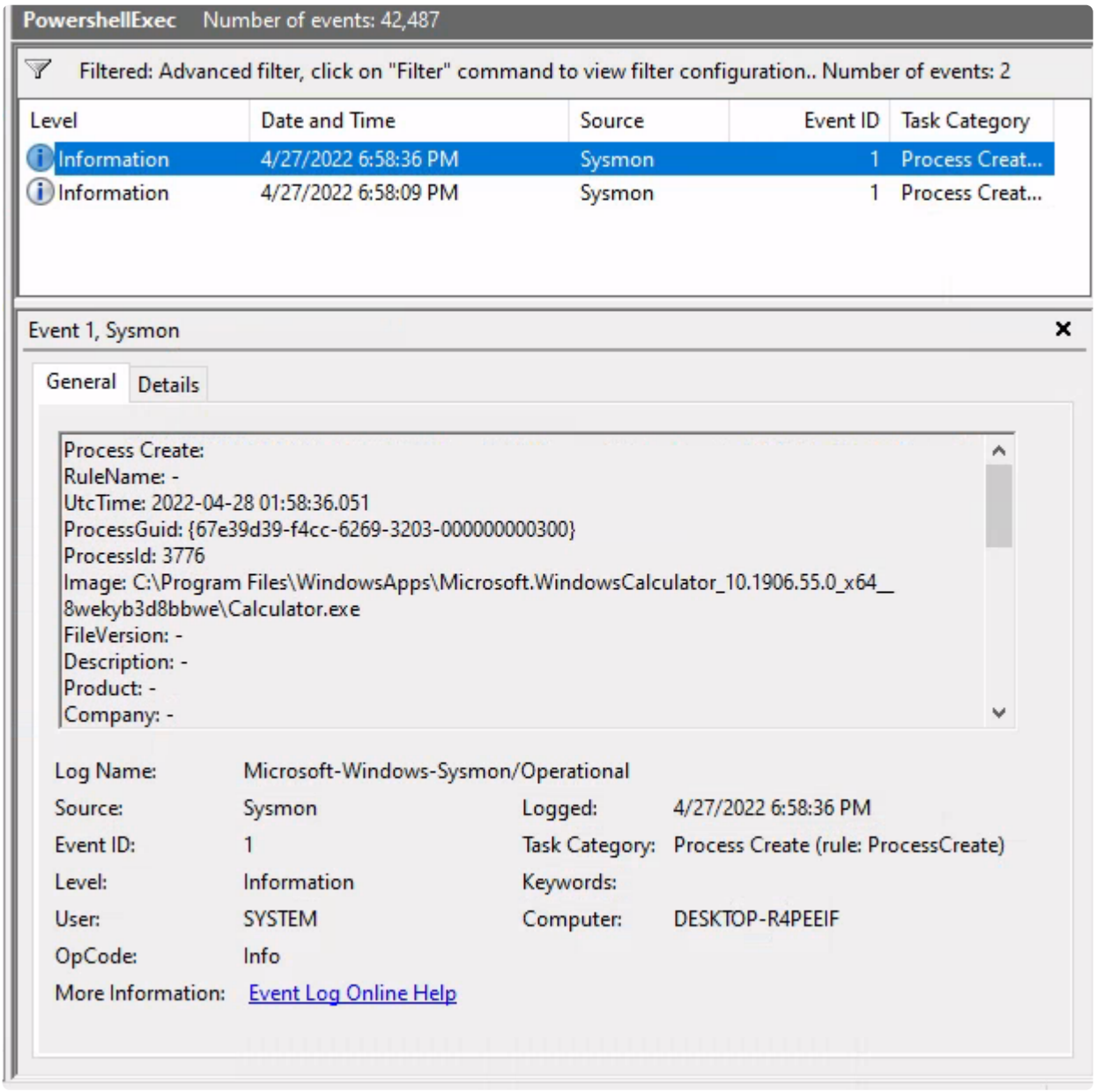
- By examining the logs located in the "C:\Logs\PowershellExec" directory, determine the process that injected into the process that executed unmanaged PowerShell code. Enter the process name as your answer. Answer format: _.exe

-> The question showed that we should look at detecting unusual parent-child relationship.
->we first looked at the EventId for sysmon for process creation:

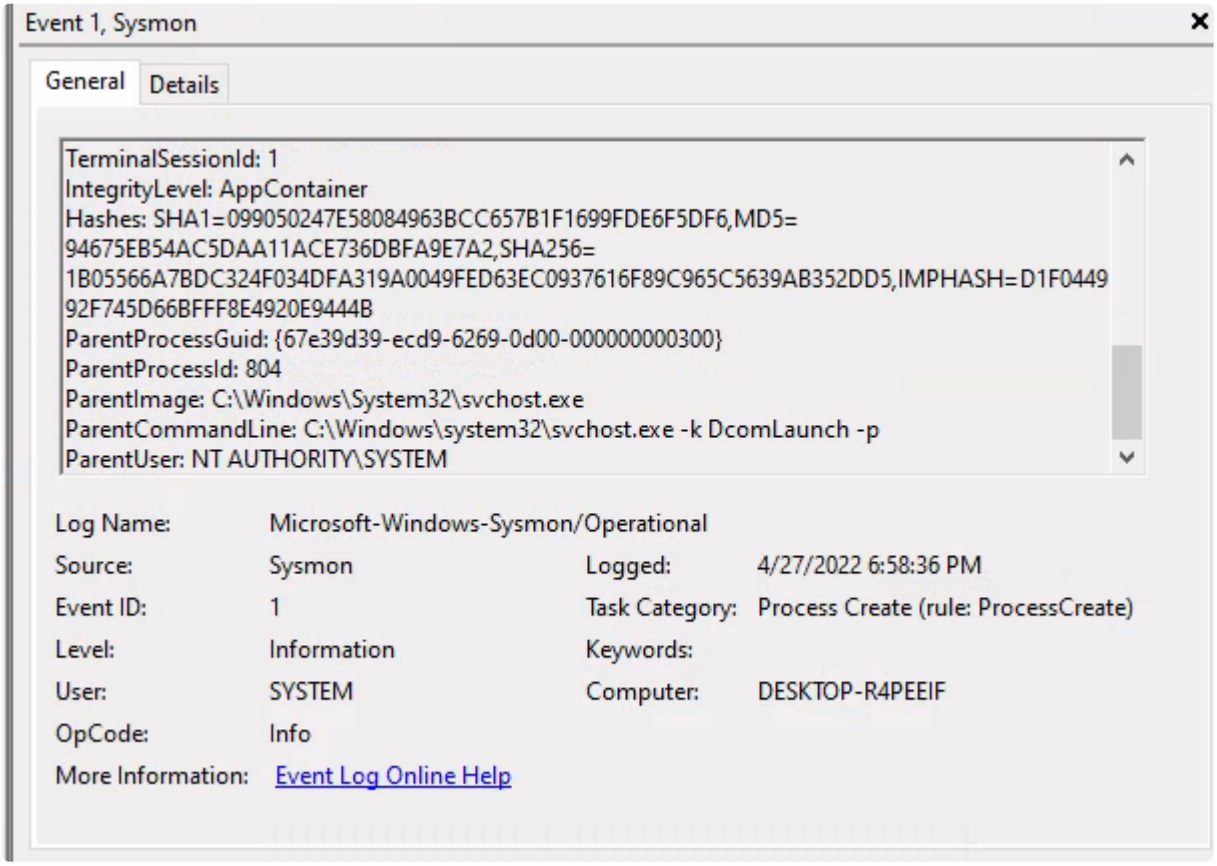
-> It is 1.

-> We now search with the following xml query

```
<Select Path="file://C:\Logs\PowershellExec\PowershellExec.evtx">*
[EventData[Data[@Name='Image']='C:\Program
Files\WindowsApps\Microsoft.WindowsCalculator_10.1906.55.0_x64__8wekyb3d
8bbwe\Calculator.exe']] and *[System[(EventID=1)]]</Select>
'C:\Program
Files\WindowsApps\Microsoft.WindowsCalculator_10.1906.55.0_x64__8wekyb3d
8bbwe\Calculator.exe'
```



-> Looking at the parent process section, we get that



-> We see that it is svchost.exe that created `Calculator.exe`, but we keep in mind the possibility of Parent PID spoofing technique that could be utilised.

-> However we're at an dead end here, so we do some research and consult the sysmon documentation and we see an interesting Event Id 8 that might be relevant here

Event ID 8: CreateRemoteThread

The `CreateRemoteThread` event detects when a process creates a thread in another process. This technique is used by malware to inject code and hide in other processes. The event indicates the source and target process. It gives information on the code that will be run in the new thread: `StartAddress`, `StartModule` and `StartFunction`. Note that `StartModule` and `StartFunction` fields are inferred, they might be empty if the starting address is outside loaded modules or known exported functions.

Event ID 9: RawAccessRead

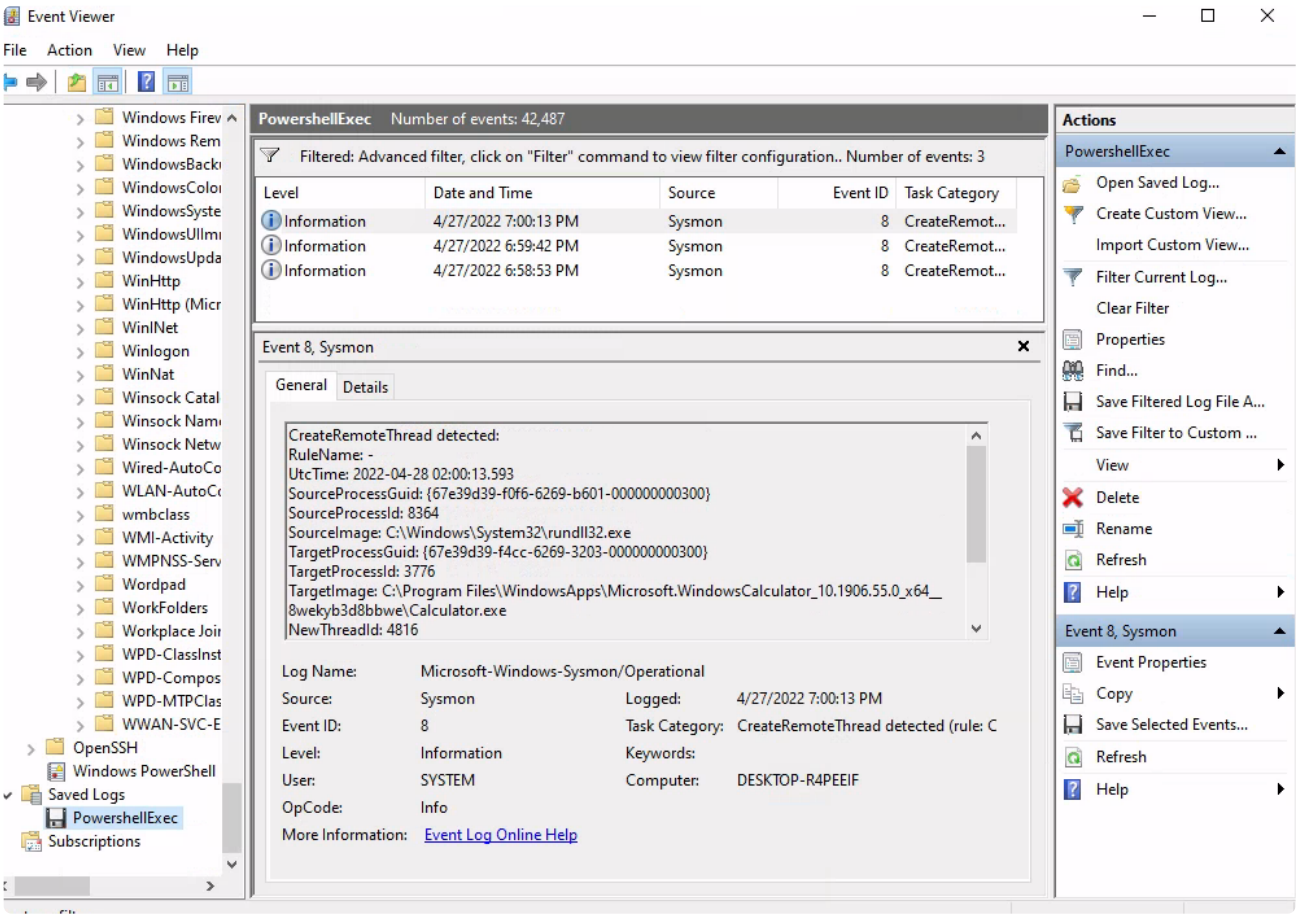
The `RawAccessRead` event detects when a process conducts reading operations from the drive using the `\\.\` denotation. This technique is often used by malware for data exfiltration of files that are locked for reading, as well as to avoid file access auditing tools. The event indicates the source process and target device.

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.

-> Let's filter that Event ID and see what results it give us, as well as getting a hang of the structure of the query:

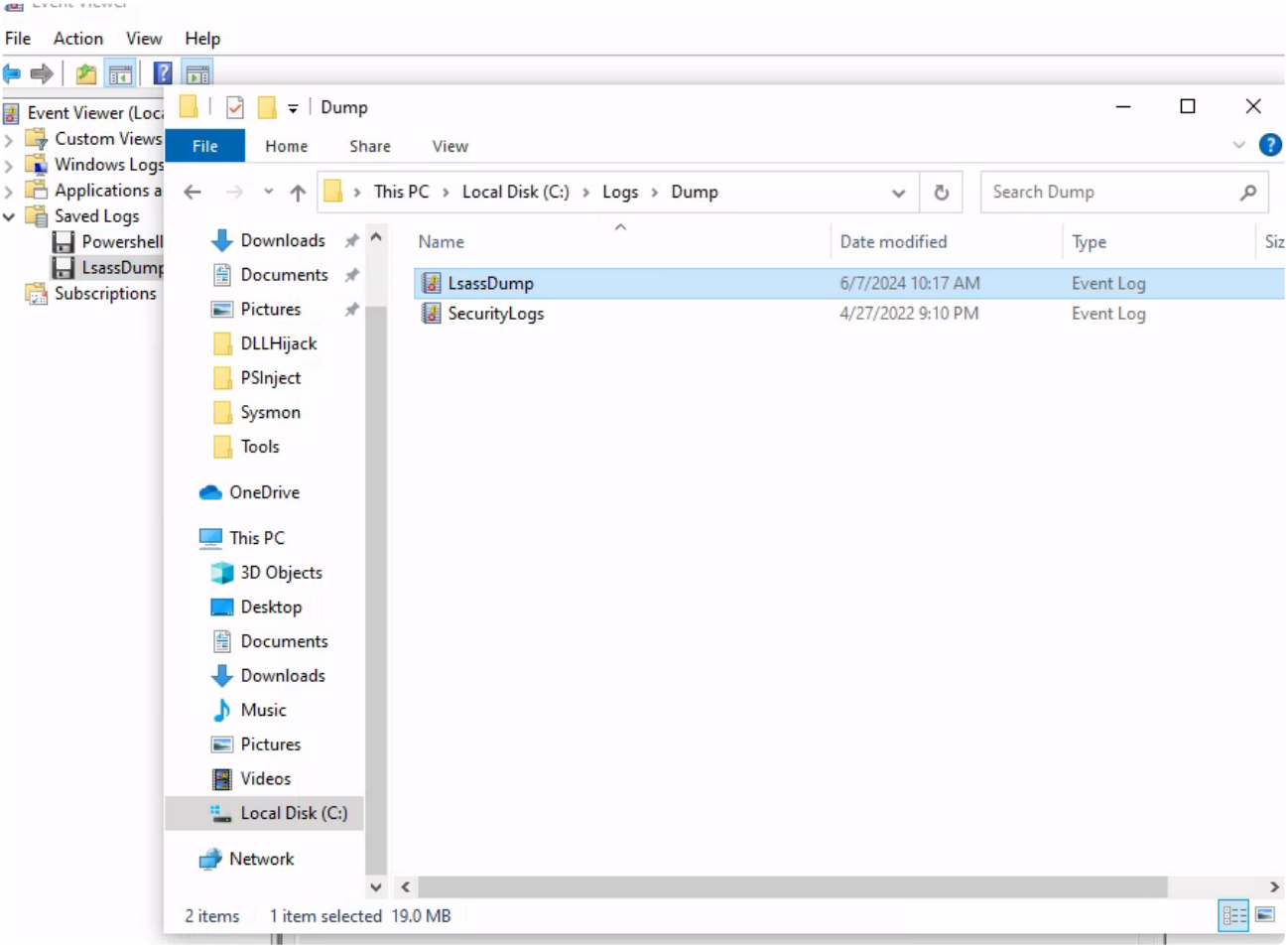
```
<Select Path="file://C:\Logs\PowershellExec\PowershellExec.evtx">*
[System[(EventID=8)]]
</Select>
```



-> we see that the parent of the process seems to be `rundll32.exe` and creates a remote thread (set of process registers for a process, including instruction pointer, stack-pointer) on to the target image `calculator.exe`.
-> Hence, the answer is `rundll32.exe`.

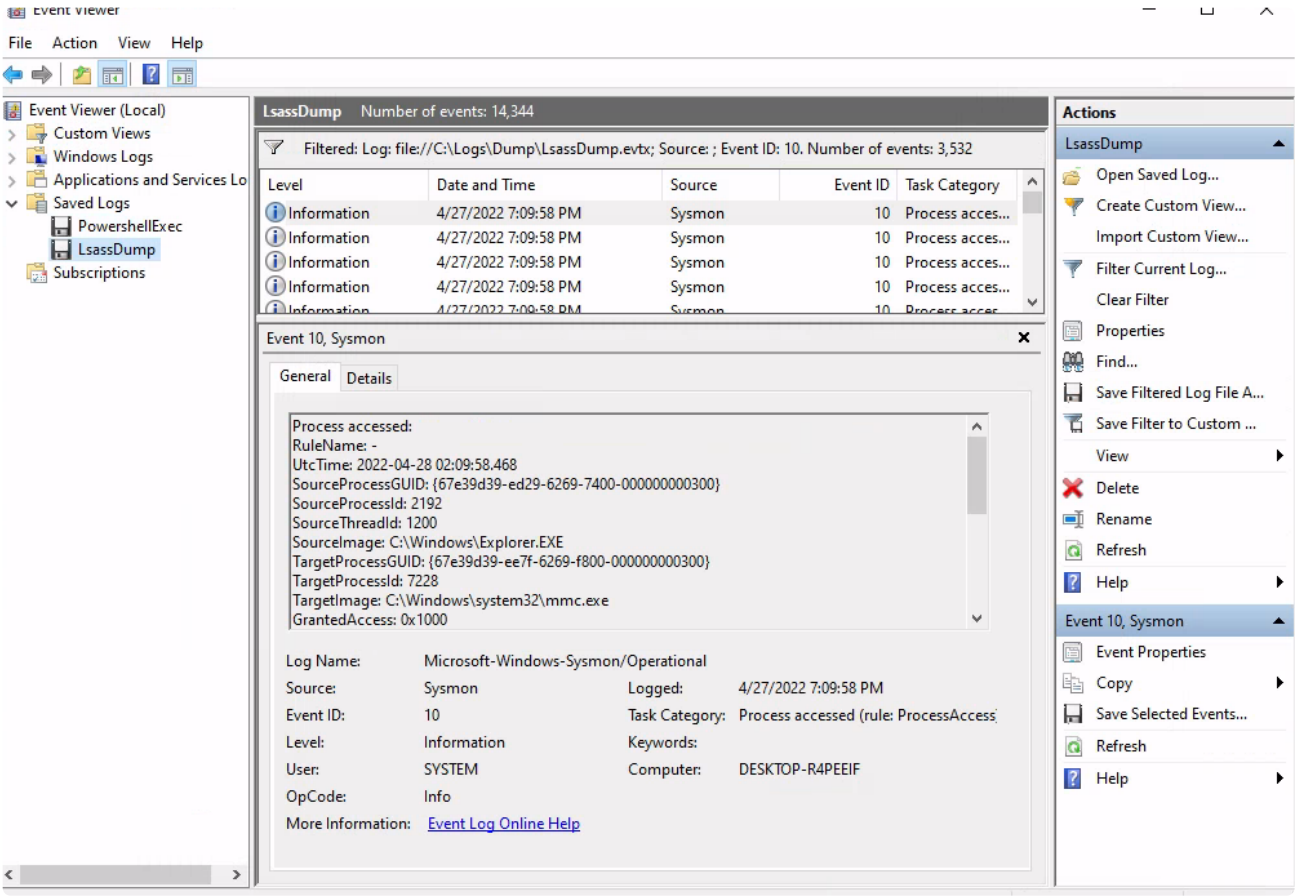
- By examining the logs located in the "C:\Logs\Dump" directory, determine the process that performed an LSASS dump. Enter the process name as your answer. Answer format: _.exe

-> we first open the log



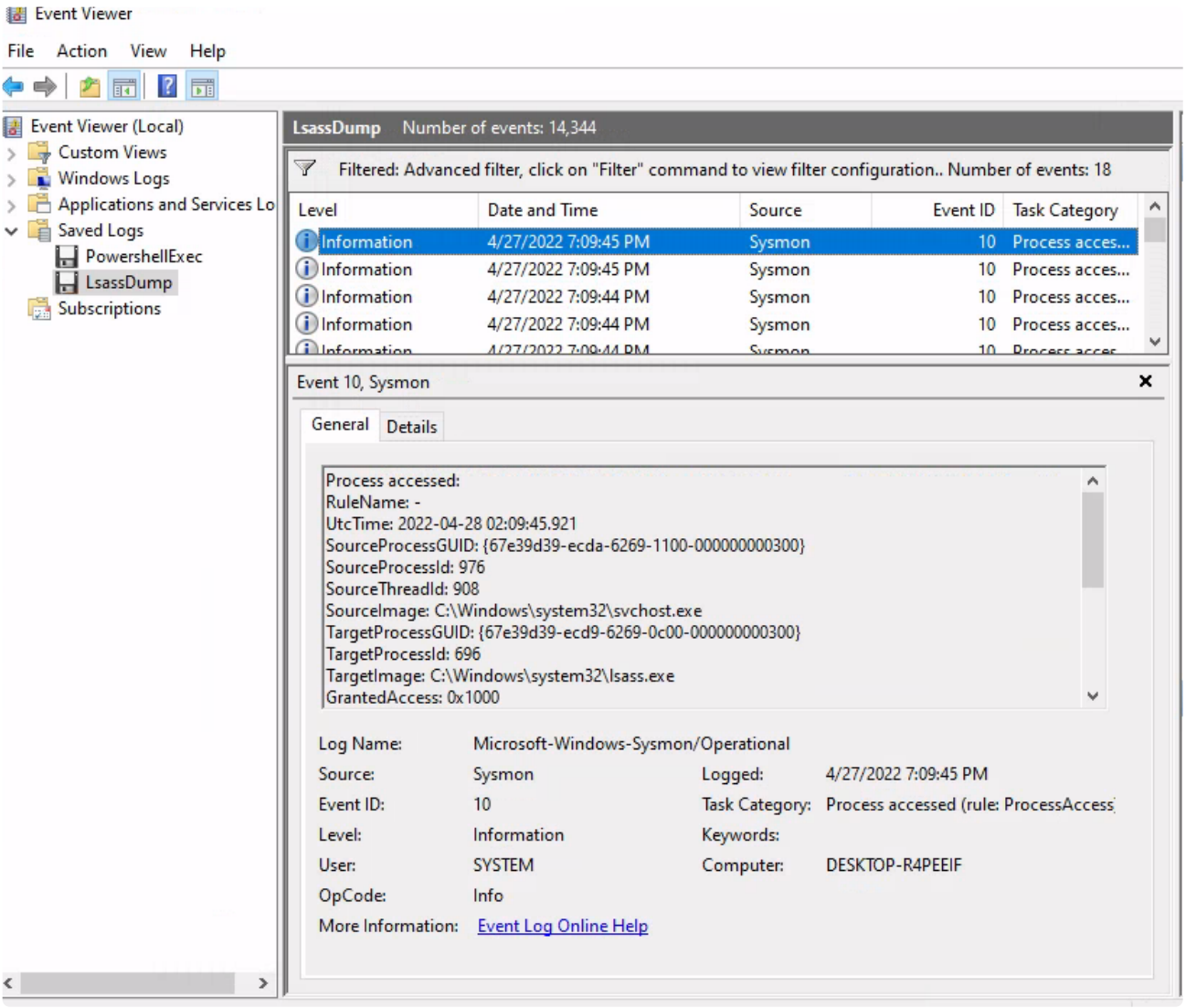
-> We see that there are two log files, LsassDump will probably of interest.

