Labs - Understanding Log Sources & Investigating with Splunk

Skills assessment

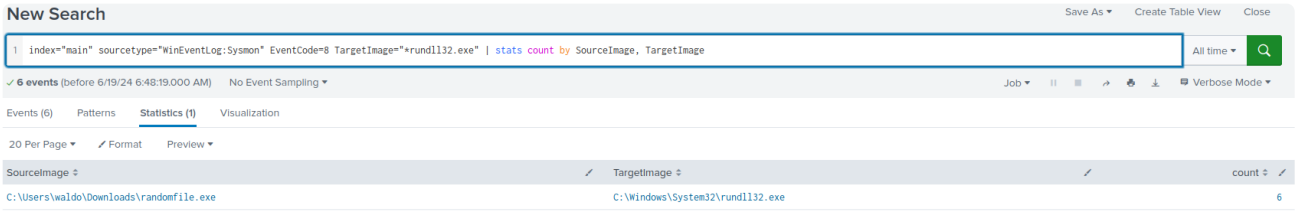
Scenario

This skills assessment section builds upon the progress made in the `Intrusion Detection With Splunk (Real-world Scenario)` section. Our objective is to identify any missing components of the attack chain and trace the malicious process responsible for initiating the infection.

Question

- Navigate to http://[Target IP]:8000, open the "Search & Reporting" application, and find through SPL searches against all data the process that created remote threads in rundll32.exe. Answer format: _.exe