-> We filter of EventID of 10:



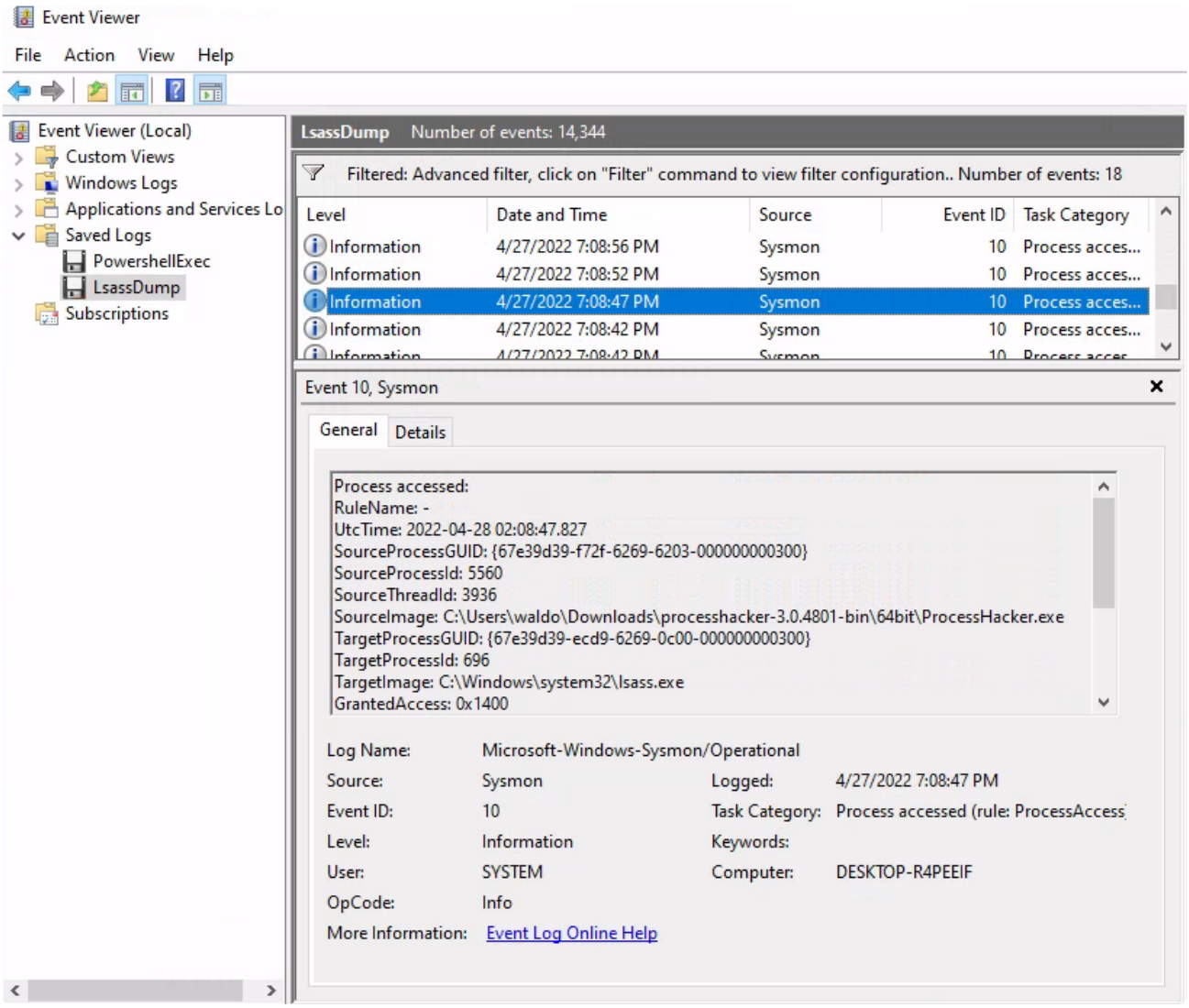
-> There are still too much logs, so we filter it again on the target `lsass.exe`

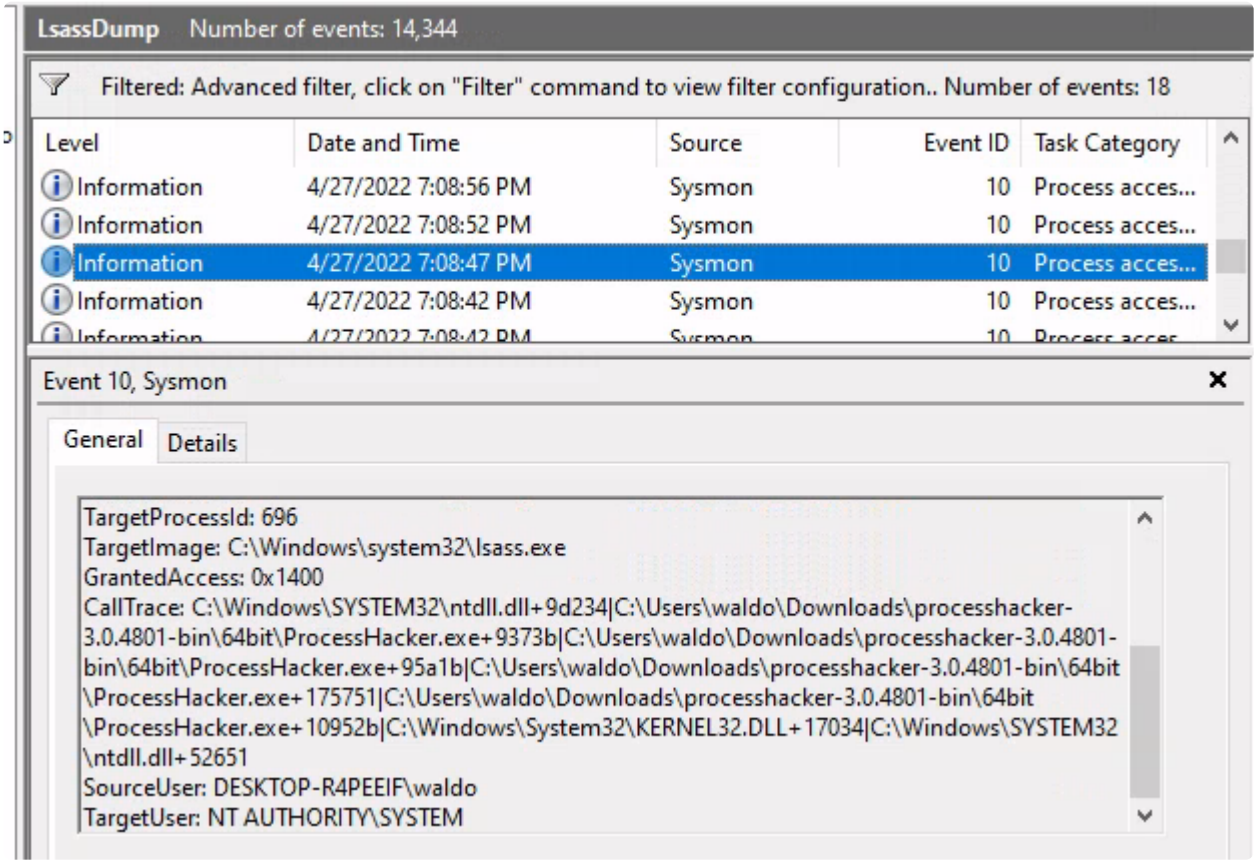
```
<Select Path="file://C:\Logs\Dump\LsassDump.evtx">*
[System[(EventID=10)]] and *
[EventData[Data[@Name='TargetImage']='C:\Windows\system32\lsass.exe']]
</Select>
```



-> We now have 18 logs, much less than before and we can analyse it.

-> We see an specifically suspicious log here

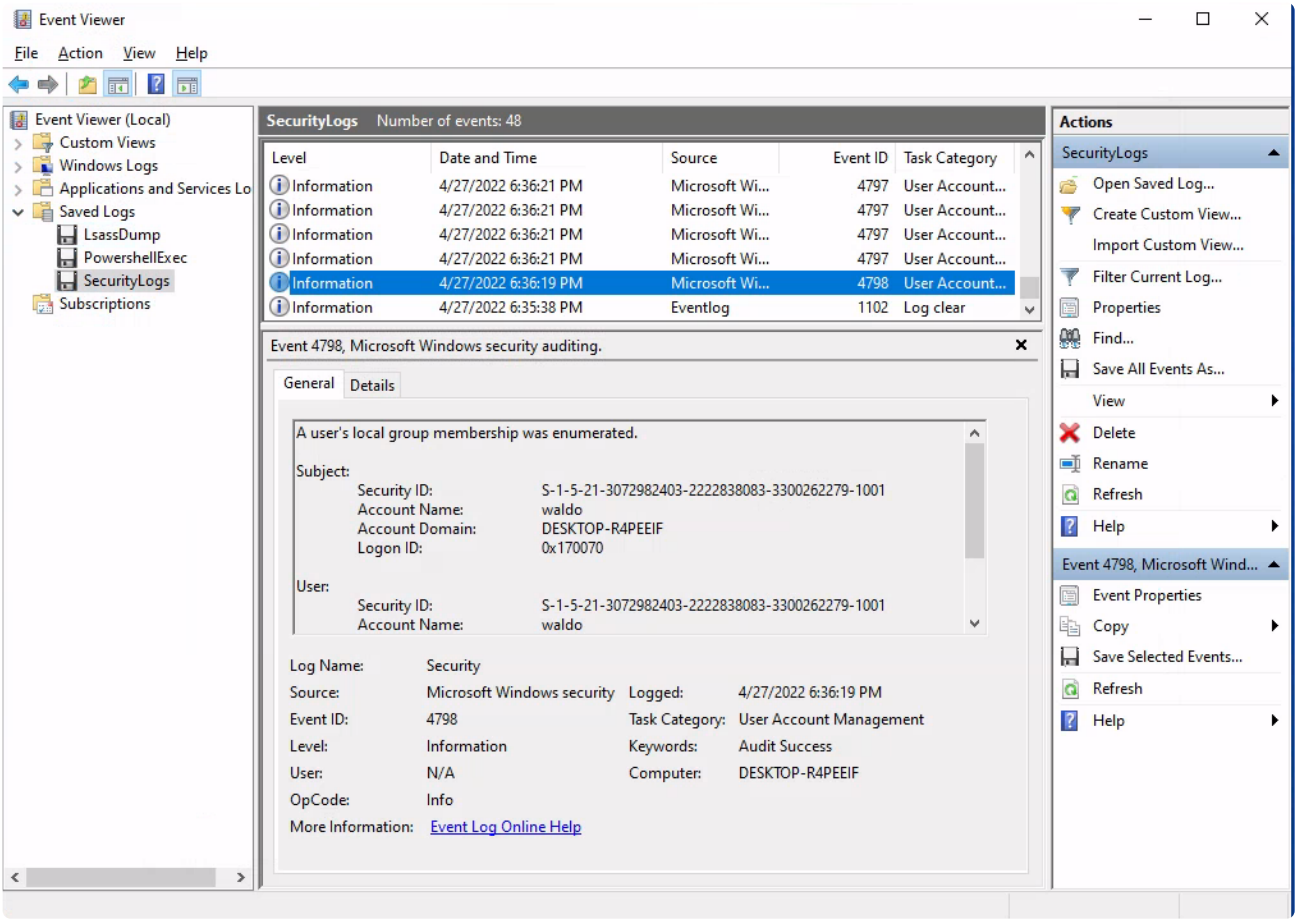

Event Viewer
File Action View Help
Event Viewer (Local)
Custom Views
Windows Logs
Applications and Services Lo
Saved Logs
PowershellExec
LsassDump
Subscriptions
LsassDump Number of events: 14,344
Filtered: Advanced filter, click on "Filter" command to view filter configuration.. Number of events: 18
Level Date and Time Source Event ID Task Category
Information 4/27/2022 7:08:56 PM Sysmon 10 Process acces...
Information 4/27/2022 7:08:52 PM Sysmon 10 Process acces...
Information 4/27/2022 7:08:47 PM Sysmon 10 Process acces...
Information 4/27/2022 7:08:42 PM Sysmon 10 Process acces...
Event 10, Sysmon
General Details
Process accessed:
RuleName: -
UtcTime: 2022-04-28 02:08:47.827
SourceProcessGUID: {67e39d39-f72f-6269-6203-000000000300}
SourceProcessId: 5560
SourceThreadId: 3936
SourceImage: C:\Users\waldo\Downloads\processhacker-3.0.4801-bin\64bit\ProcessHacker.exe
TargetProcessGUID: {67e39d39-ecd9-6269-0c00-000000000300}
TargetProcessId: 696
TargetImage: C:\Windows\system32\lsass.exe
GrantedAccess: 0x1400
Log Name: Microsoft-Windows-Sysmon/Operational
Source: Sysmon Logged: 4/27/2022 7:08:47 PM
Event ID: 10 Task Category: Process accessed (rule: ProcessAccess
Level: Information Keywords:
User: SYSTEM Computer: DESKTOP-R4PEEIF
OpCode: Info
More Information: Event Log Online Help



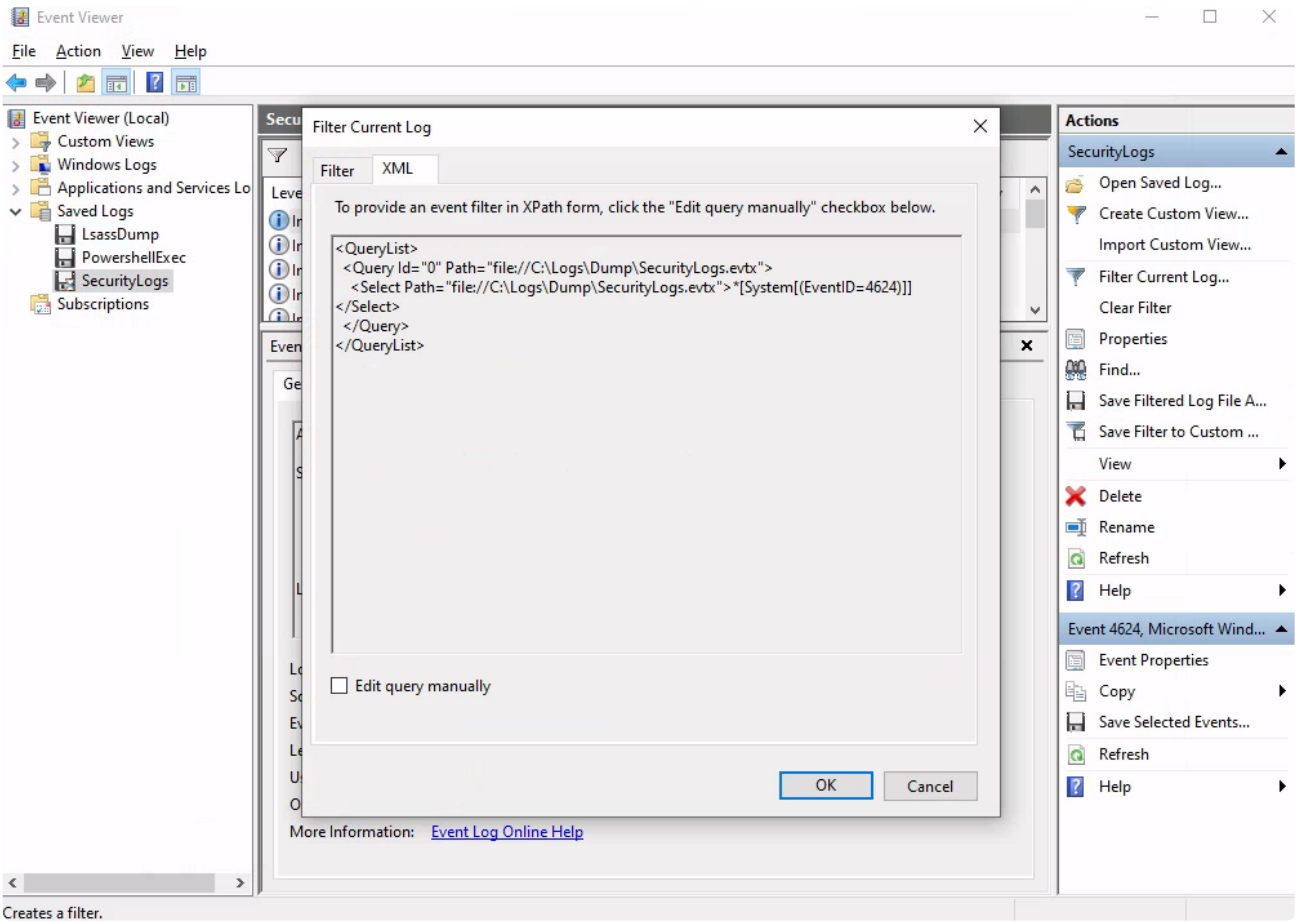
-> The source is different from the target and is calling from an random directory.
-> waldo user also shouldn't have or need to access lsass for any reason.
-> hence, this is the process performing an lsass dump

- By examining the logs located in the "C:\Logs\Dump" directory, determine if an ill-intended login took place after the LSASS dump. Answer format: Yes or No

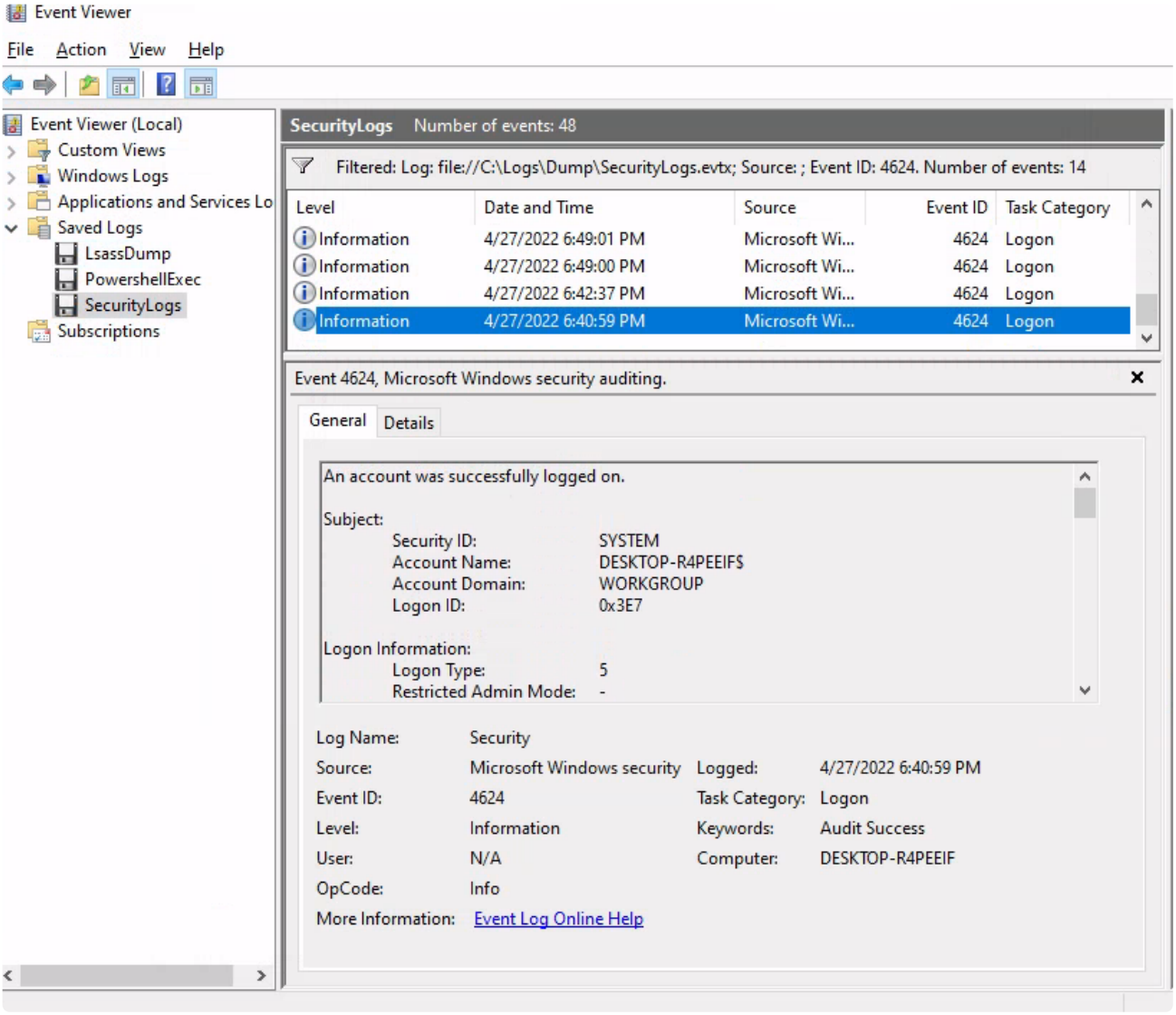
-> We first look at the Security Logs in the folder

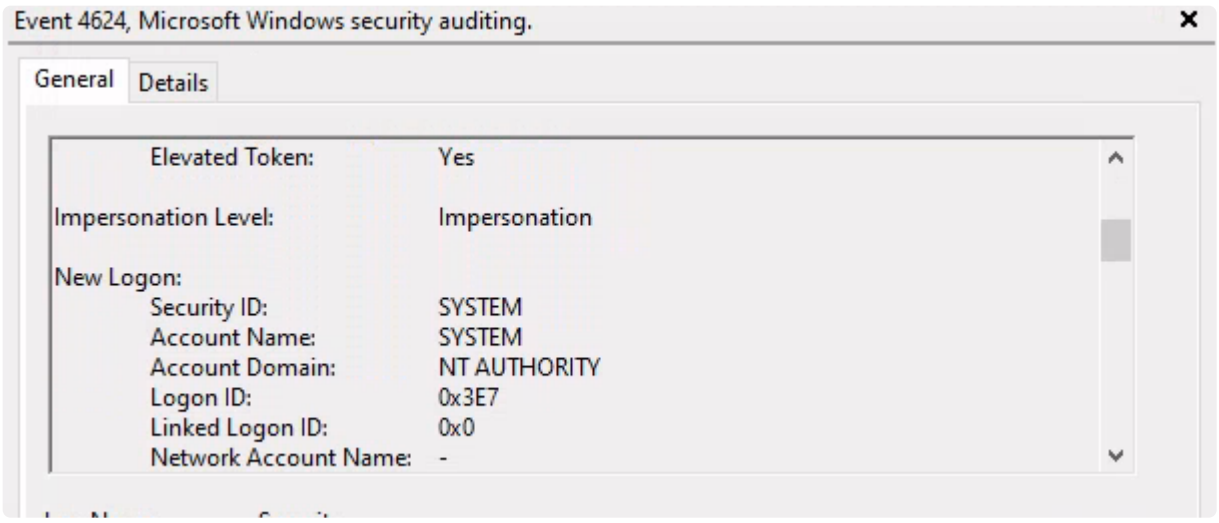


-> We would look at successful attempt to verify the action, filtering EventID 4624



-> Looking at the logs, we see that:



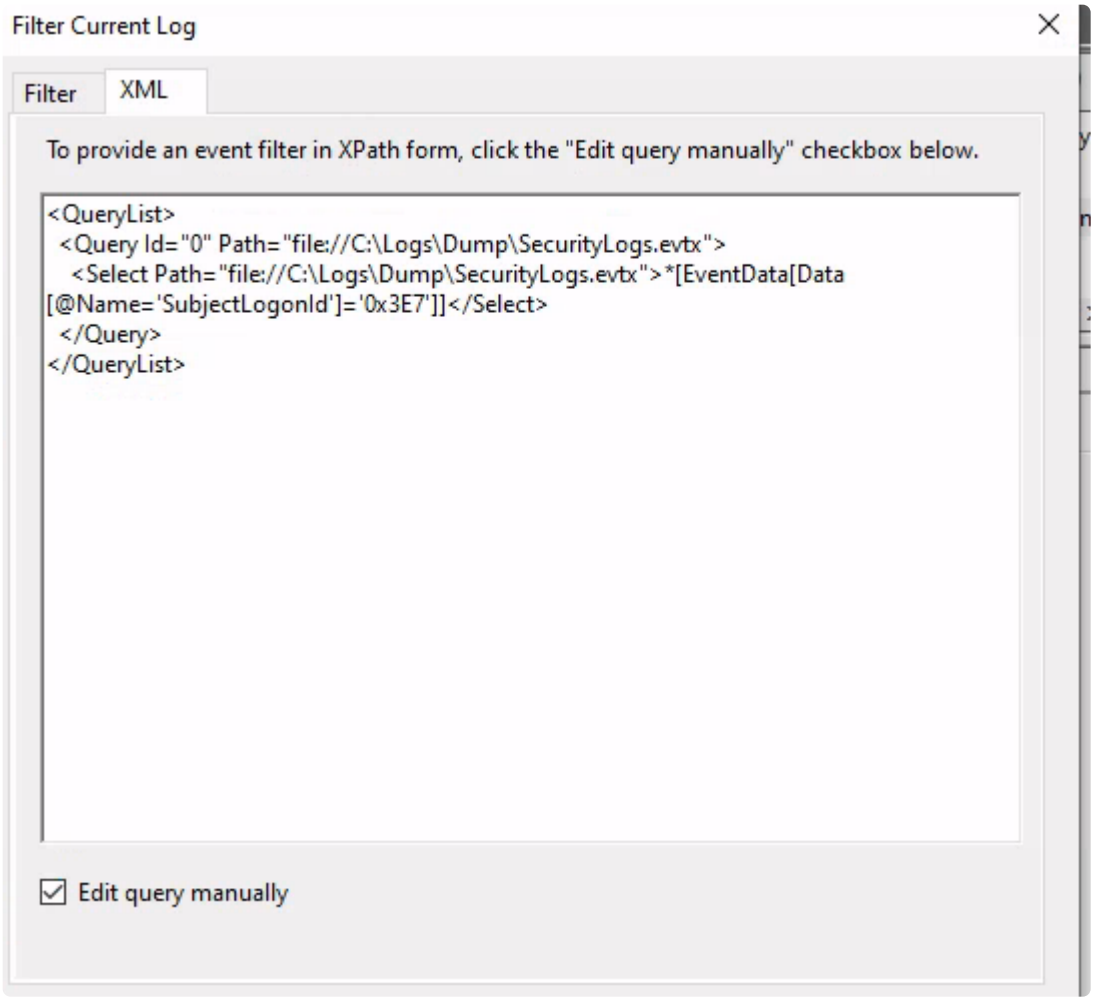


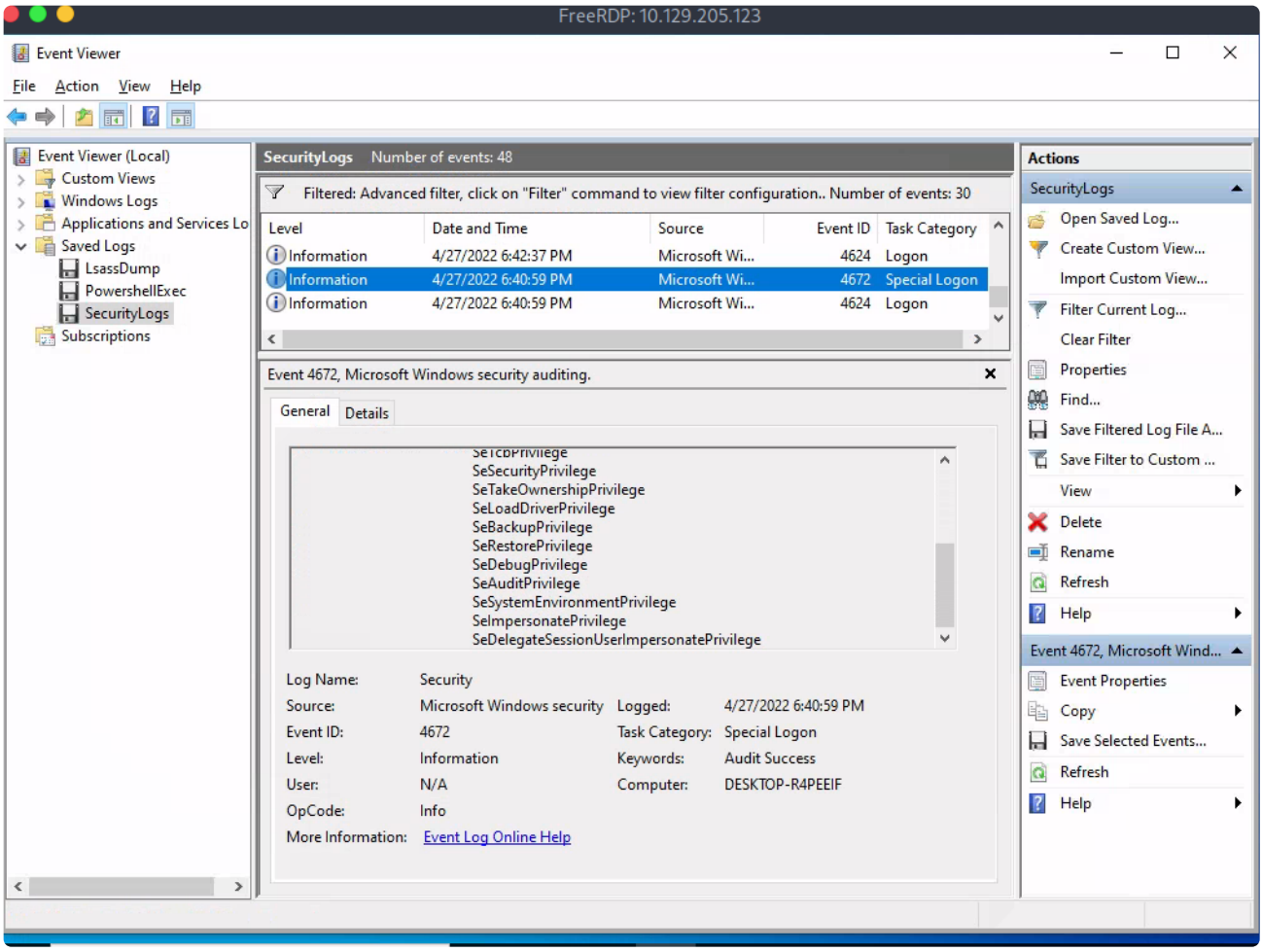
-> An machine account logged on, with elevated privileges and with logon type 5 (uncommon and it represents an new services have been initiated).

-> We also see a lot of logon types 5 in this short span of time period.

-> Hence, the computer is fully compromised and immediate action is needed.

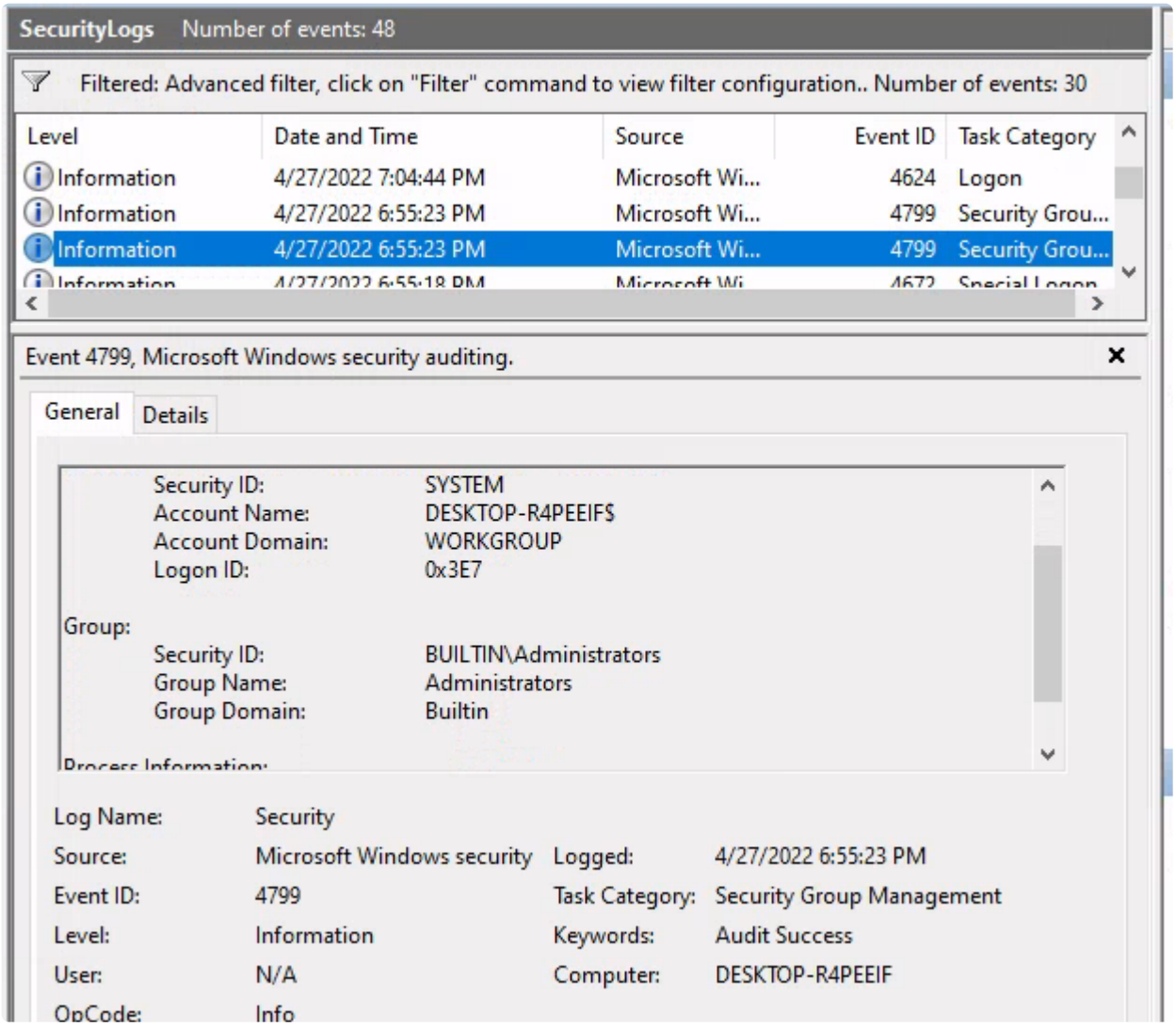
-> Filtering on the Logon ID, we see the following:

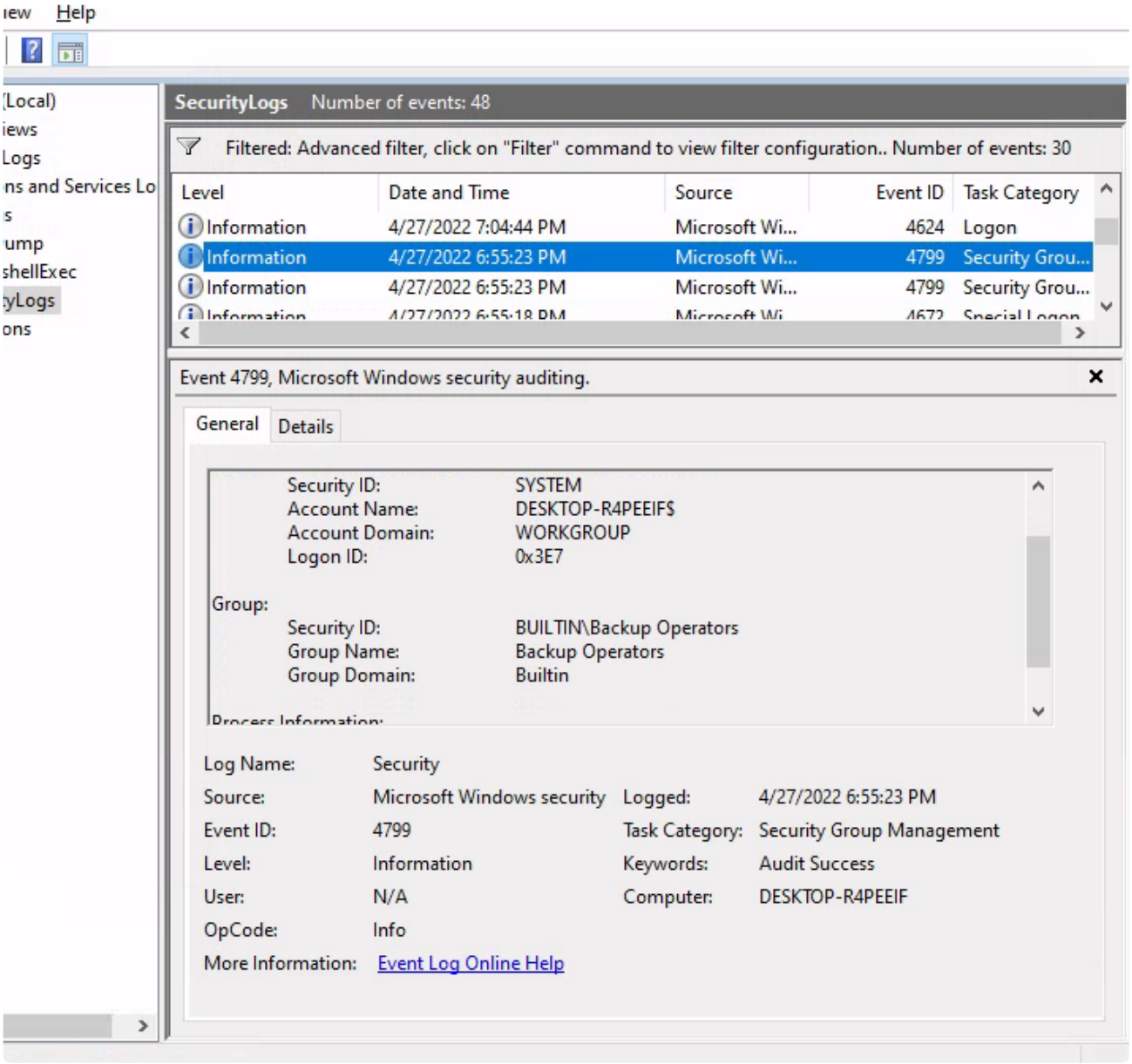

Filter Current Log
Filter
XML
To provide an event filter in XPath form, click the "Edit query manually" checkbox below.
<QueryList>
<Query Id="0" Path="file://C:\Logs\Dump\SecurityLogs.evtx">
<Select Path="file://C:\Logs\Dump\SecurityLogs.evtx">*[EventData[Data
[@Name='SubjectLogonId']='0x3E7']]</Select>
</Query>
</QueryList>
Edit query manually



-> We see a special logon permission is granted, with lots of privileges granted.

-> Furthermore, looking at the log, we see that

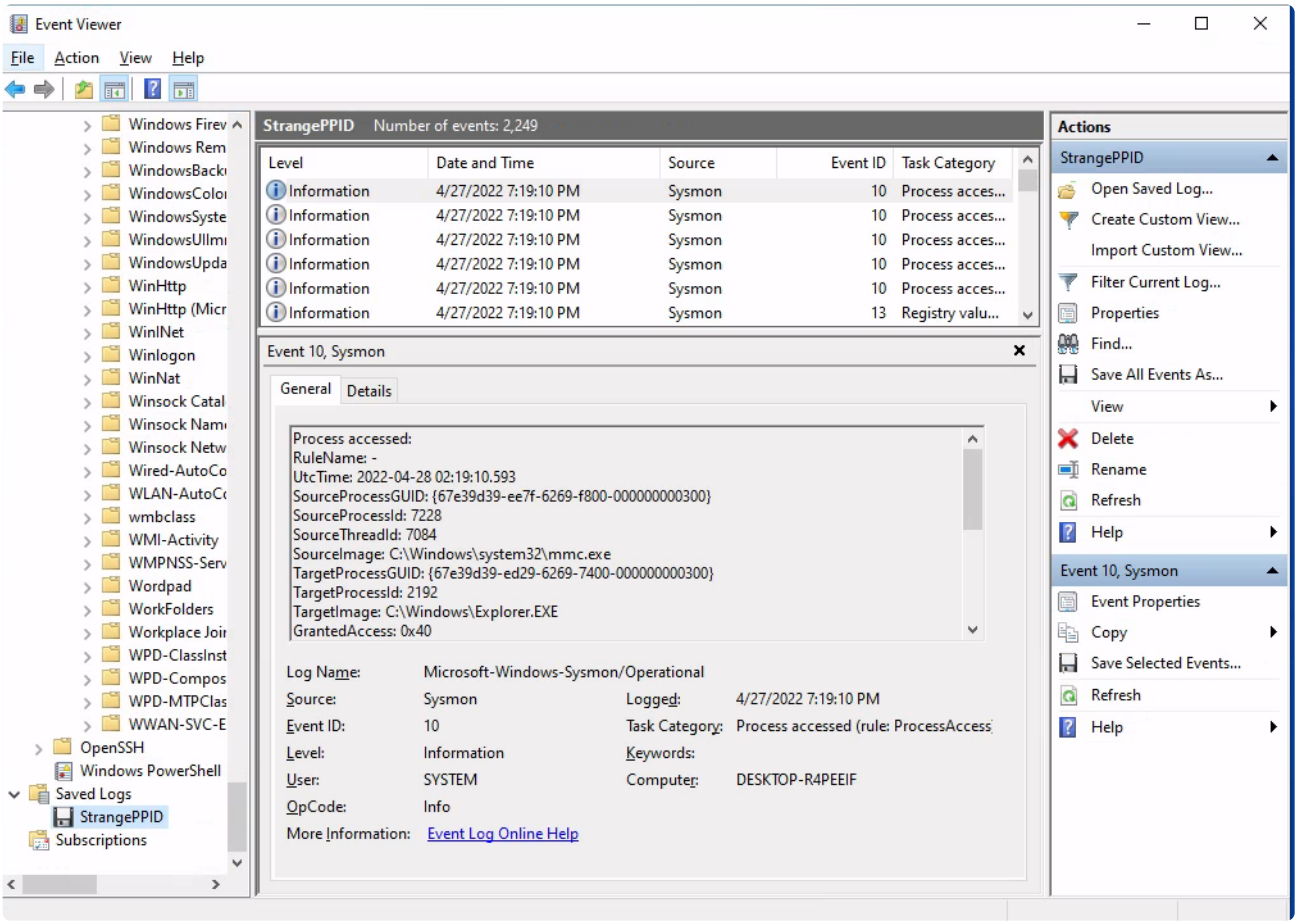

SecurityLogs Number of events: 48
Filtered: Advanced filter, click on "Filter" command to view filter configuration.. Number of events: 30
Level Date and Time Source Event ID Task Category
Information 4/27/2022 7:04:44 PM Microsoft Wi... 4624 Logon
Information 4/27/2022 6:55:23 PM Microsoft Wi... 4799 Security Grou...
Information 4/27/2022 6:55:23 PM Microsoft Wi... 4799 Security Grou...
Event 4799, Microsoft Windows security auditing.
General Details
Security ID: SYSTEM
Account Name: DESKTOP-R4PEEIF$
Account Domain: WORKGROUP
Logon ID: 0x3E7
Group:
Security ID: BUILTIN\Administrators
Group Name: Administrators
Group Domain: Builtin
Log Name: Security
Source: Microsoft Windows security
Logged: 4/27/2022 6:55:23 PM
Event ID: 4799
Task Category: Security Group Management
Level: Information
Keywords: Audit Success
User: N/A
Computer: DESKTOP-R4PEEIF
OpCode: Info