  -> We craft the spl queries as follows (eventcode 8 for remote threads, target process as rundll32.exe)

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=8
TargetImage="*rundll32.exe" | stats count by SourceImage, TargetImage
```

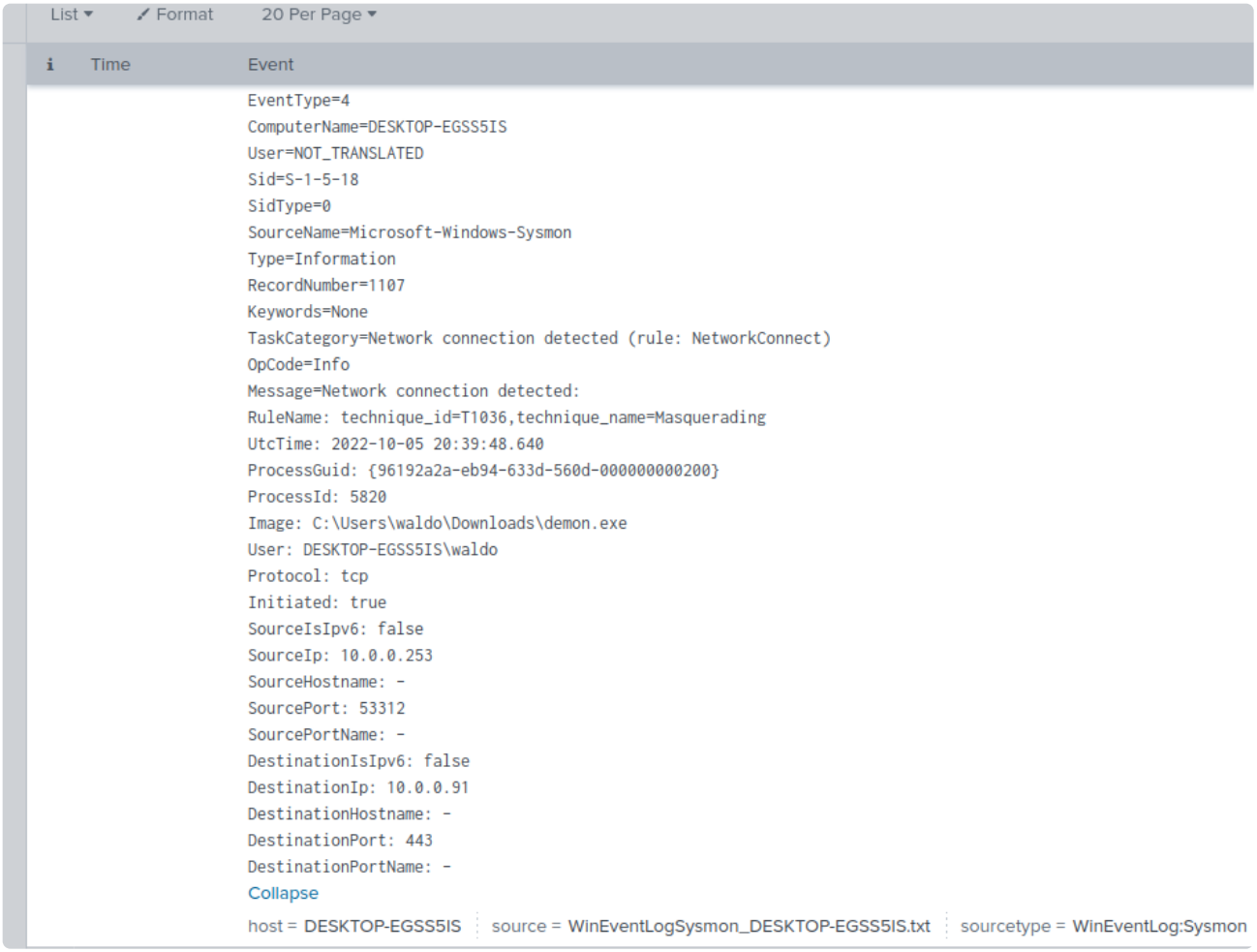


-> We see that it is randomfile.exe creating threads in rundll32.exe

- Navigate to http://[Target IP]:8000, open the "Search & Reporting" application, and find through SPL searches against all data the process that started the infection. Answer format: _.exe
  -> Here, starting the infection means that the very start of the actual exploitation process after the attacker landed a foothold.