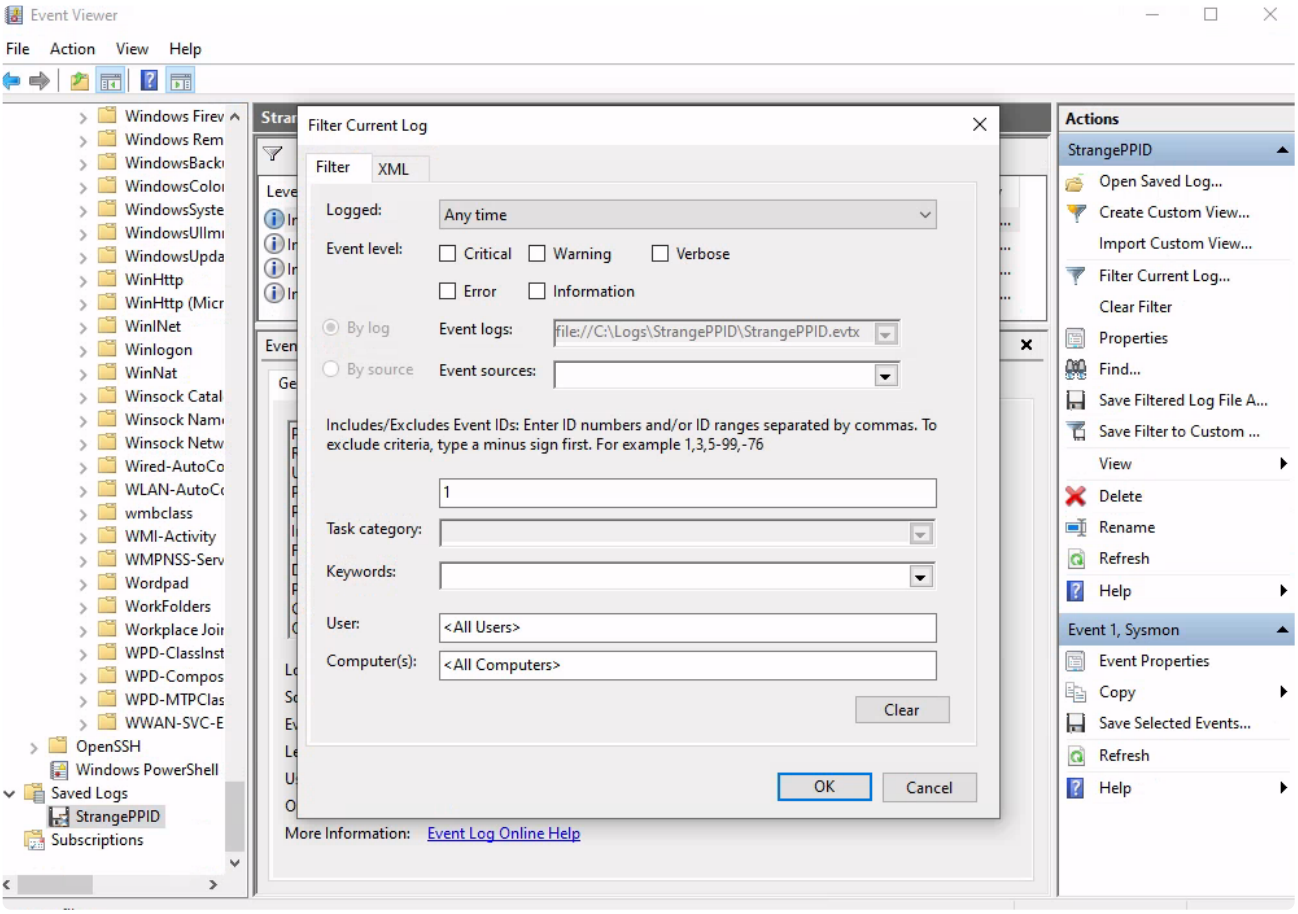
-> The administrator and backup operator group of the domain are enumerated, indicating attackers progress towards full domain compromise.
-> However, we can see that nothing "ill-intended" (like changing group policy or settings...) has been performed just yet, so the answer would be no, which seems counter-intuitive (though there is an login and it is enumerating stuff, maybe it has not found the way to compromise the domain yet).

- By examining the logs located in the "C:\\Logs\StrangePPID" directory, determine a process that was used to temporarily execute code based on a strange parent-child relationship. Enter the process name as your answer. Answer format: _.exe

-> We first examine the logs.

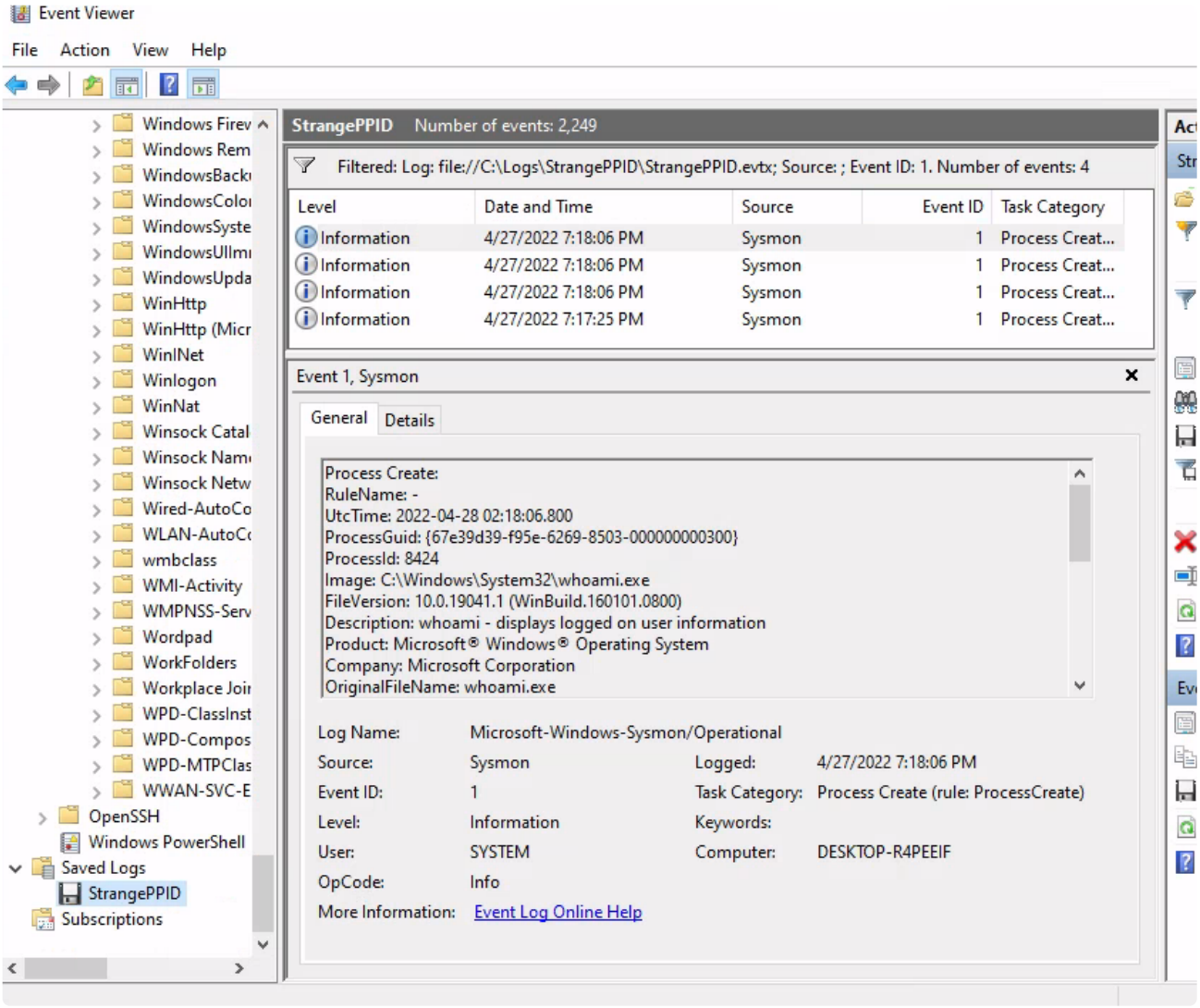


-> Now, since that a process is used to execute code and has a strange parent-child relationship, a good try would be to look filter event id=1, as for Sysmon that means "process is created".

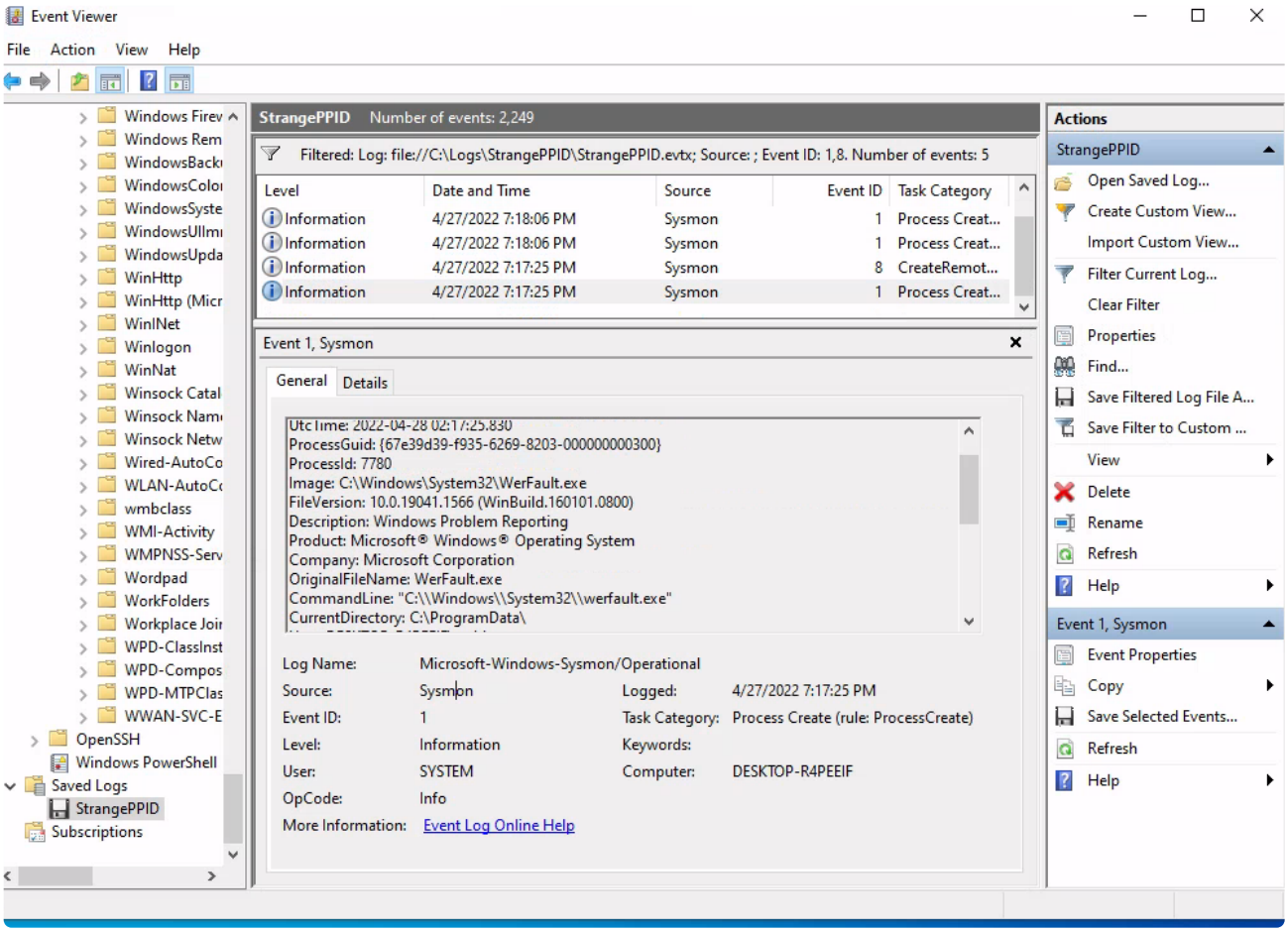


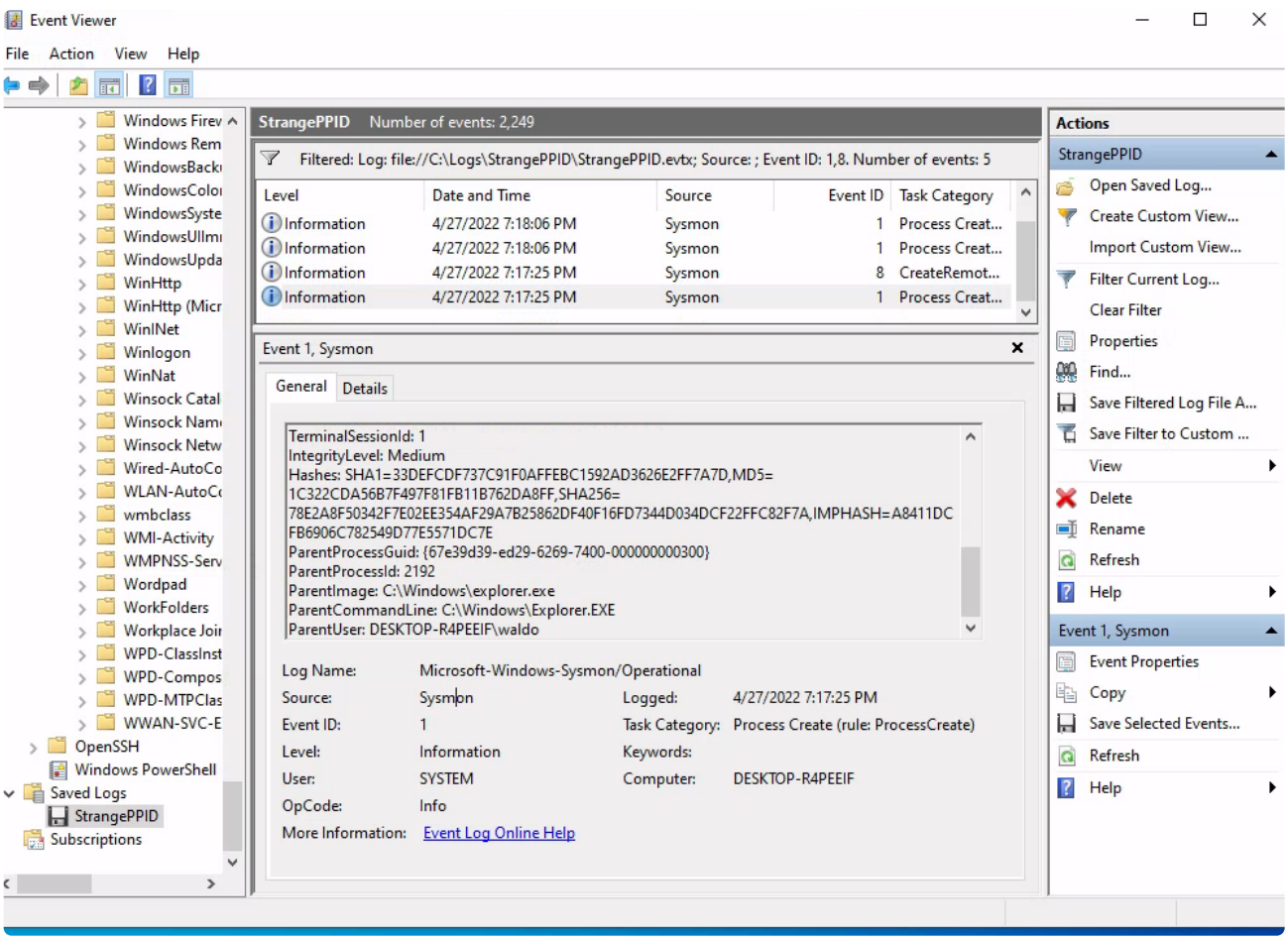
-> Filtering for event ID 1

-> We see the following logs:



-> Looking down at the log, we see the following:

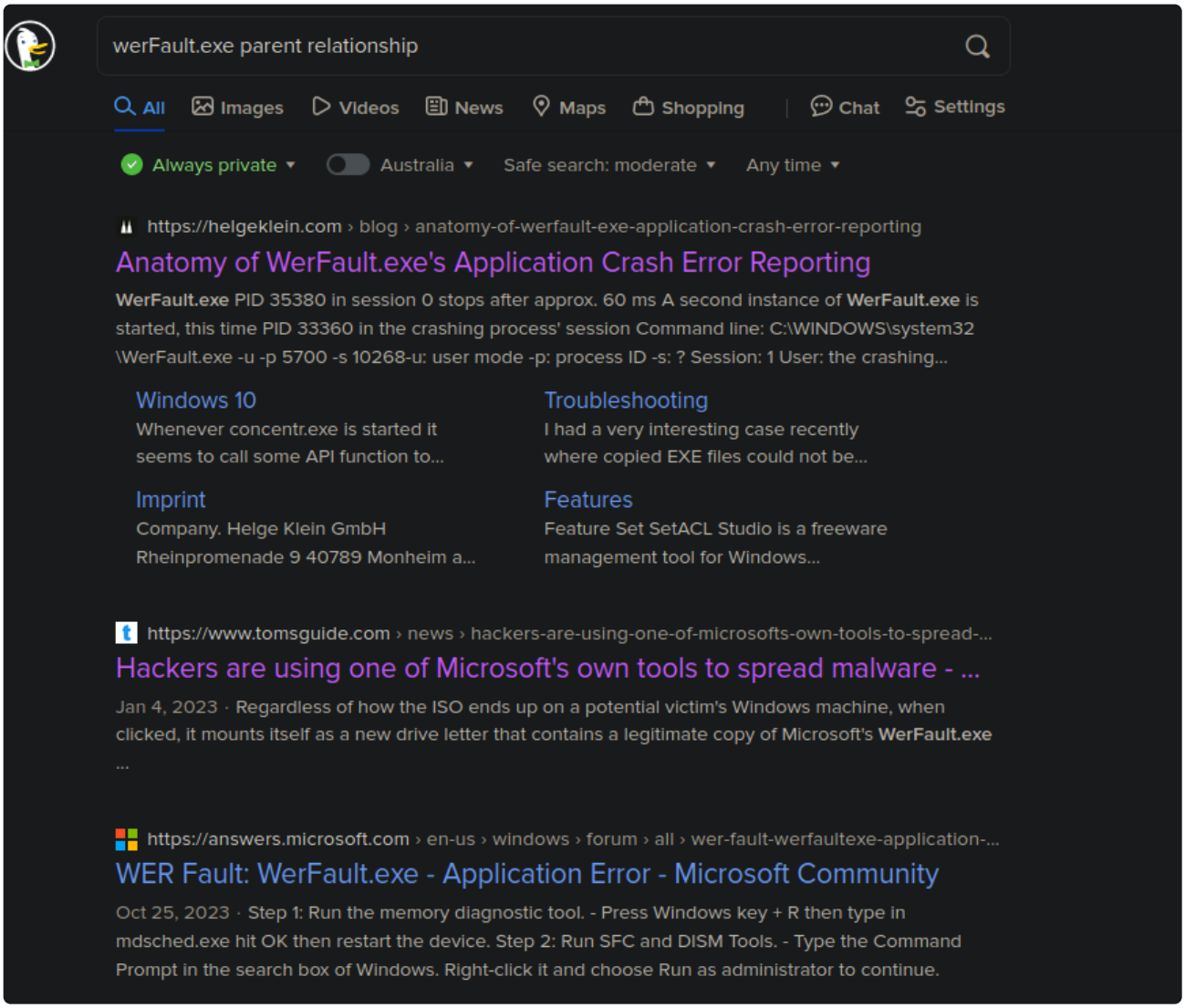




-> We know that by the child-parent relationship that Explorer.exe can start process

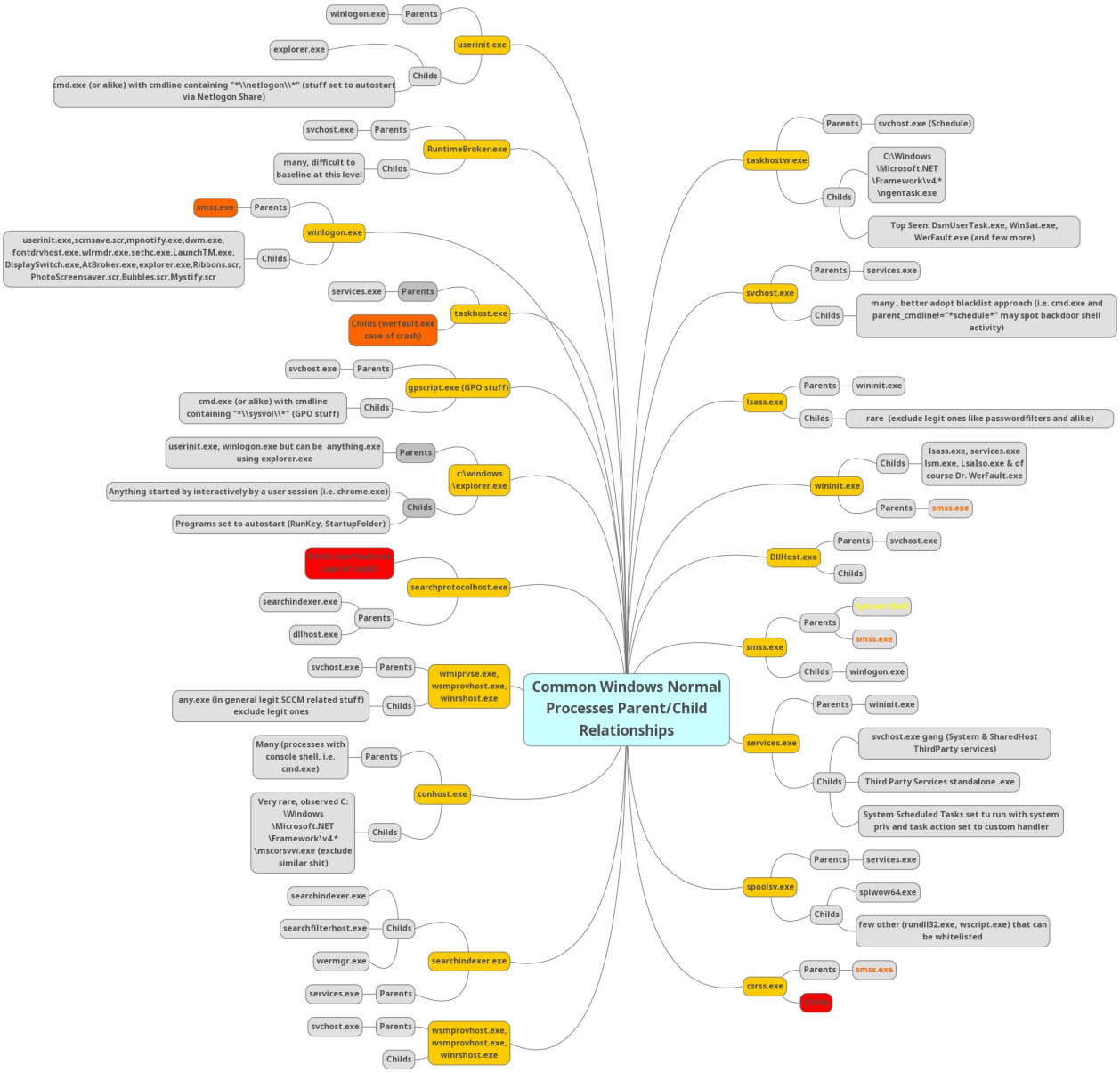
interactively using the user's session. However, werfault.exe is not something that can be started interactively as it is an error application for crash error reporting (it appears automatically when an error happens)

-> Furthermore, looking at the result of some searching, we can see how weFault.exe is potentially being abused by attackers:



-> Hence, the suspicious Parent-child relationship that is executing code is werfault.exe.