-> We first look for the earliest events happening between the c2 server and the host for their initial interaction (which most likely uses some reverse shell)

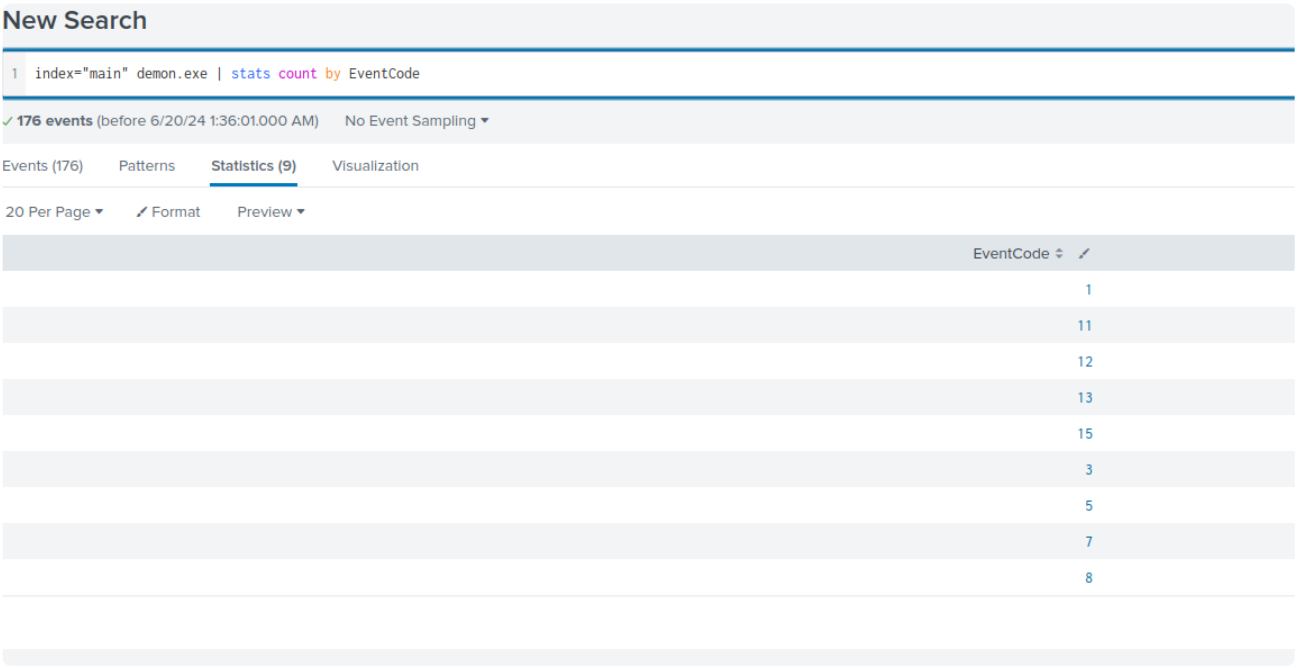
```
index="main" EventCode=3 (DestinationIp=10.0.0.186 OR
DestinationIp=10.0.0.91) | reverse
```



-> One of the earliest event occurred on 2022-10-05 20:39:48 and we can see it uses the name of demon.exe as an reverse shell.

- Now we look at the event code related to this malicious file to see what type of events the attacker is creating.

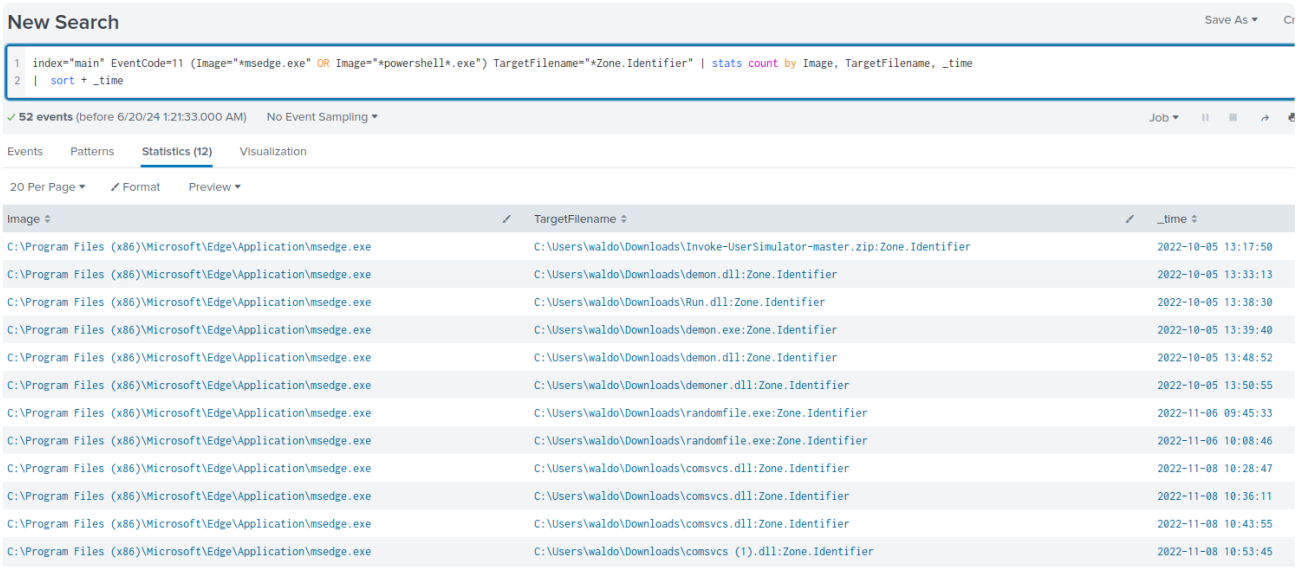
```
index="main" demon.exe | stats count by EventCode
```



-> We see that it has couple of event code related to this binary.