-> The common Windows normal Processs Parent/Child Relationships

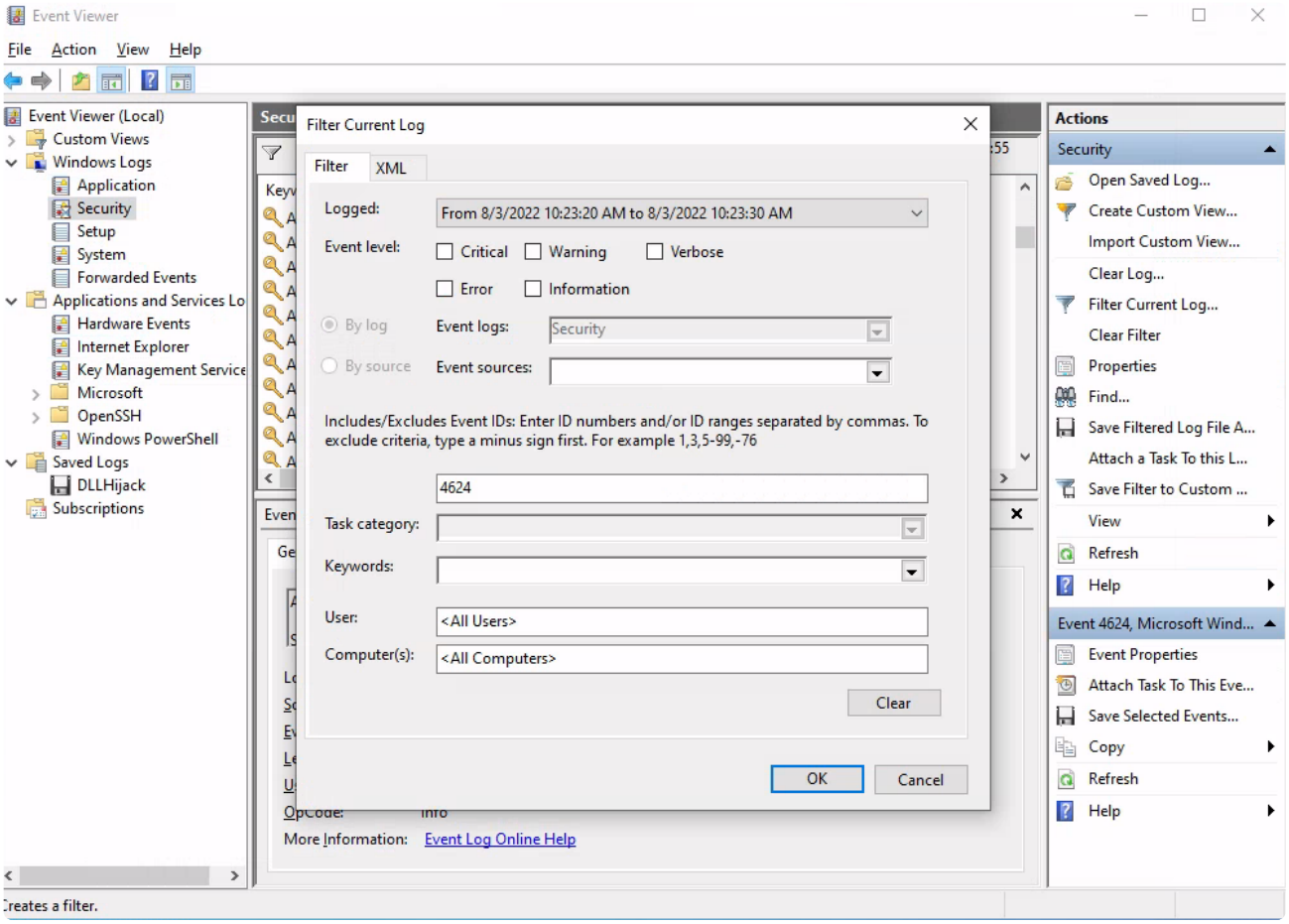


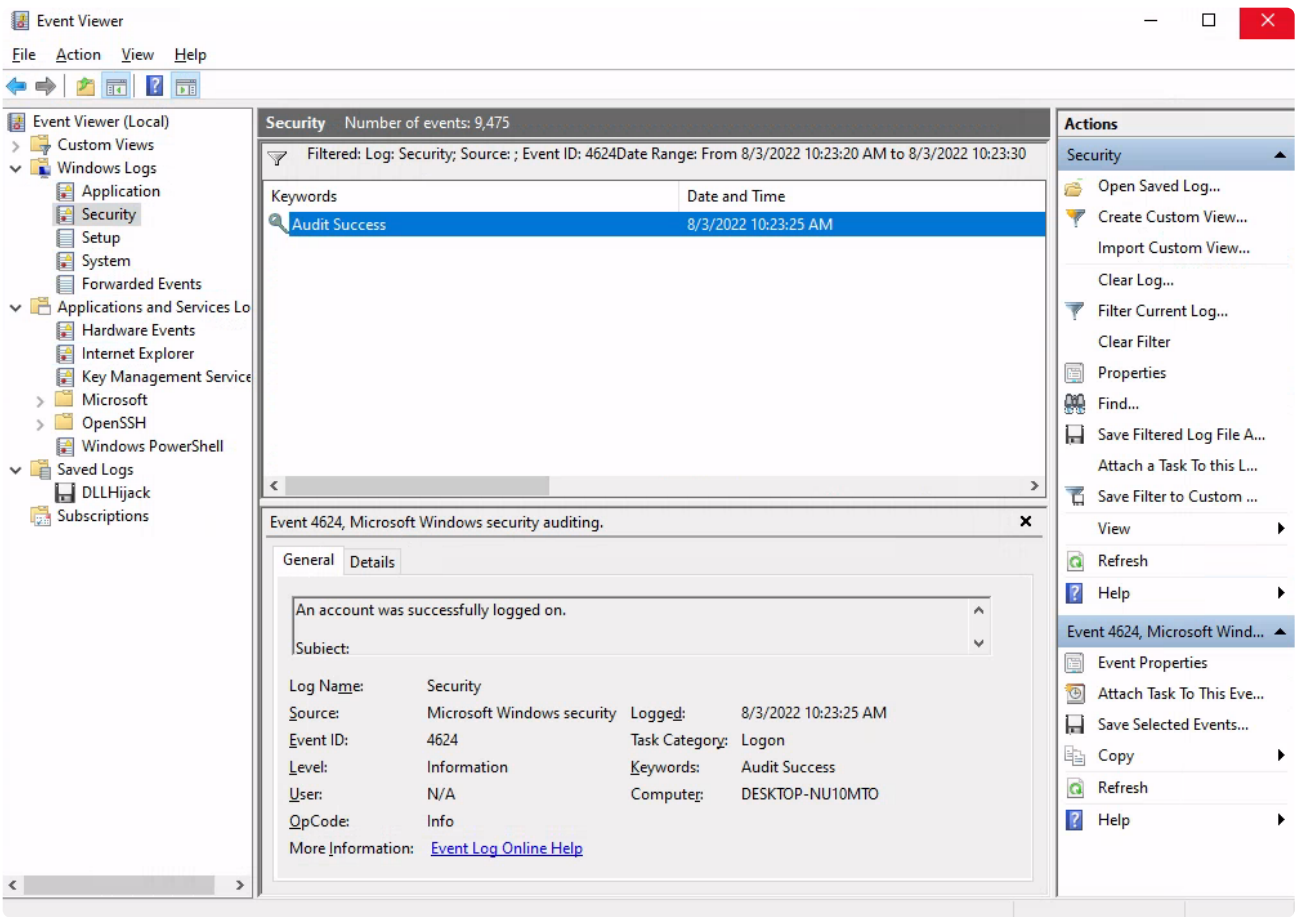
Windows Event Logs

Question

- Analyze the event with ID 4624, that took place on 8/3/2022 at 10:23:25. Conduct a similar investigation as outlined in this section and provide the name of the executable responsible for the modification of the auditing settings as your answer.

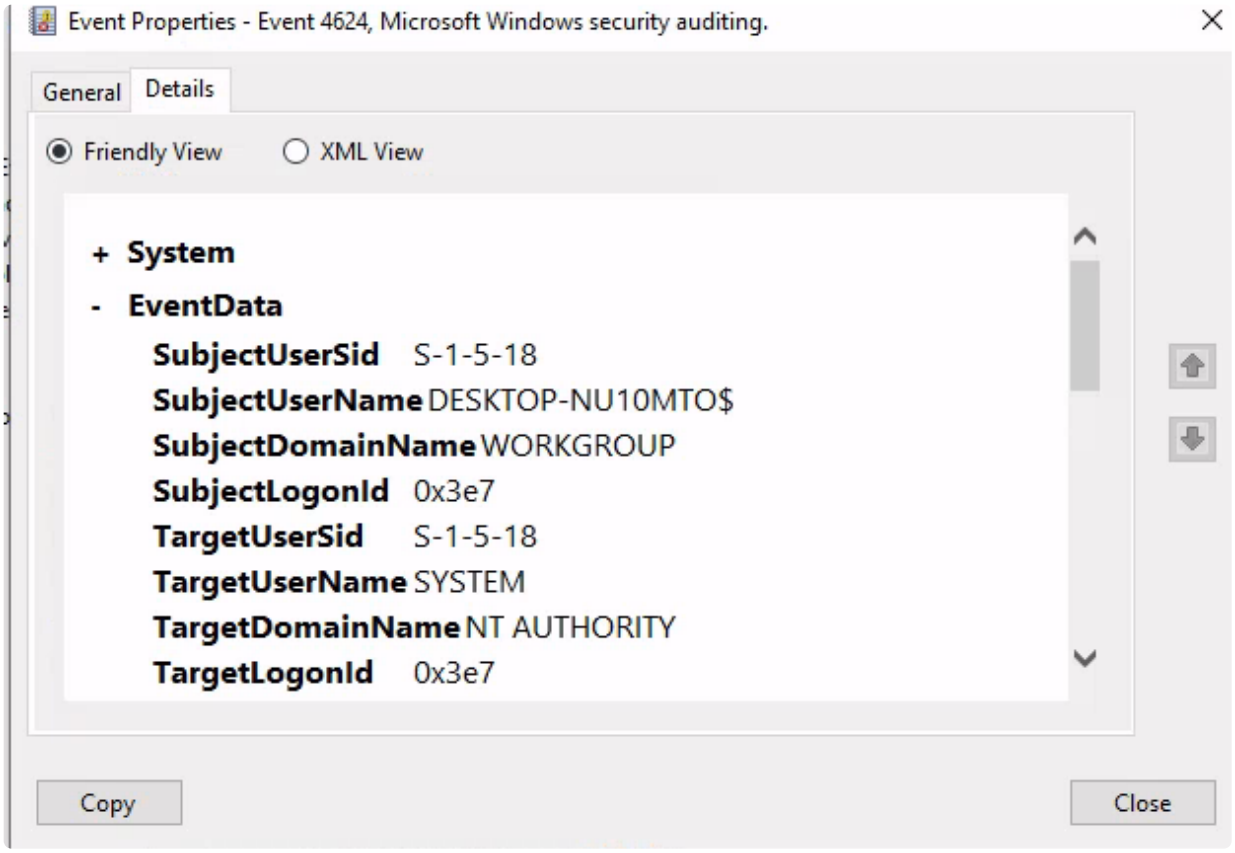
-> Go to security logs and filter logs





-> And we first obtain the log we need to investigate.

- Now we investigate the details of the log

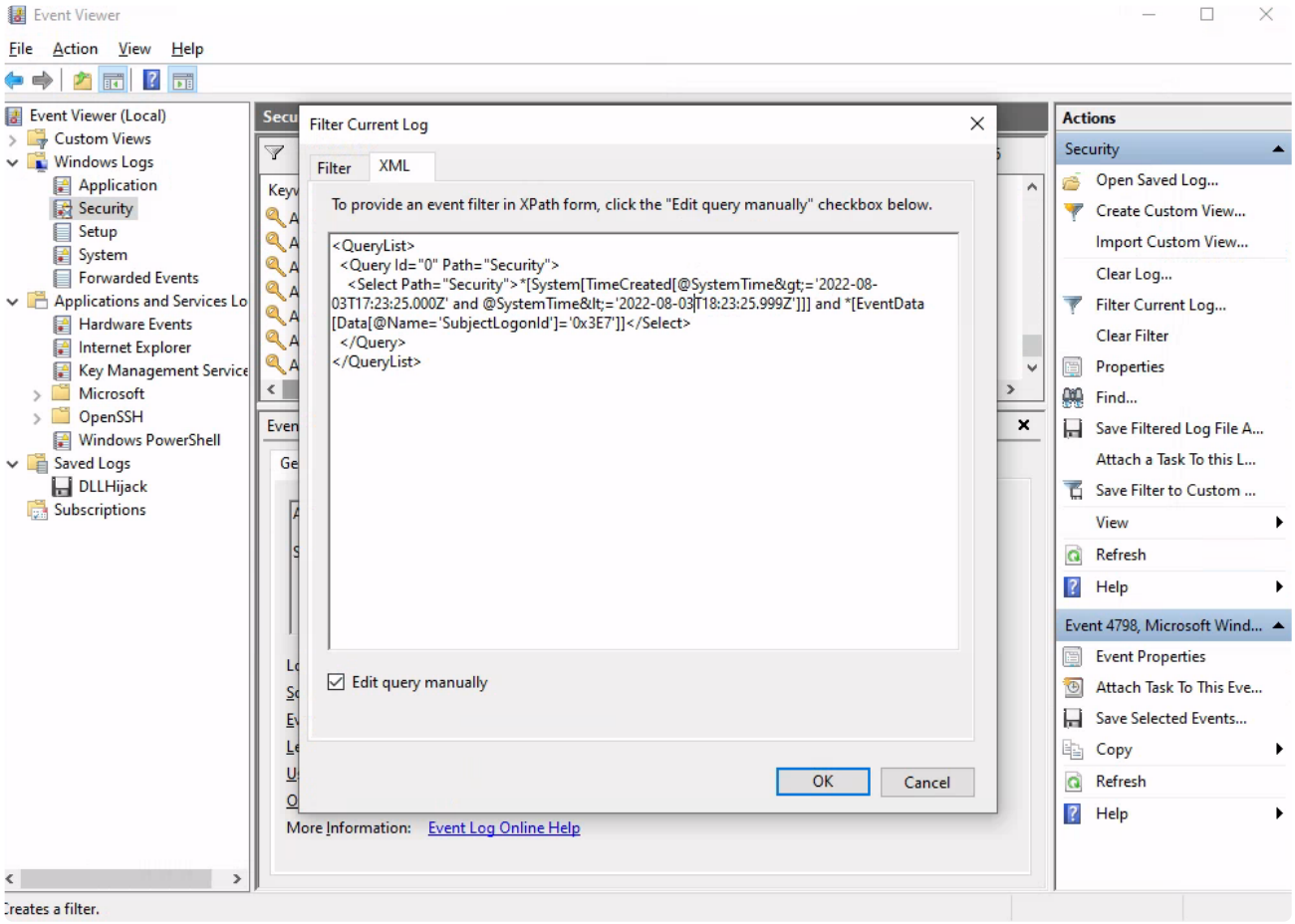


-> Next, we investigate the logs related to this logon id, `0x3e7` through modifying the `xml` view as shown below

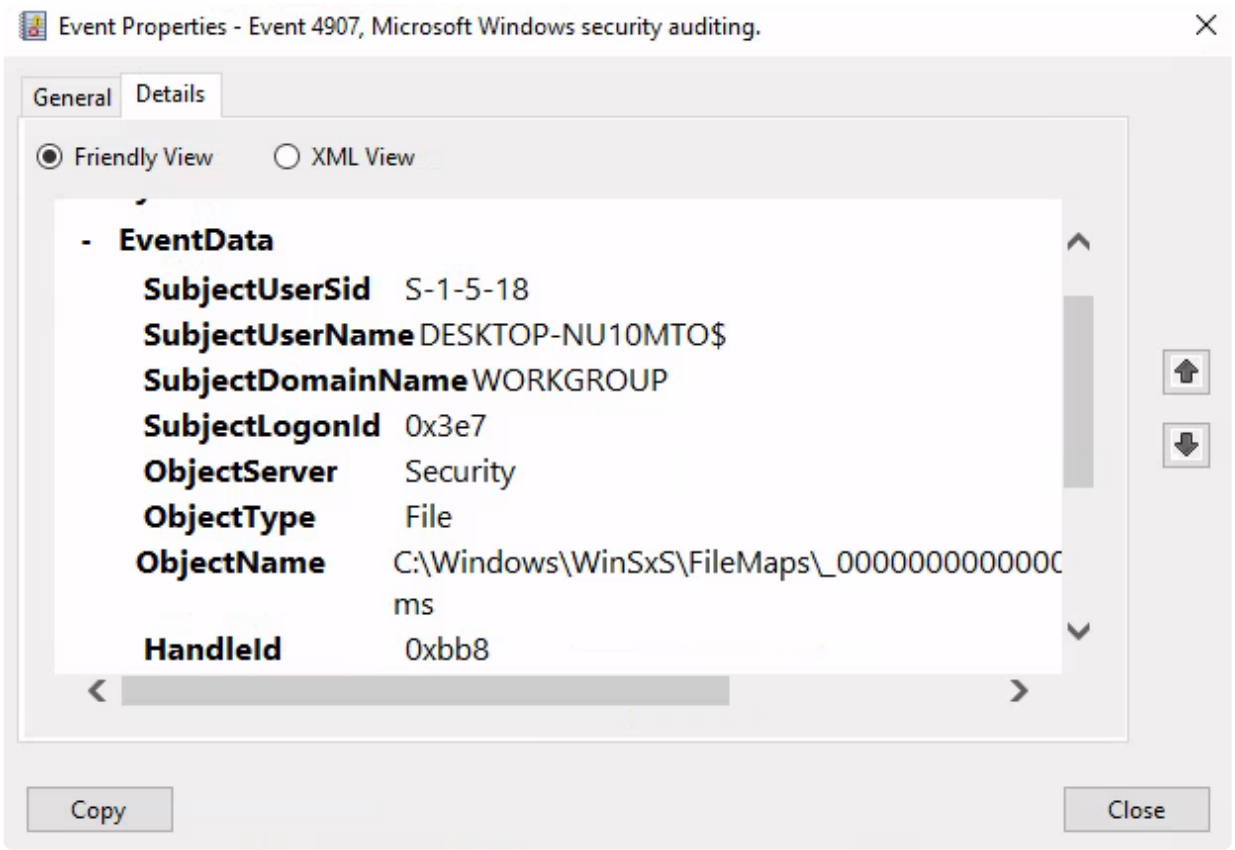
-> Note that we pick a suitable time interval, 1 hour within the logon time.

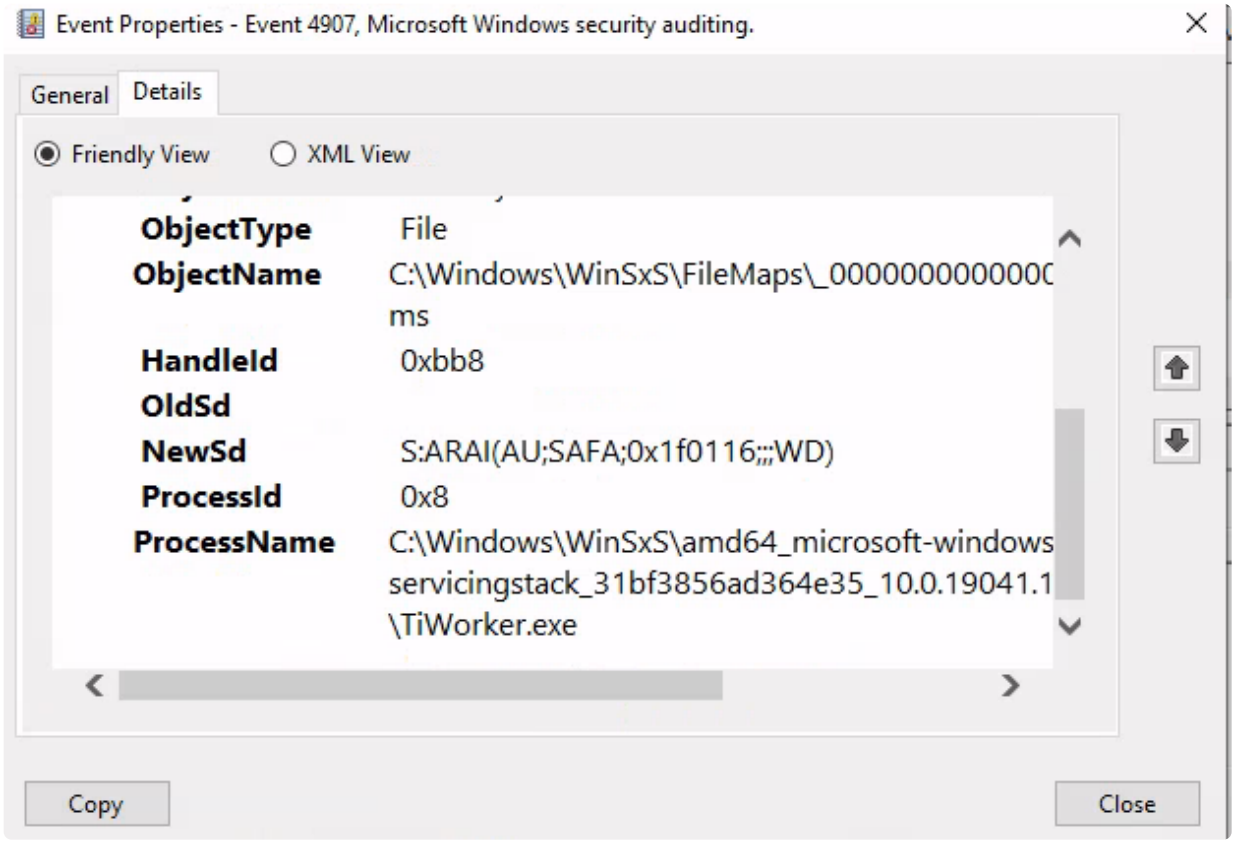
-> Also note that window system log has an `+7 hours` positive offset in 24 hours, so `10:25` am becomes `17:25` in the xml query.

```
 <Select Path="Security">*[System[TimeCreated[@SystemTime&gt;='2022-08-03T17:23:25.000Z' and @SystemTime&lt;='2022-08-03T18:23:25.999Z']]] and *[EventData[Data[@Name='SubjectLogonId']='0x3E7']]</Select>
```



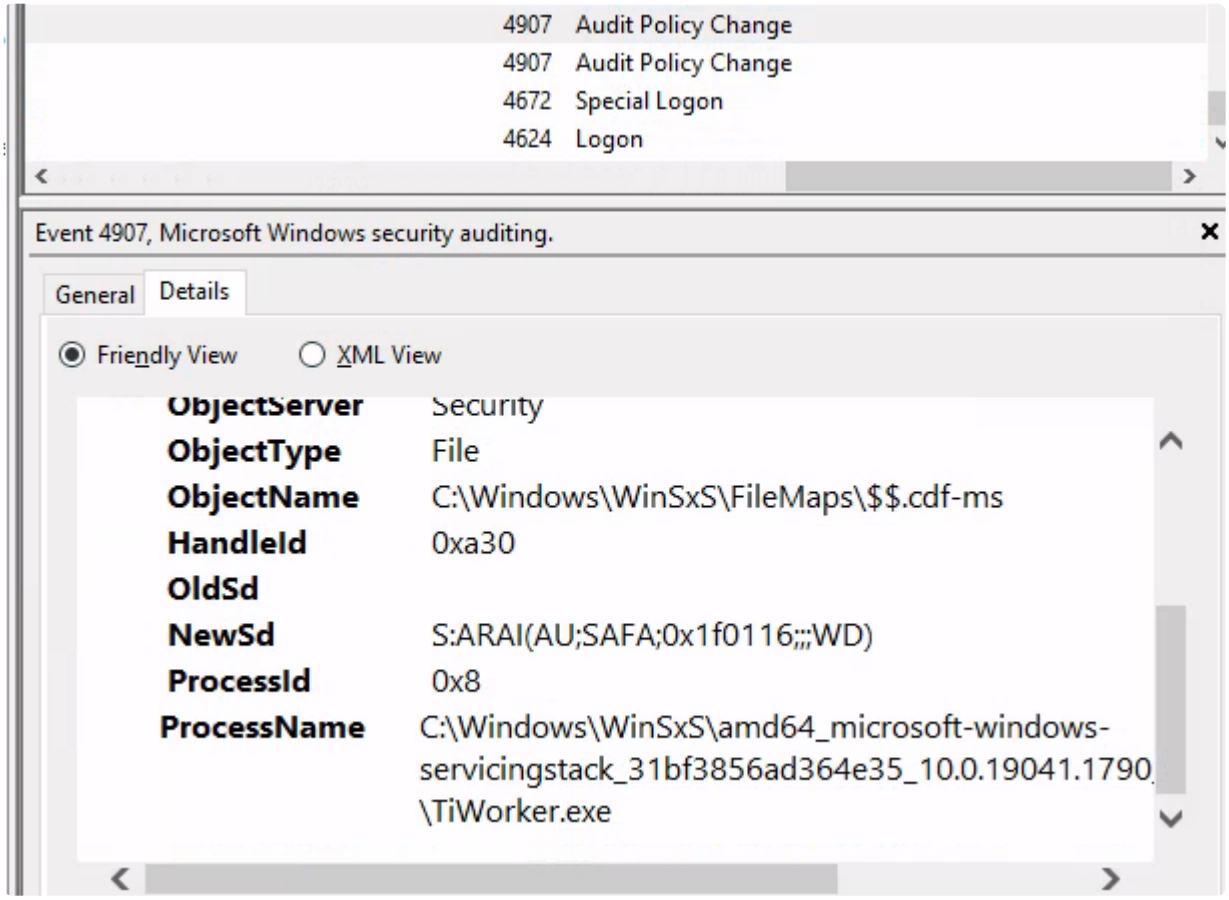
- Next, we identified an event with ID 4907, which indicates a change in SACL of an object, as shown below:





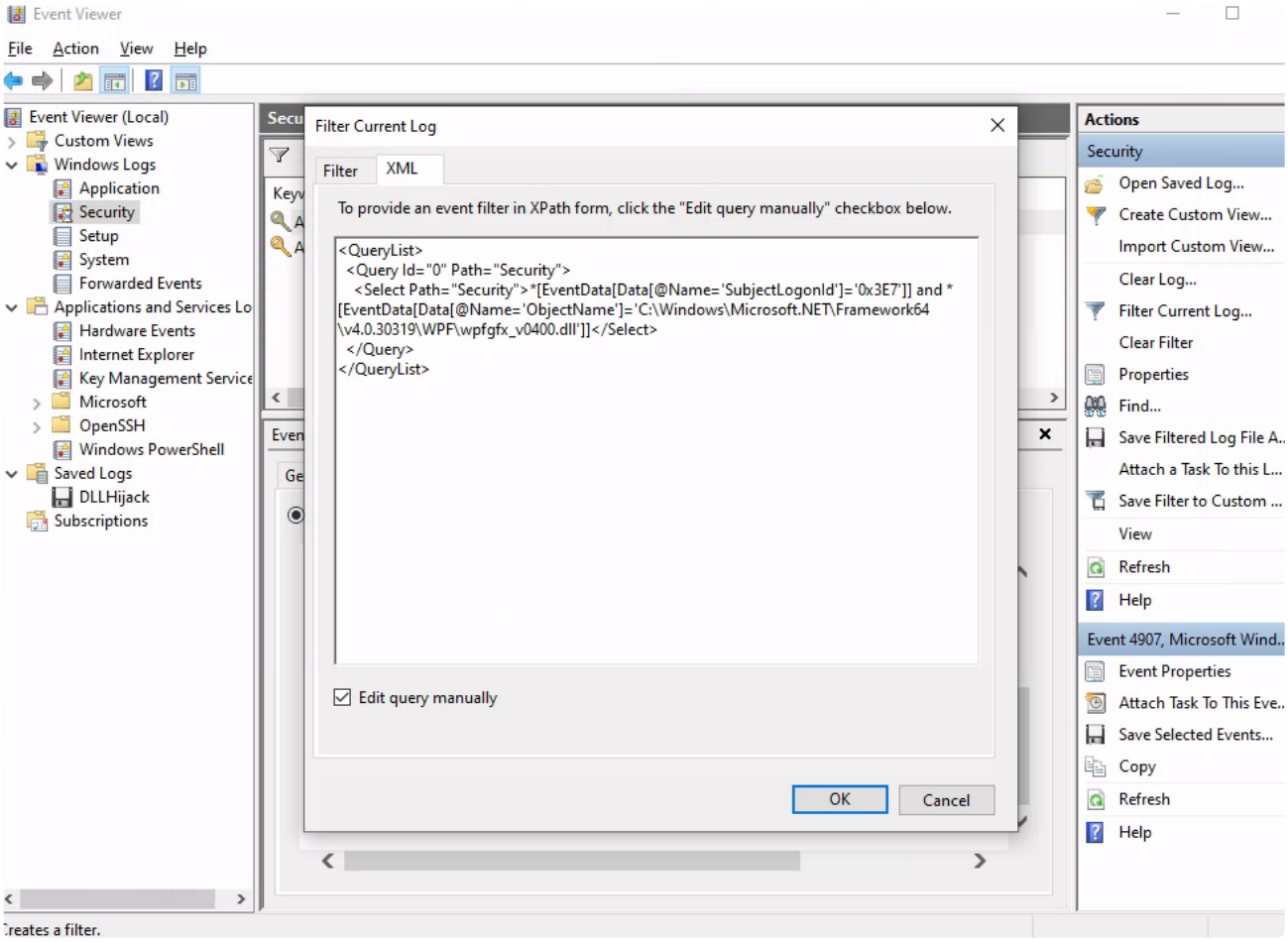
-> We see that the Process responsible for the change is `TiWorker.exe`

-> Looking at the next event 4907 which happened at the same time, we can confirm that the change is caused by `TiWorker.exe`

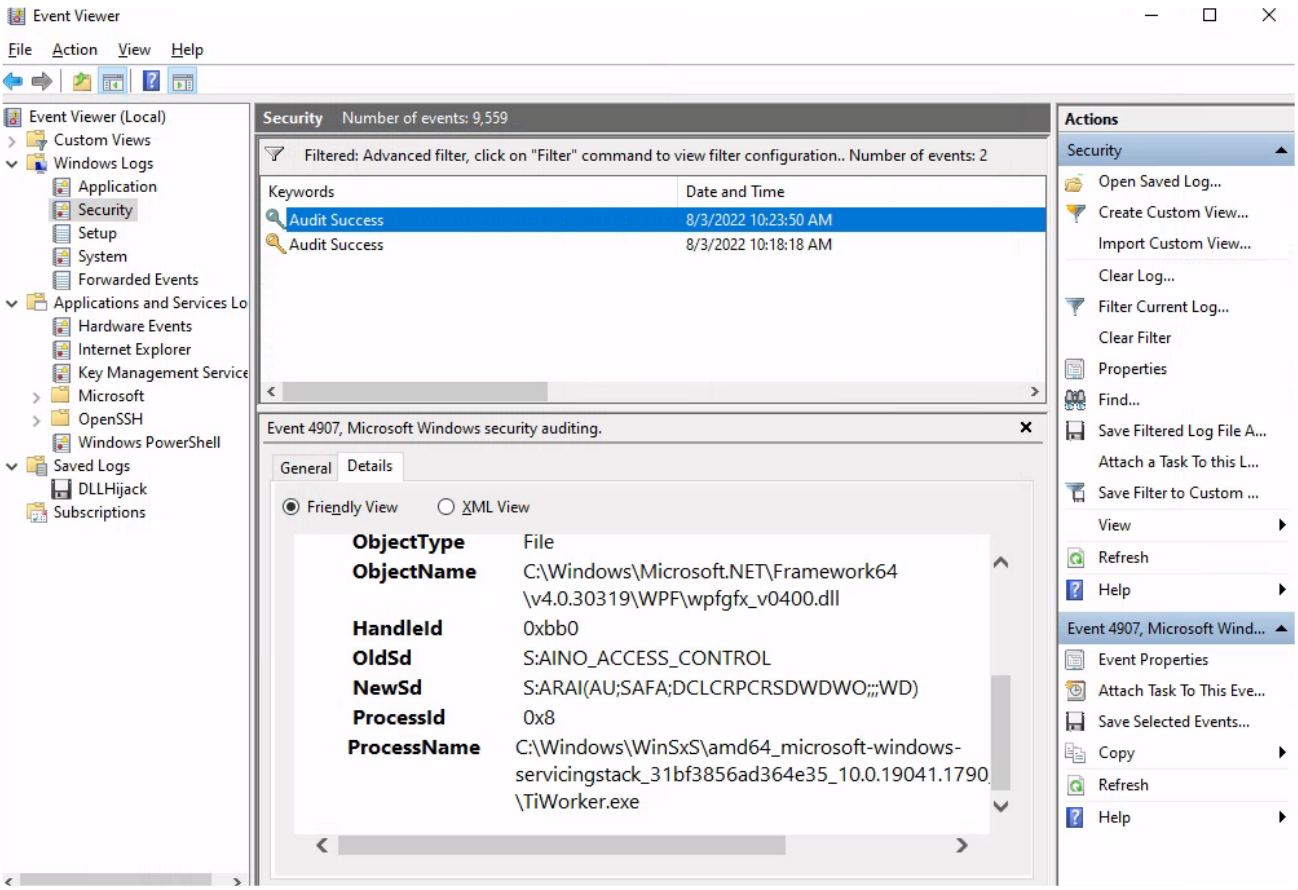


- Build an XML query to determine if the previously mentioned executable modified the auditing settings of C:\Windows\Microsoft.NET\Framework64\v4.0.30319\WPF\wpfgfx_v0400.dll. Enter the time of the identified event in the format HH:MM:SS as your answer.
  -> We keep the XML query as in the previous question, but instead change the object name to the required one and also removing the time restriction we imposed previously

```
 <Select Path="Security">*
[EventData[Data[@Name='SubjectLogonId']='0x3E7']] and *
[EventData[Data[@Name='ObjectName']='C:\Windows\Microsoft.NET\Framework6
4\v4.0.30319\WPF\wpfgfx_v0400.dll']]</Select>
```



-> We see that it generated 2 logs that modified audit settings, where we are interested in the first one, where audit policy setting is modified by `TiWorker.exe`



-> Hence, we obtained the audit setting is changed at 10:23:50 at 8/3/2022.

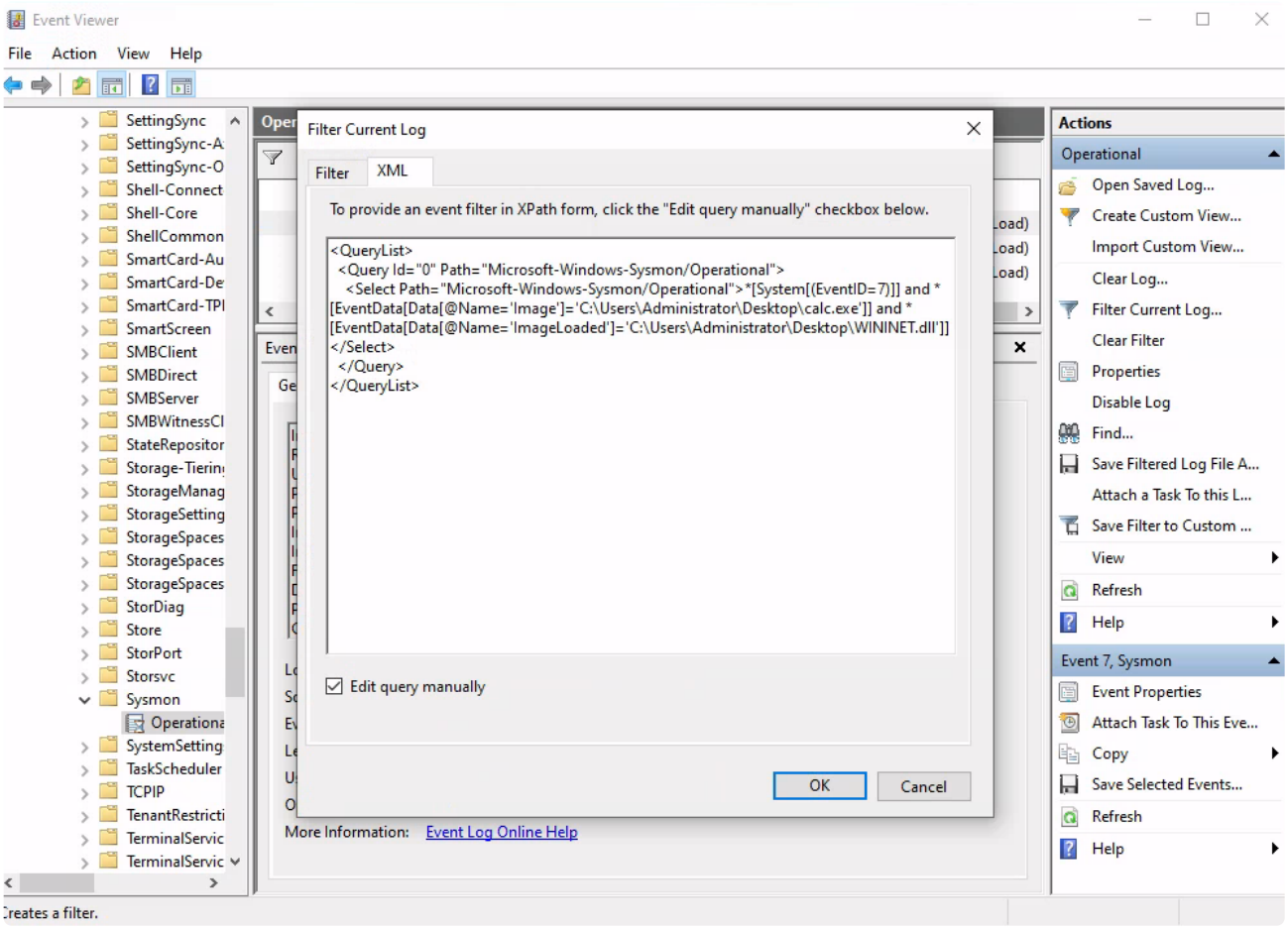
Analyzing Evil With Sysmon & Event Logs

Question

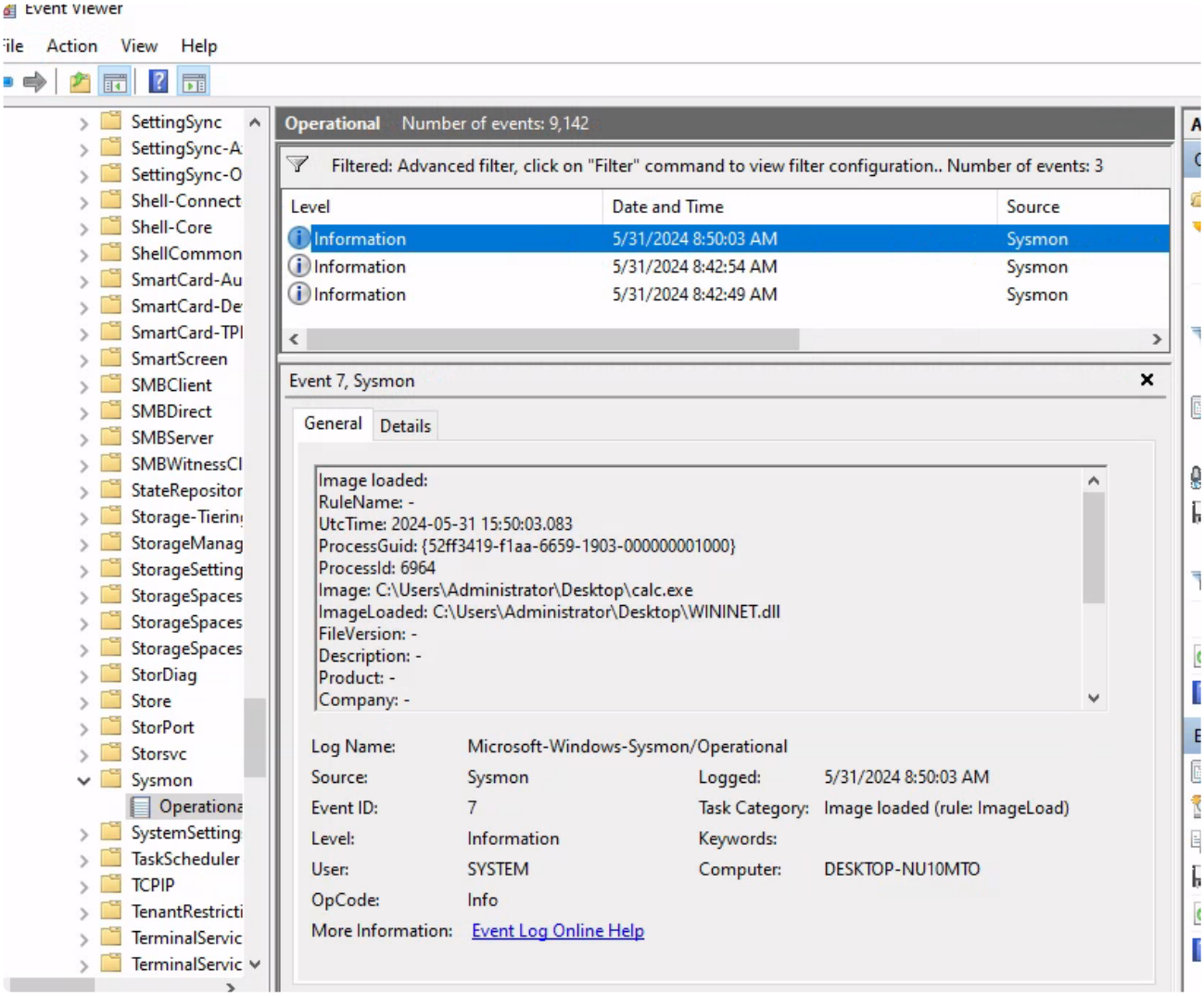
- Replicate the DLL hijacking attack described in this section and provide the SHA256 hash of the malicious WININET.dll as your answer. "C:\Tools\Sysmon" and "C:\Tools\Reflective DLLInjection" on the spawned target contain everything you need.

-> We practicality follow the section, with the following addition:
-> Look for the specific DLL and application loaded, using the following XML query.