-> We can look into event code of 11 of file create and see what other files are downloaded.

- We look at the downloaded files on the internet.

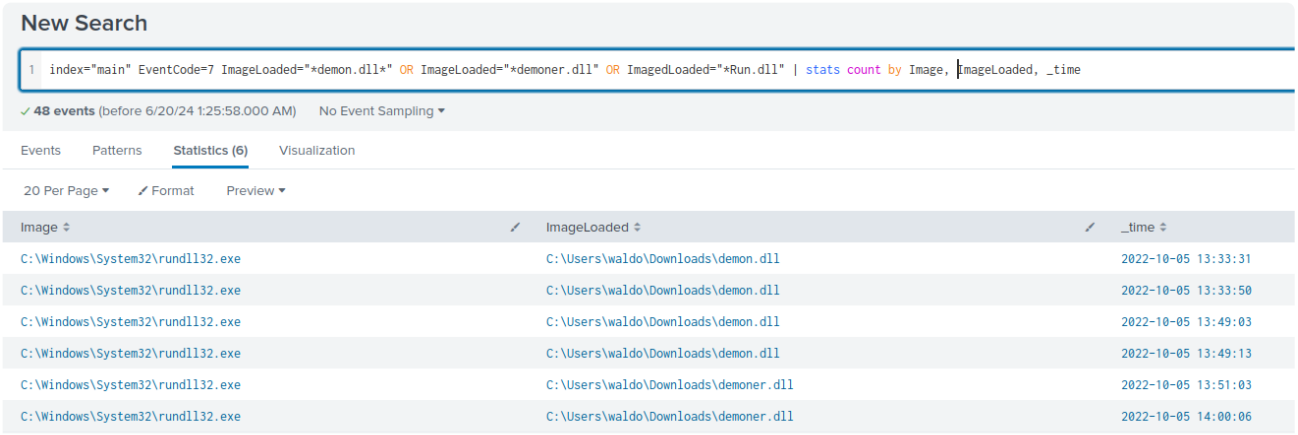
```
index="main" EventCode=11 (Image="*msedge.exe" OR
Image="*powershell*.exe") TargetFilename="*Zone.Identifier" | stats
count by Image, TargetFilename, _time
|  sort + _time
```



-> Also, the time for demon.exe is downloaded about 13:39:40 at 2022-10-05
-> We see related dll's of run.dll, demon.dll and demoner.dll.

- Let's see what files loads any of these dll downloaded (potentially malicious) and detect them:

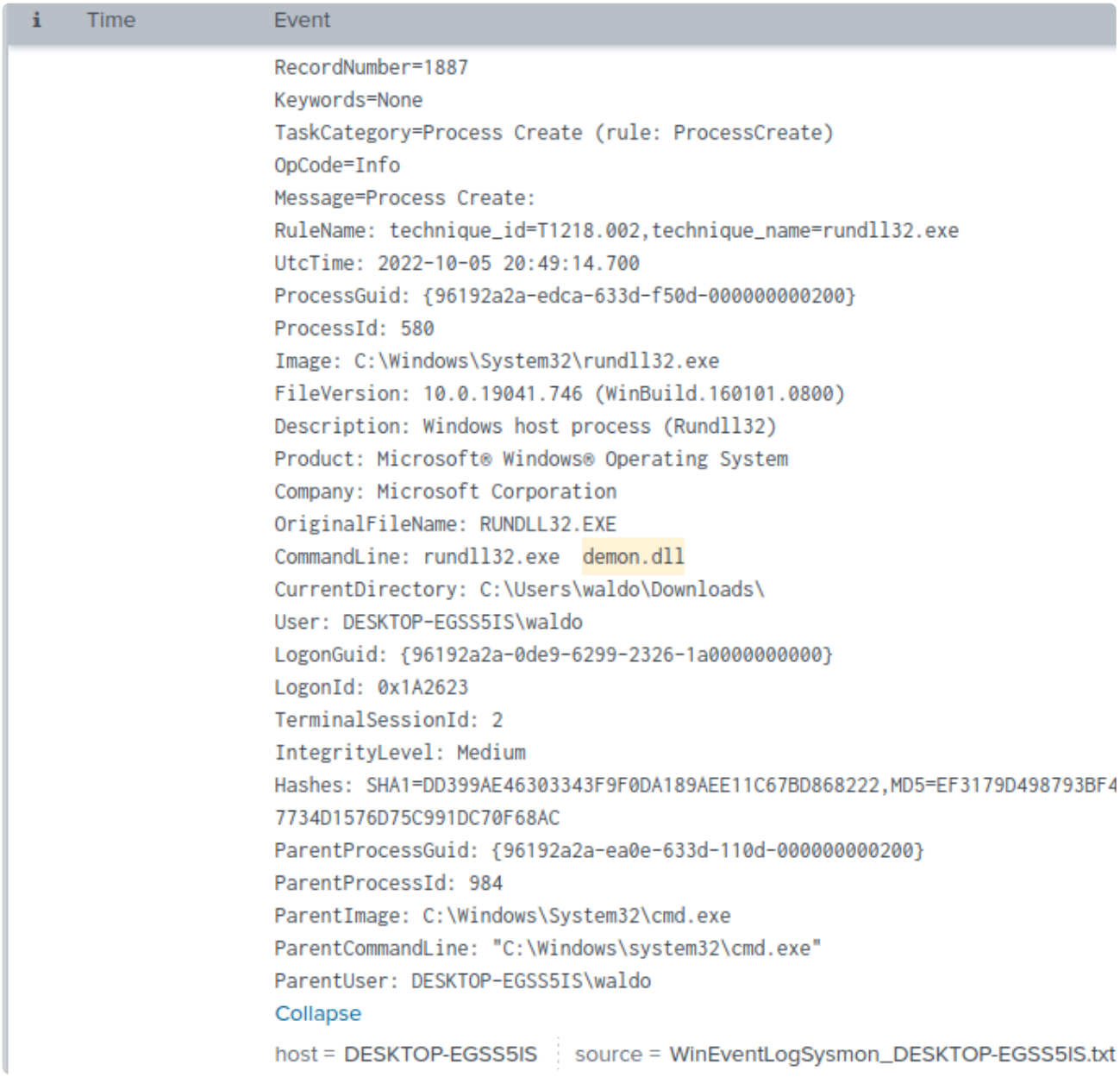
```
index="main" EventCode=7 ImageLoaded="*demon.dll*" OR
ImageLoaded="*demoner.dll" OR ImagedLoaded="*Run.dll" | stats count by
Image, ImageLoaded, _time
```



-> Hence, we see that it is rundll32.exe causing the start of the infection (start of exploitation attack chain after landing a foothold) through loading this malicious dll's.

- Furthermore, we could verify this by looking at the events associated with demon.dll.

```
index="main" *demon.dll*
```



-> And we see the rundll32.exe loaded demon.dll, veriyfing our finding.

Splunk Fundamentals

Introduction To Splunk & SPL

Question

- Navigate to http://[Target IP]:8000, open the "Search & Reporting" application, and find through an SPL search against all data the account name with the highest amount of Kerberos authentication ticket requests. Enter it as your answer.

-> The first step is to know that kerberos authentication meant that it is an TGS or TGT requests.

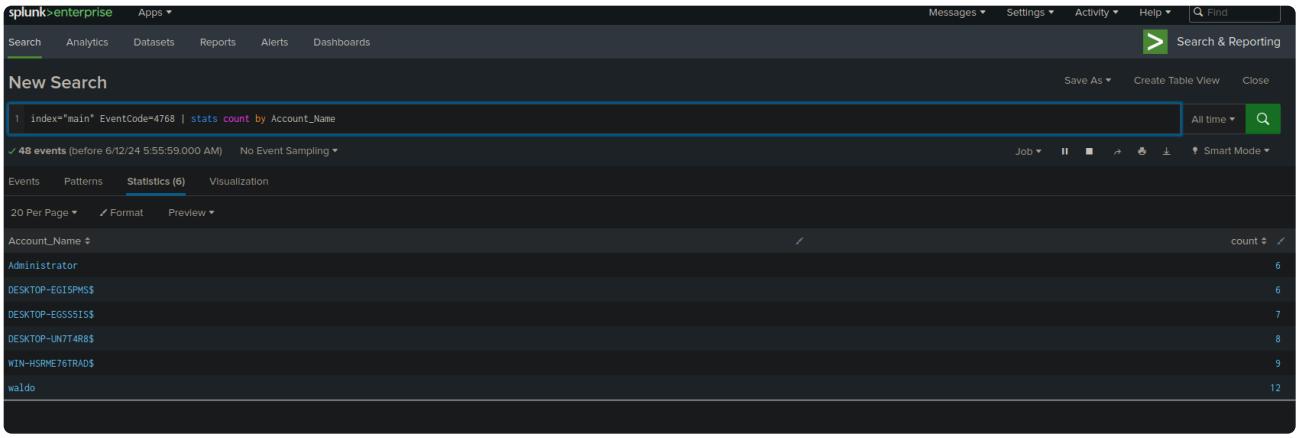
4768(S, F): A Kerberos authentication ticket (TGT) was requested.

Article • 10/20/2021 • 1 contributor

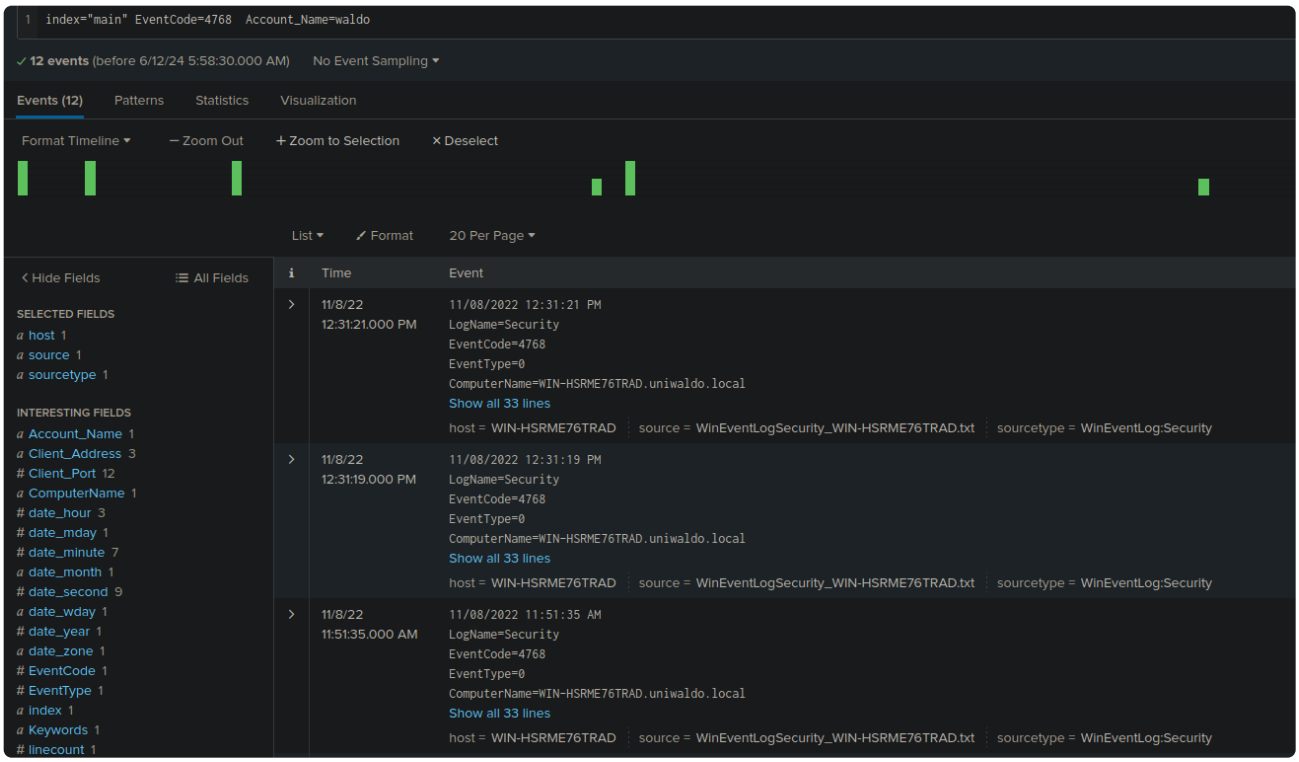
In this article

-> Doing some quick search shows that it is an TGT request.