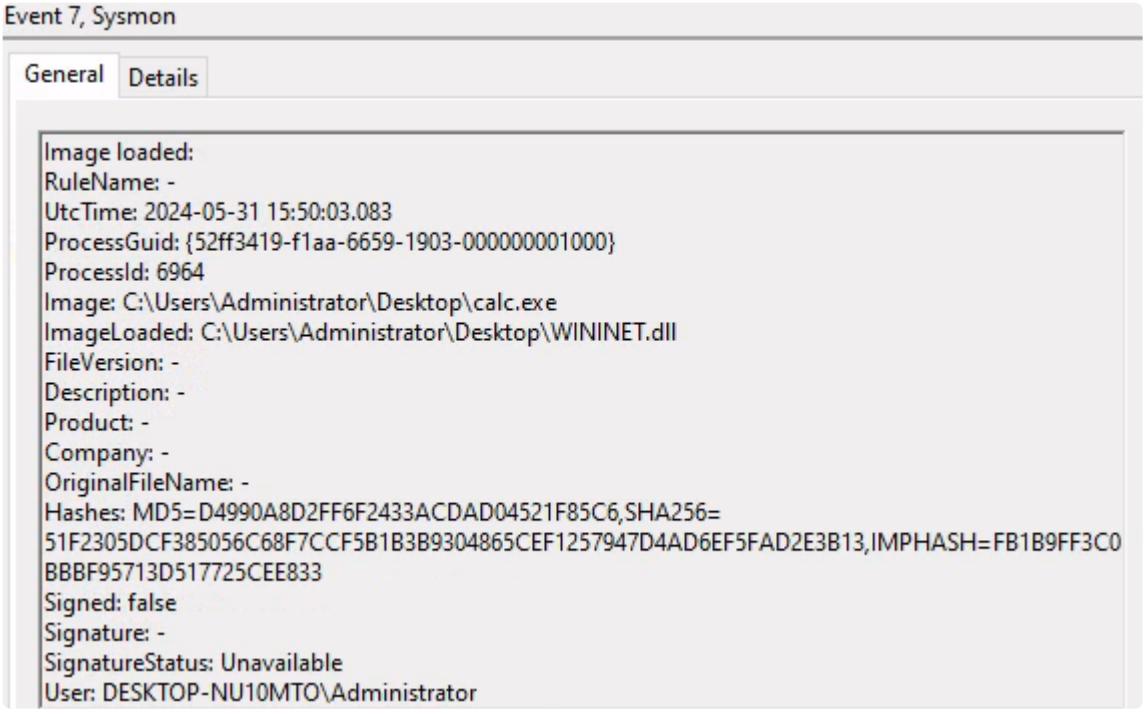
```
    <Select Path="Microsoft-Windows-Sysmon/Operational">*
[System[(EventID=7)]] and *
[EventData[Data[@Name='Image']='C:\Users\Administrator\Desktop\calc.exe'
]] and *
[EventData[Data[@Name='ImageLoaded']='C:\Users\Administrator\Desktop\WIN
INET.dll']]</Select>
```


Event Viewer
File Action View Help
SettingSync
SettingSync-A
SettingSync-O
Shell-Connect
Shell-Core
ShellCommon
SmartCard-Au
SmartCard-De
SmartCard-TPI
SmartScreen
SMBClient
SMBDirect
SMBServer
SMBWitnessCl
StateRepositor
Storage-Tierin
StorageManag
StorageSetting
StorageSpaces
StorageSpaces
StorageSpaces
StorDiag
Store
StorPort
Storsvc
Sysmon
Operationa
SystemSetting
TaskScheduler
TCPIP
TenantRestricti
TerminalServic
TerminalServic
Filter Current Log
Filter
XML
To provide an event filter in XPath form, click the "Edit query manually" checkbox below.
<QueryList>
<Query Id="0" Path="Microsoft-Windows-Sysmon/Operational">
<Select Path="Microsoft-Windows-Sysmon/Operational">*[System[(EventID=7)]] and *
[EventData[Data[@Name='Image']='C:\Users\Administrator\Desktop\calc.exe']] and *
[EventData[Data[@Name='ImageLoaded']='C:\Users\Administrator\Desktop\WININET.dll']]
</Select>
</Query>
</QueryList>
Edit query manually
OK
Cancel
More Information:
Event Log Online Help
Actions
Operational
Open Saved Log...
Create Custom View...
Import Custom View...
Clear Log...
Filter Current Log...
Clear Filter
Properties
Disable Log
Find...
Save Filtered Log File A...
Attach a Task To this L...
Save Filter to Custom ...
View
Refresh
Help
Event 7, Sysmon
Event Properties
Attach Task To This Eve...
Copy
Save Selected Events...
Refresh
Help
Creates a filter.

-> Obtained the following logs



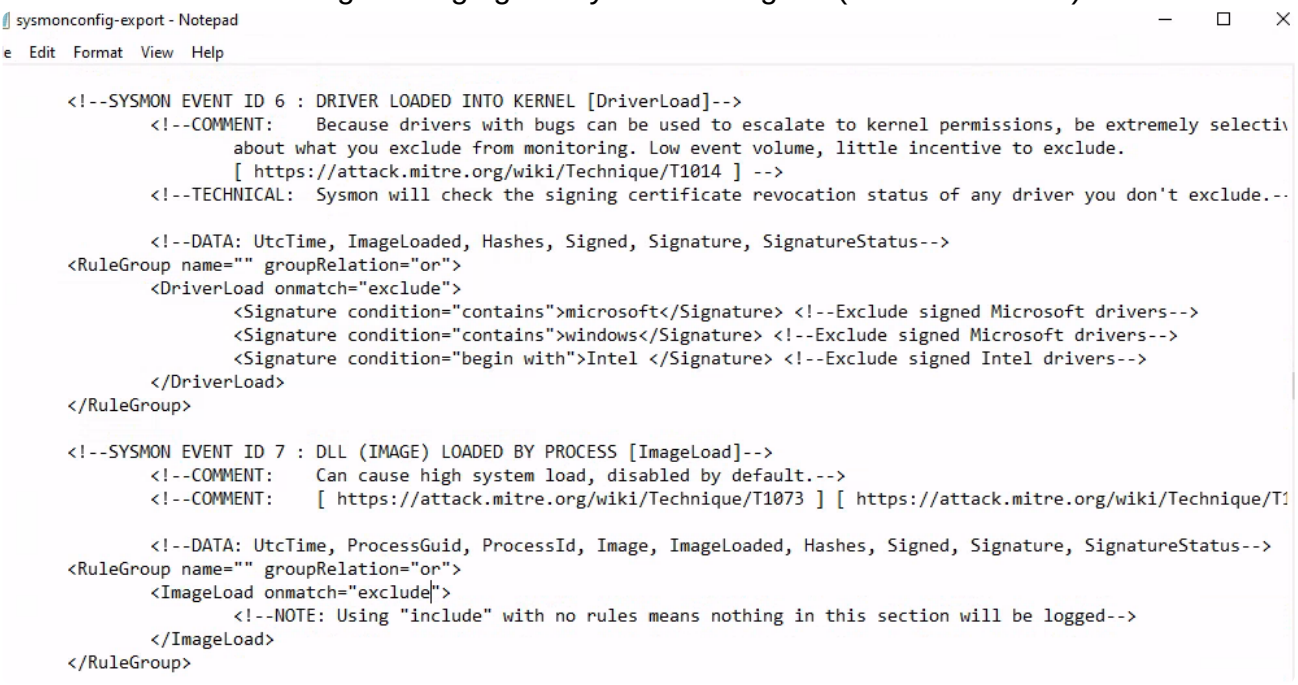
-> Examining one of the result, we see



-> And we obtain the various hashes, with the SHA256 hash as `51F2305DCF385056C68F7CCF5B1B3B9304865CEF1257947D4AD6EF5FAD2E3B13`

- Replicate the Unmanaged PowerShell attack described in this section and provide the SHA256 hash of clrjit.dll that spoolsv.exe will load as your answer. "C:\Tools\Sysmon" and "C:\Tools\PSInject" on the spawned target contain everything you need.

-> We need to ensure that event type is always configured.
-> We can do that through changing the sysmon config file (exclude event 7):

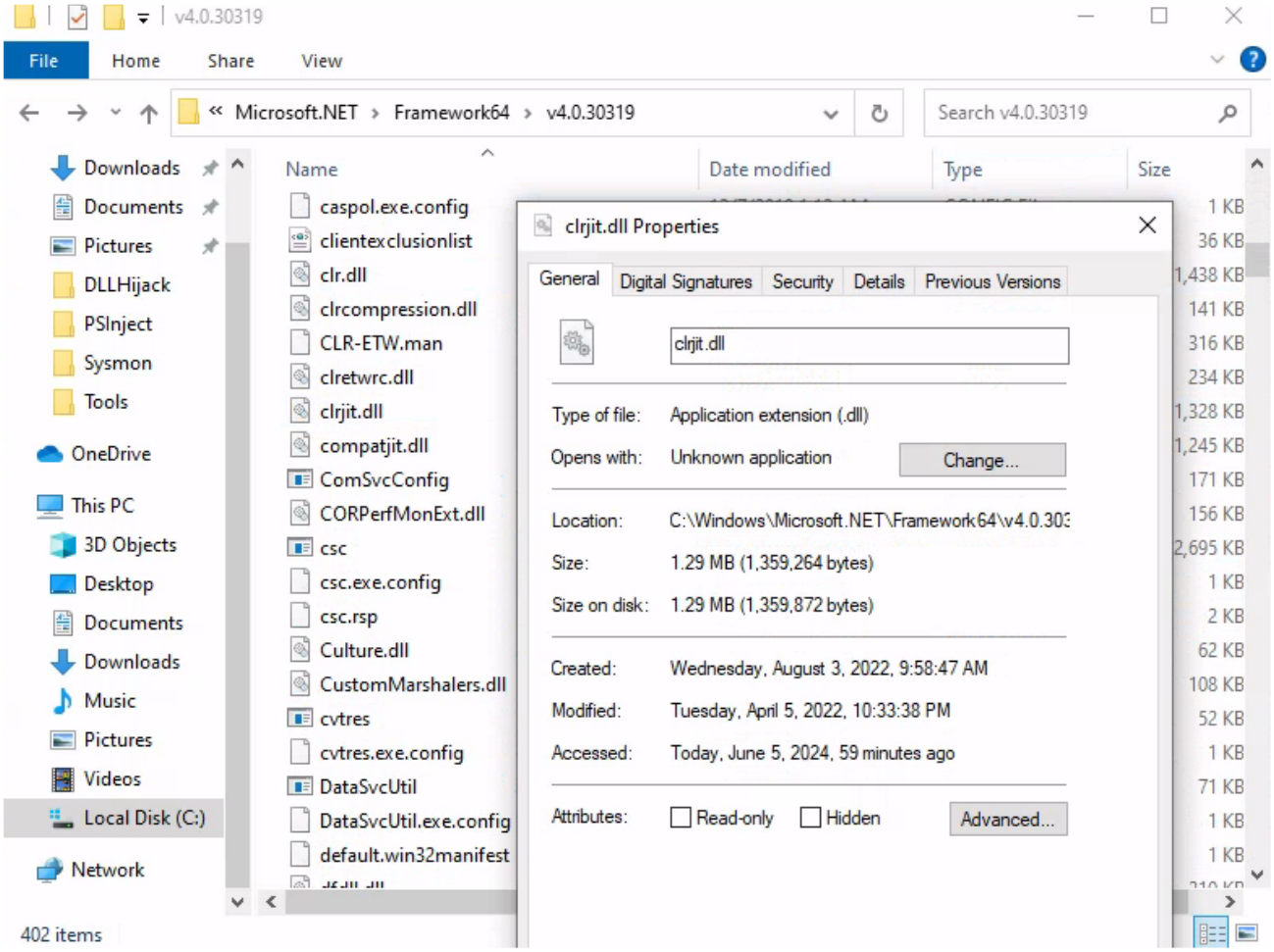


-> We update the sysmon configuration

```
sysmon.exe -c sysmonconfig-export.xml
```

-> After that, we follow the steps in the section.

-> Then we find the appropriate location of the `clrjit.dll` we want to search for

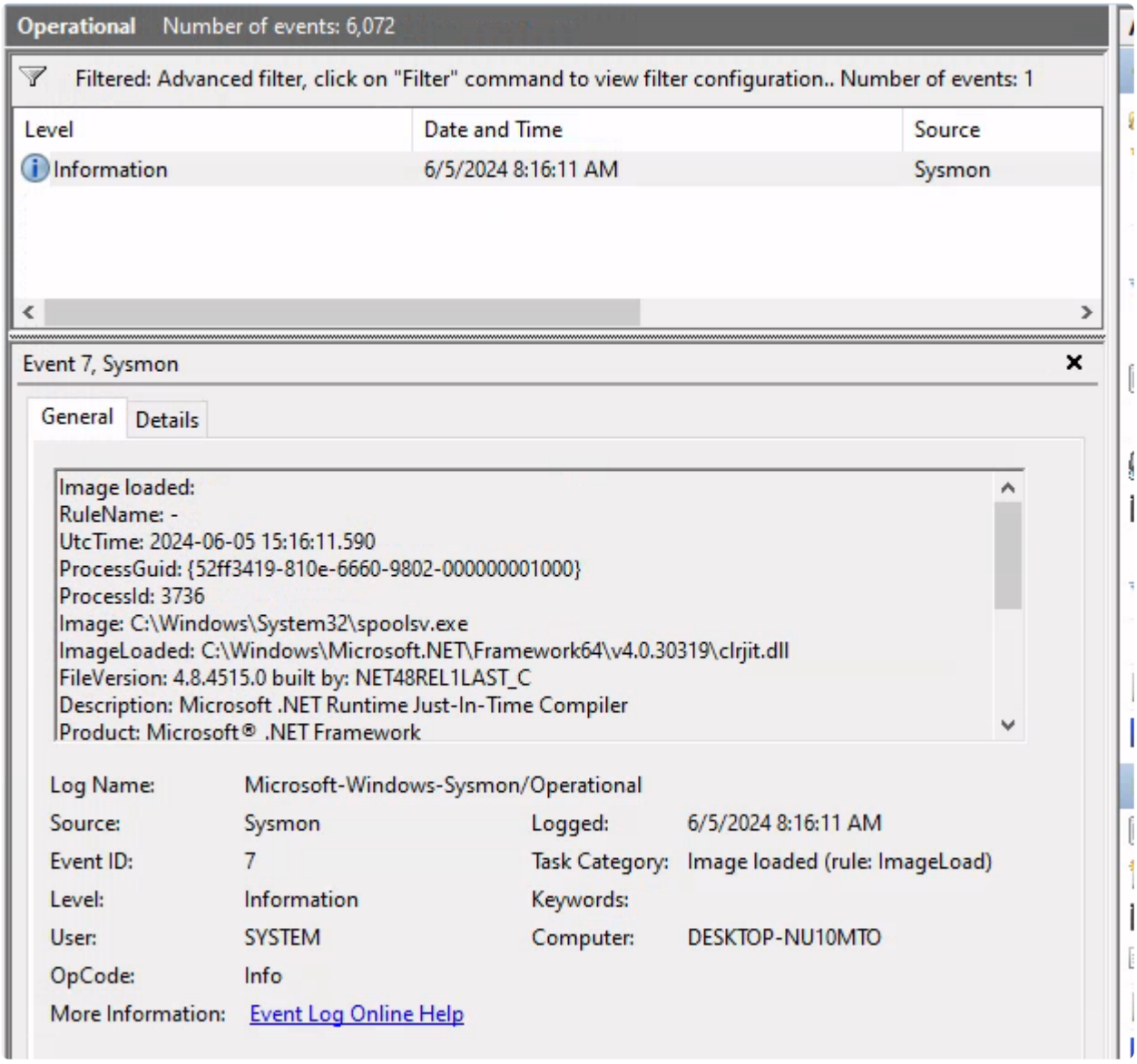


-> Which we get as `C:\Windows\Microsoft.NET\Framework64\v4.0.30319\clrjit.dll`

-> Then we go to the event log page, go to the sysmon log.

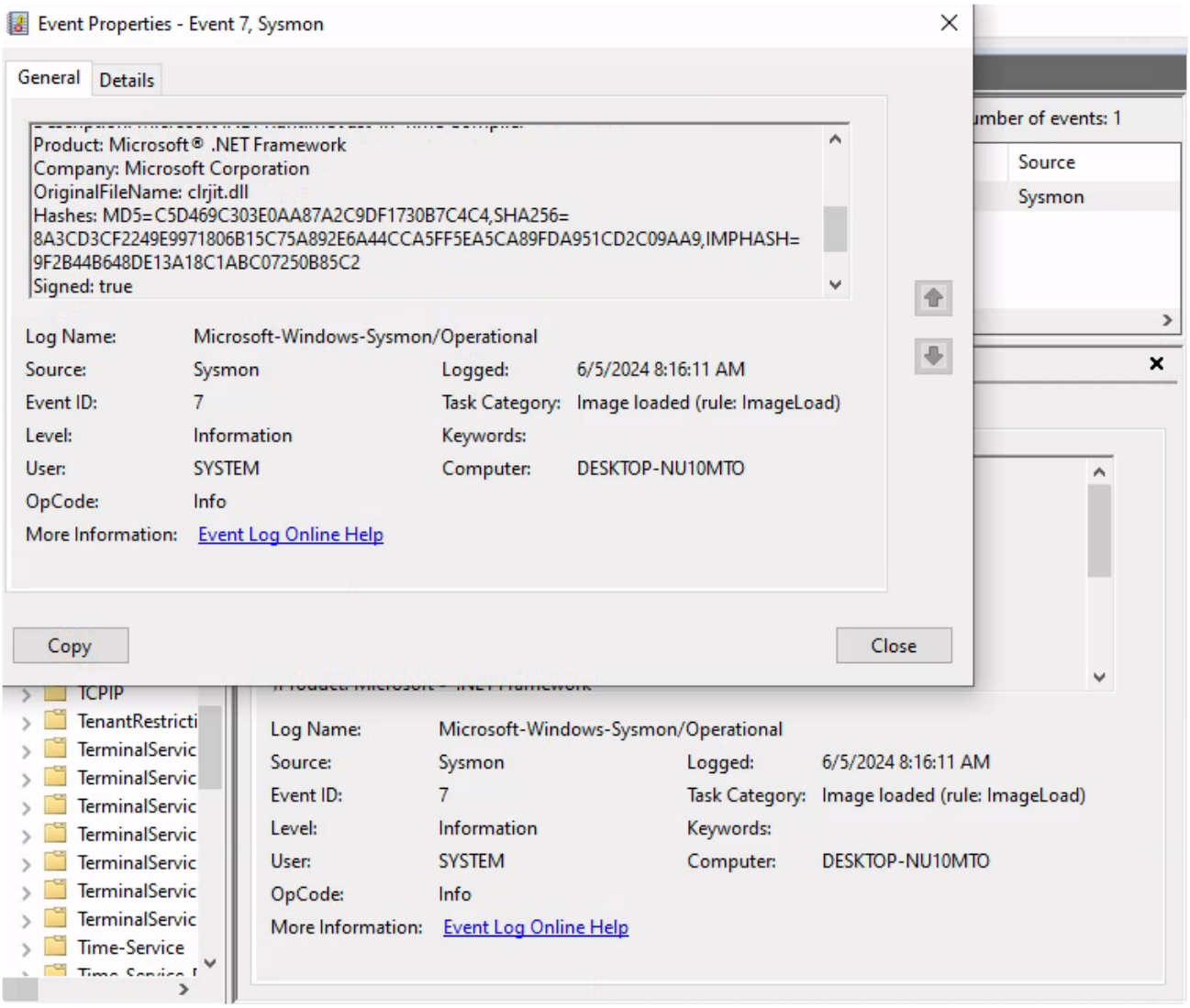
-> Then we search for the appropriate Sysmon log using information we gathered

```
    <Select Path="Microsoft-Windows-Sysmon/Operational">*
[System[(EventID=7)]] and *
[EventData[Data[@Name='Image']='C:\Windows\System32\spoolsv.exe']] and *
[EventData[Data[@Name='ImageLoaded']='C:\Windows\Microsoft.NET\Framework
64\v4.0.30319\clrjit.dll']]
    </Select>
```



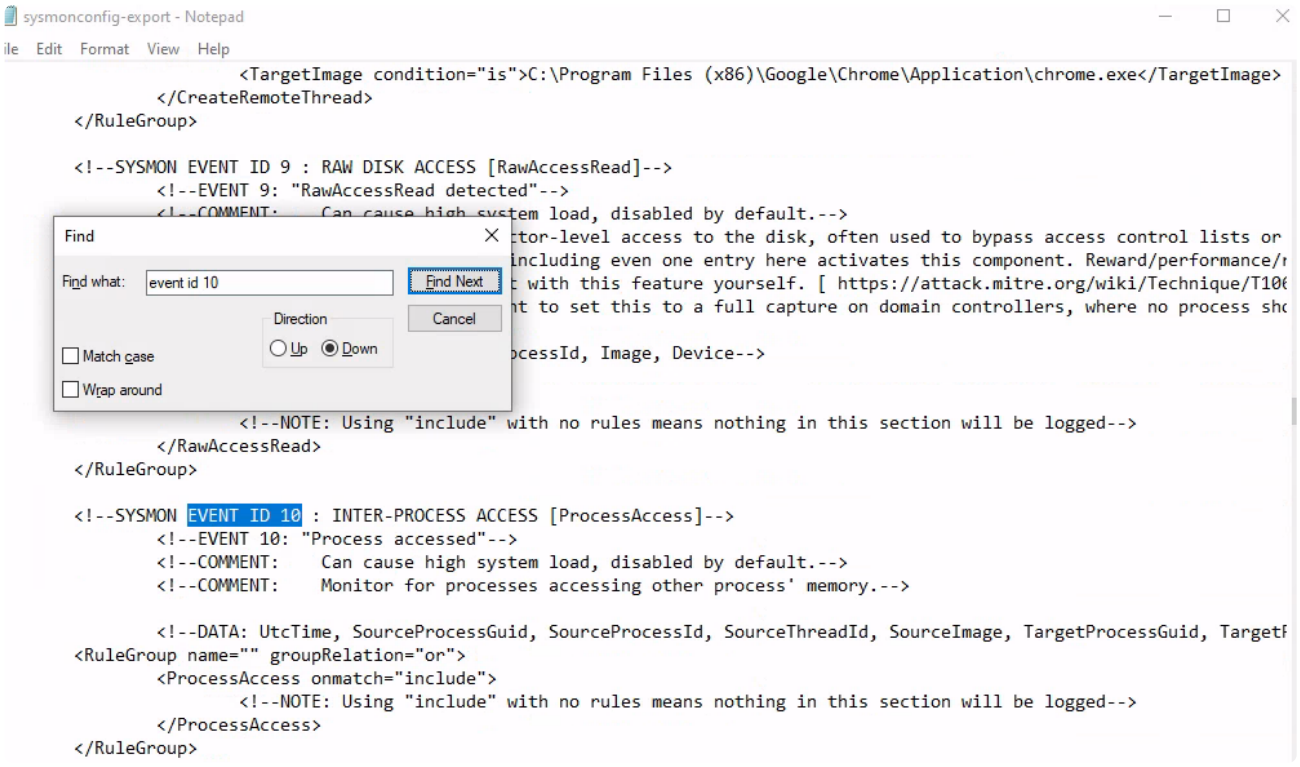
-> And we obtain one log we are interested in.

-> Looking in the log, we obtain the hash we want.

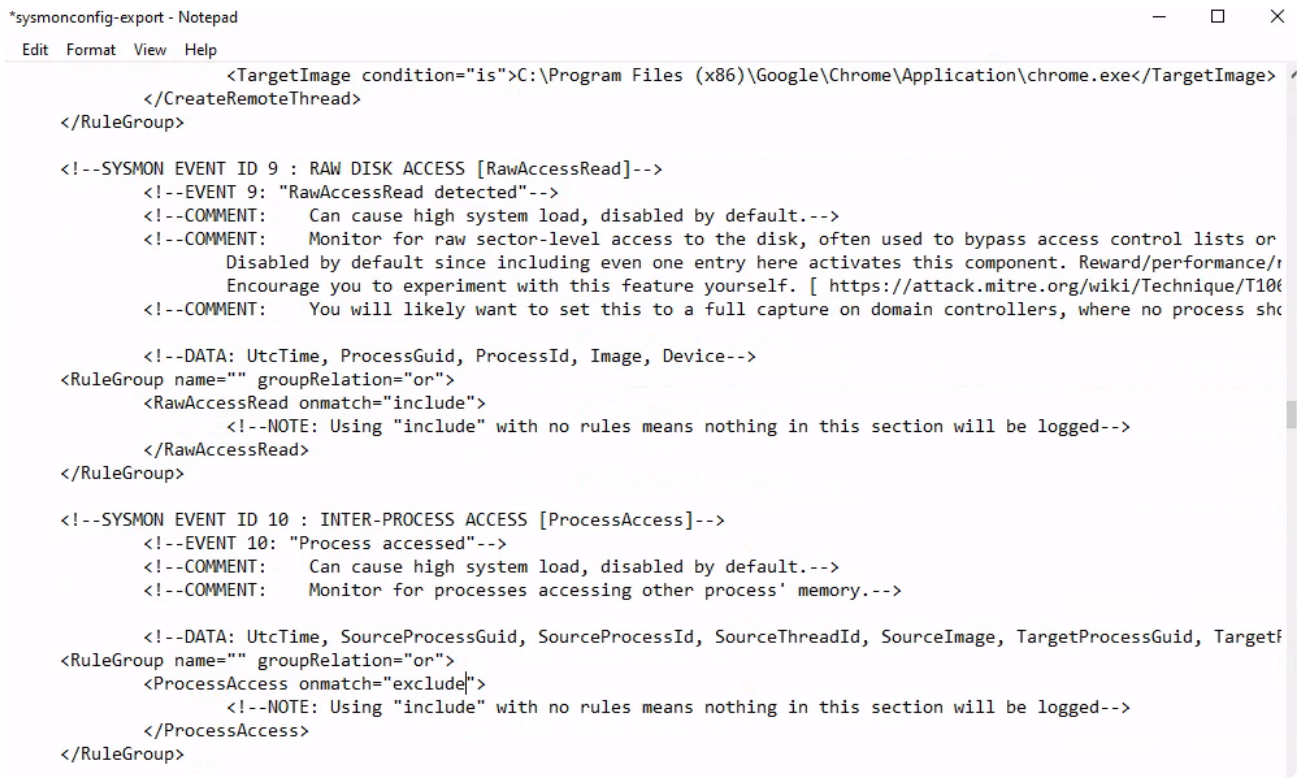


- Replicate the Credential Dumping attack described in this section and provide the NTLM hash of the Administrator user as your answer. "C:\Tools\Sysmon" and "C:\Tools\Mimikatz" on the spawned target contain everything you need.

-> We first checked the sysmon configuration file whether event id of 10 is enabled.



-> We see it is not, so we enabled it.



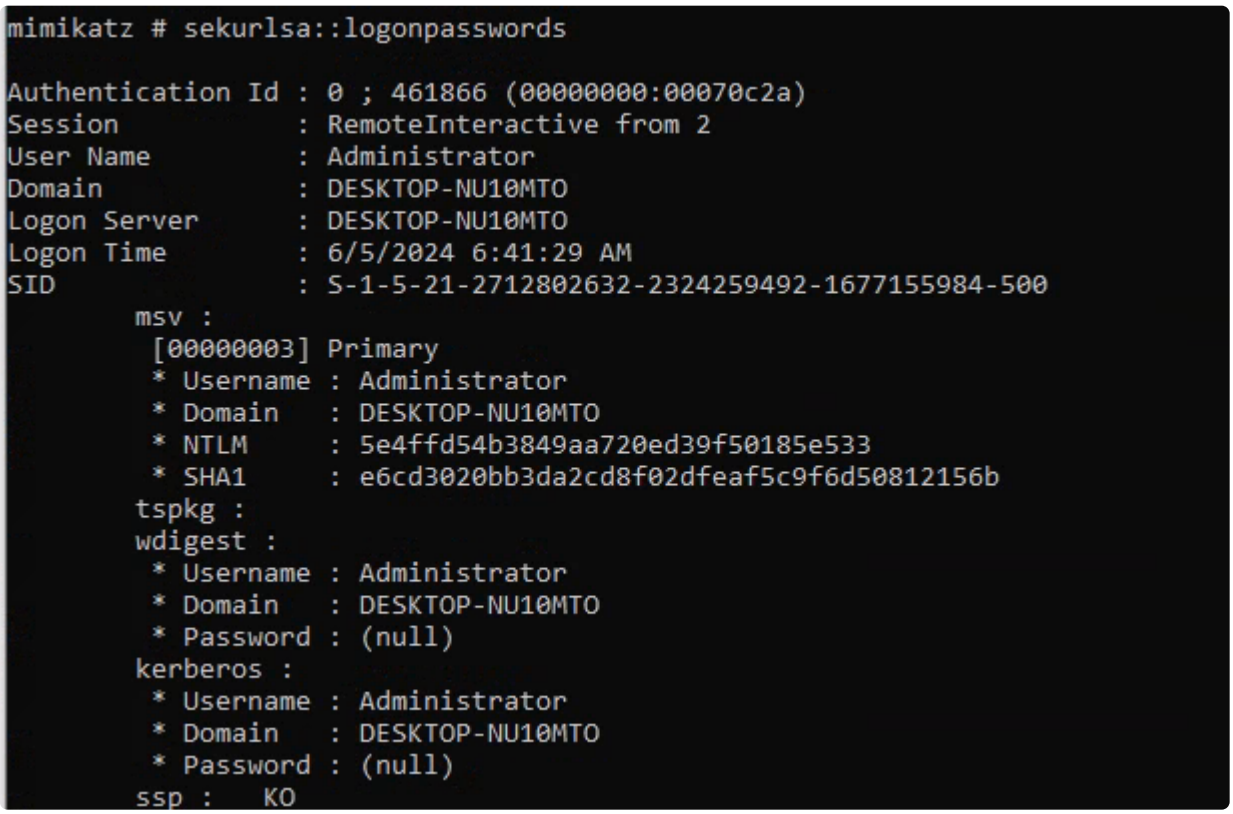
-> We change it to "'exclude" then we save and exit.

-> We update the Sysmon config file

```
sysmon.exe -c sysmonconfig-export.xml
```

-> Then, we execute mimkatz in the folder `C:\tools\mimkatz`

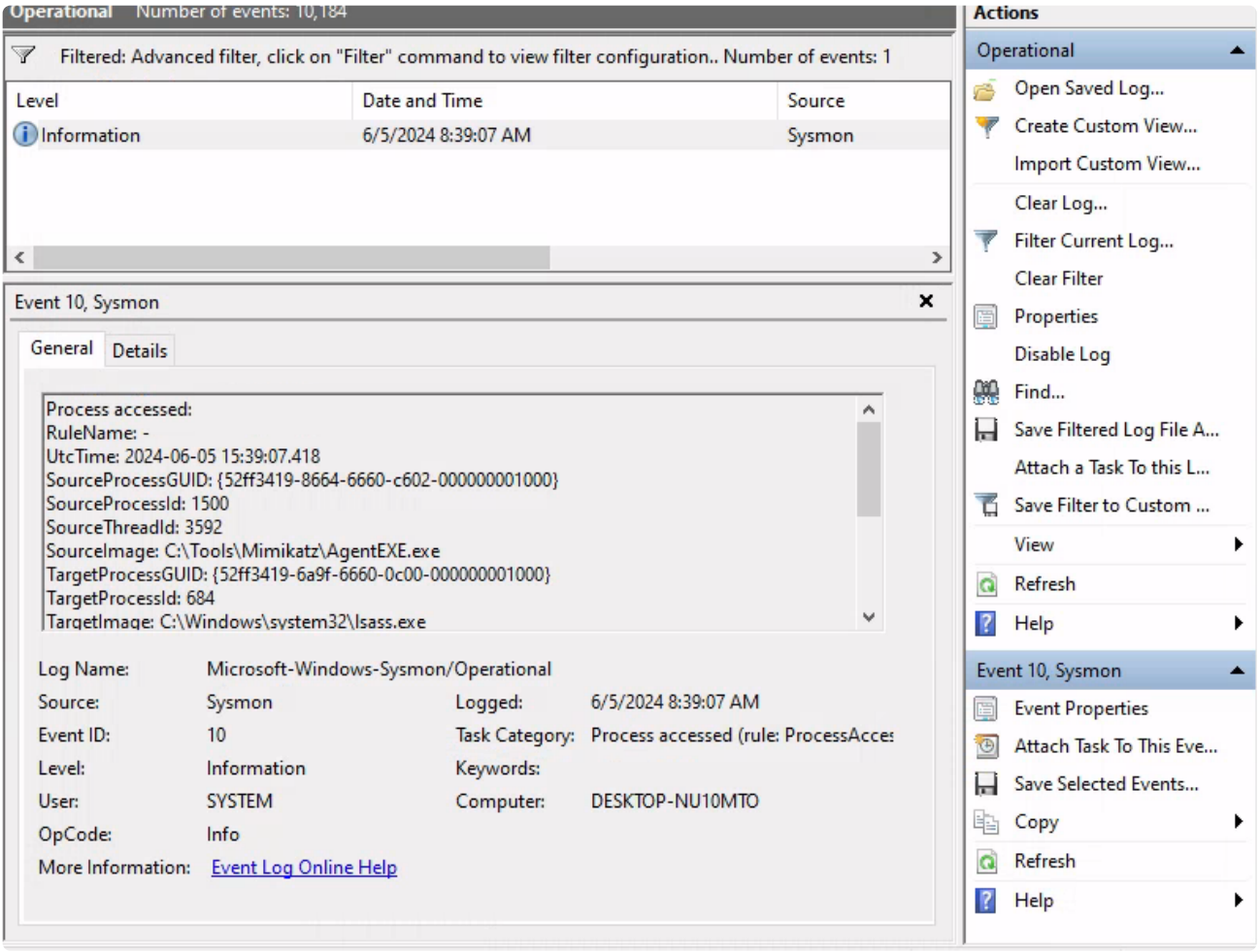
```
AgentEXE.exe
privilege::debug
sekurlsa::logonpasswords
```



-> And we get the hash we want.

-> As an bonus, we look into the sysmon log through using the following XML query

```
    <Select Path="Microsoft-Windows-Sysmon/Operational">*
[System[(EventID=10)]] and *
[EventData[Data[@Name='SourceImage']='C:\Tools\Mimikatz\AgentEXE.exe']]
and *
[EventData[Data[@Name='TargetImage']='C:\Windows\system32\lsass.exe']]
    </Select>
```



-> Where we obtained the following log, which looks very suspicious!

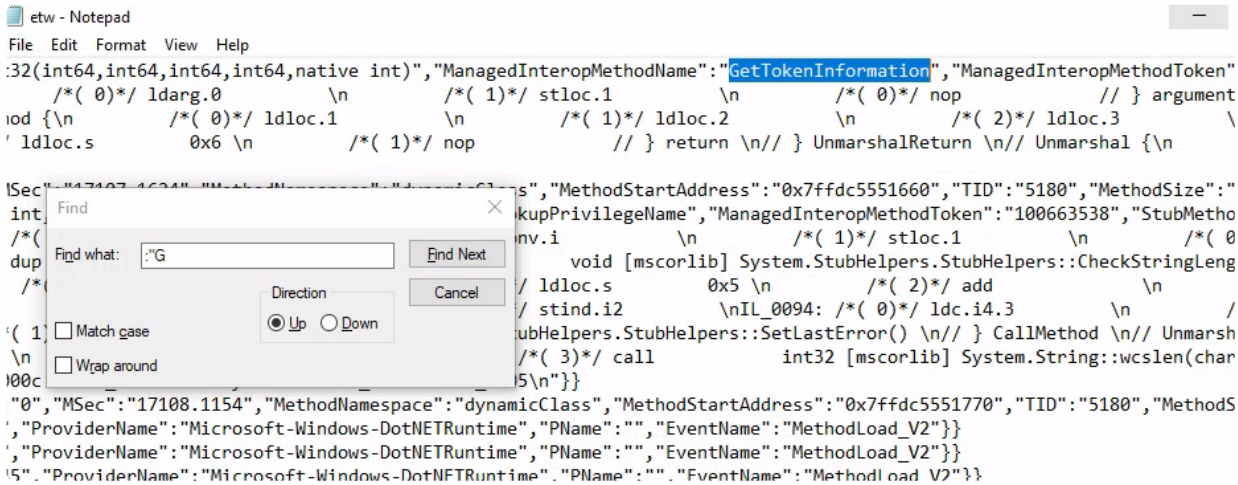
Additional Telemetry Sources

Tapping Into ETW

Question

- Replicate executing Seatbelt and SilkETW as described in this section and provide the ManagedInteropMethodName that starts with "G" and ends with "ion" as your answer. "c:\Tools\SilkETW_SilkService_v8\v8" and "C:\Tools\GhostPack Compiled Binaries" on the spawned target contain everything you need.

-> Follow exactly in the section, then searching accordingly with `:"G` and get it

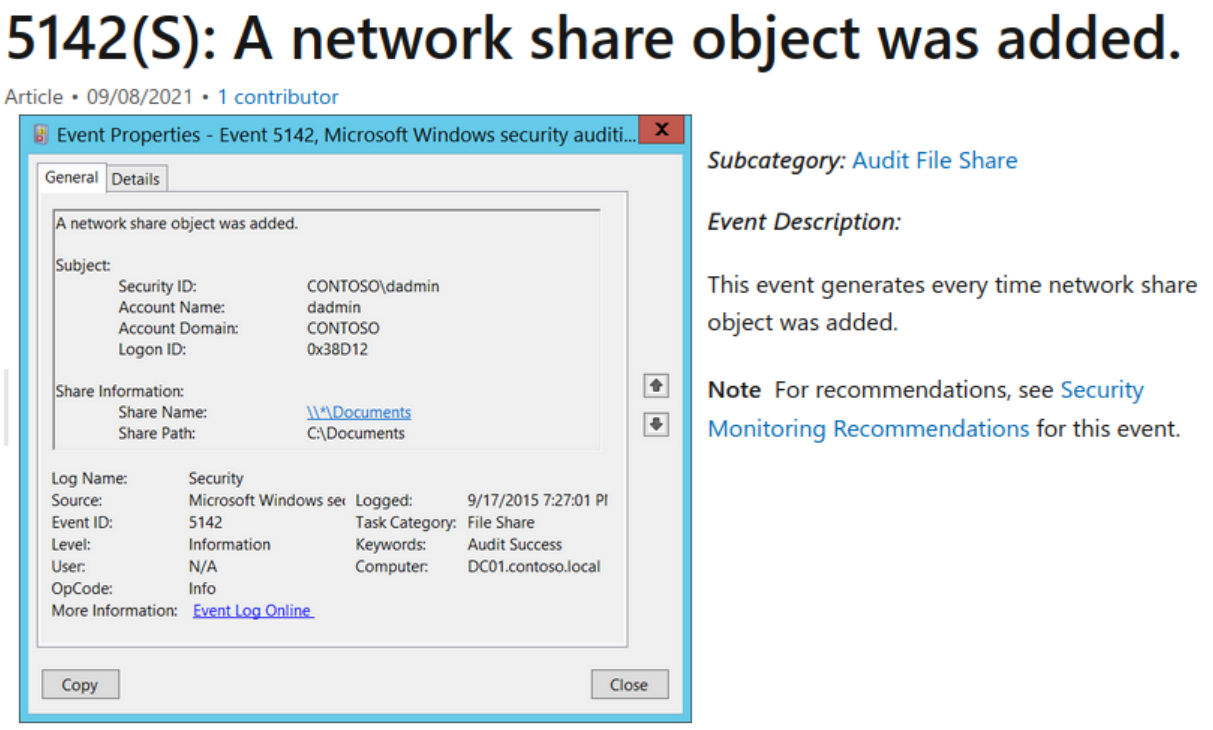


Analyzing Windows Event Logs En masse

Get-WinEvent

Question

- Utilize the Get-WinEvent cmdlet to traverse all event logs located within the "C:\Tools\chainsaw\EVTX-ATTACK-SAMPLES\Lateral Movement" directory and determine when the *\PRINT share was added. Enter the time of the identified event in the format HH:MM:SS as your answer.
- We look up the Event id for share creation

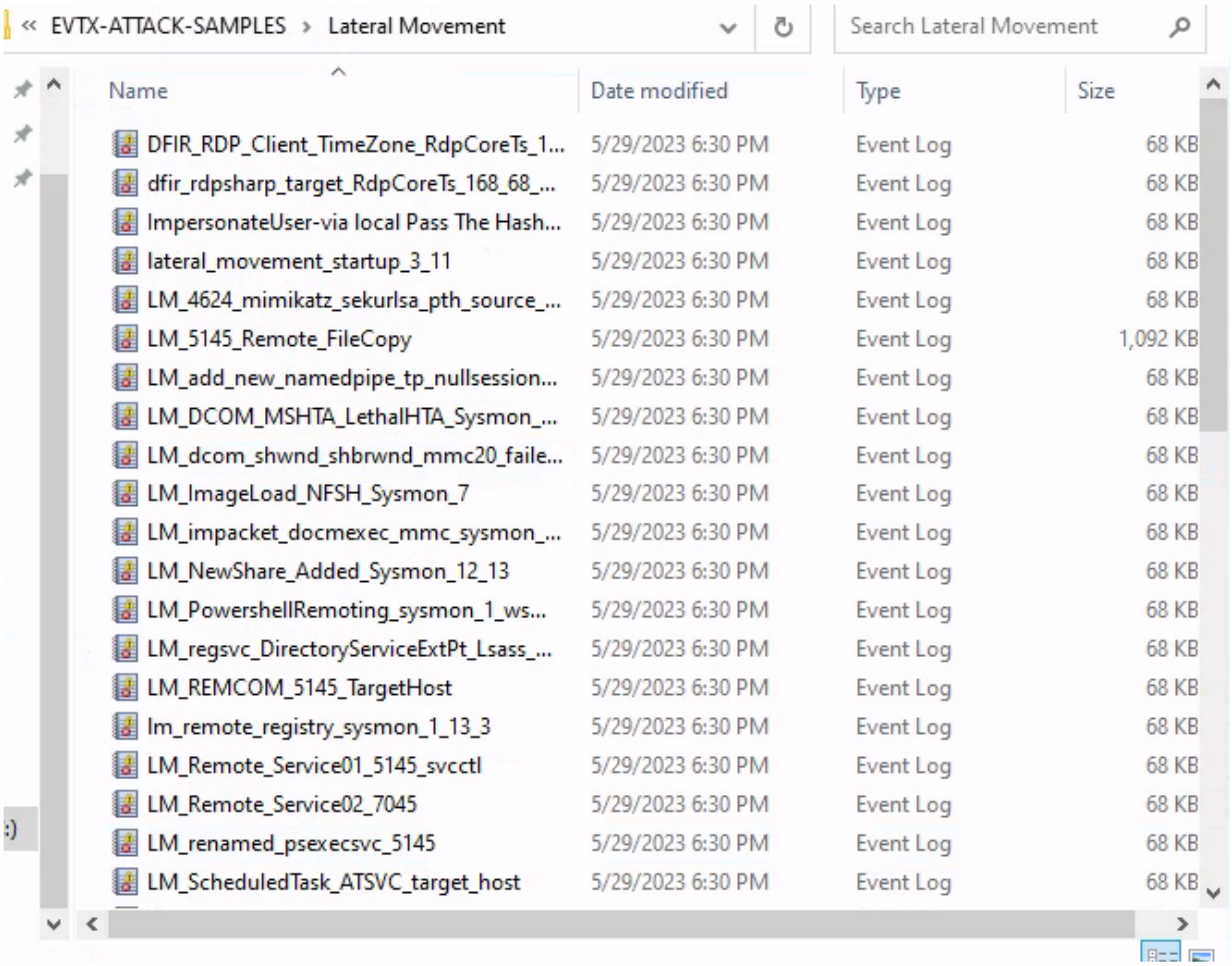


Event XML:

```
- <Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
- <System>
 <Provider Name="Microsoft-Windows-Security-Auditing" Guid="{54849625-5478-4994-A5BA-3E3B0328(
 <EventID>5142</EventID>
 <Version>0</Version>
 <Level>0</Level>
 <Task>12808</Task>
 <Opcode>0</Opcode>
 <Keywords>0x8020000000000000</Keywords>
 <TimeCreated SystemTime="2015-09-18T02:27:01.206646900Z" />
 <EventRecordID>268462</EventRecordID>
 <Correlation />
 <Execution ProcessID="4" ThreadID="4304" />
 <Channel>Security</Channel>
 <Computer>DC01.contoso.local</Computer>
 <Security />
 </System>
- <EventData>
 <Data Name="SubjectUserSid">S-1-5-21-3457937927-2839227994-823803824-1104</Data>
 <Data Name="SubjectUserName">dadmin</Data>
 <Data Name="SubjectDomainName">CONTOSO</Data>
 <Data Name="SubjectLogonId">0x38d12</Data>
 <Data Name="ShareName">\\\\\*\\Documents</Data>
 <Data Name="ShareLocalPath">C:\\Documents</Data>
 </EventData>
 </Event>
```

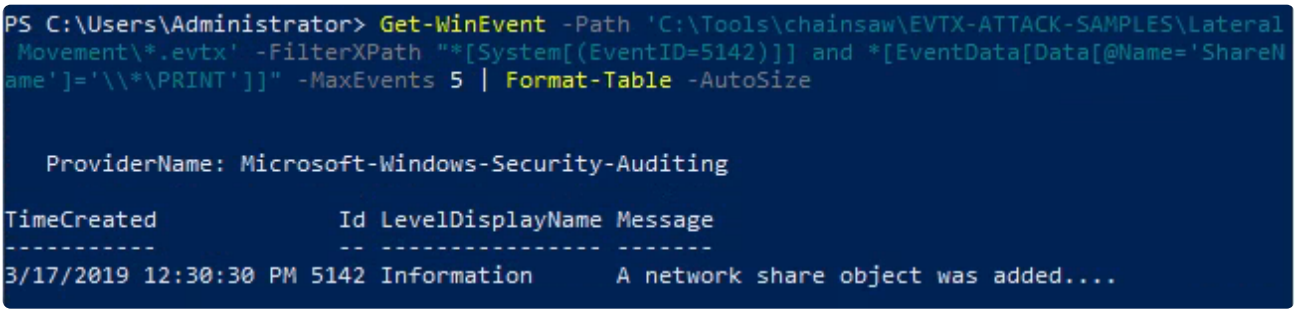
-> We see the interesting thing we need to filter are EventID and ShareName

- Observing the share, a-lot of log files



-> First find event id of adding share object, then apply xpath filter accordingly. (more flexibility over the property type than filter events based on property values and hashtable filtering doesn't work?)

```
Get-WinEvent -Path 'C:\Tools\chainsaw\EVTX-ATTACK-SAMPLES\Lateral Movement\*.evtx' -FilterXPath "*[System[(EventID=5142)]] and *[EventData[Data[@Name='ShareName']='\\*\PRINT']]" -MaxEvents 5 | Format-Table -AutoSize
```



-> Through which we get 12:30:30 as the answer required.