-> However, some extra tinkering about how it is this helpful in detecting an attacker? Thought tgs requests might be helpful for an kerberoast detection.
-> We use the following spl to query to sort the count by account name with the appropriate event code

```
index="main" EventCode=4768 | stats count by Account_Name
```



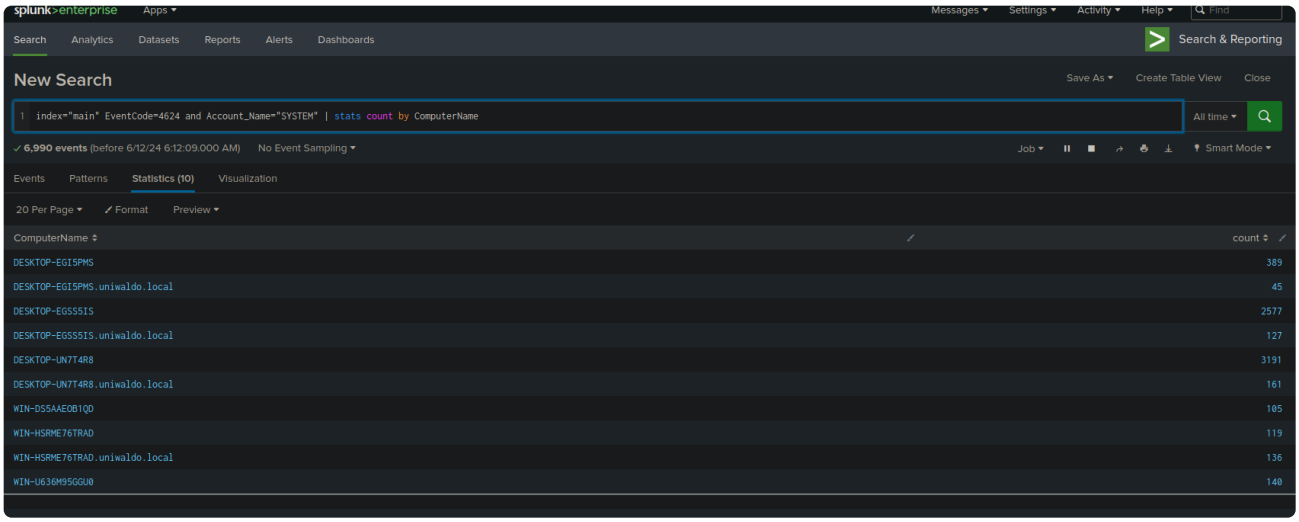
-> So we get waldo as the answer.
-> Extra investigation through waldo's behaviour, it is weird why he is requesting all these tickets in a short time span period:



-> Requesting two TGT in 2 seconds interval...? I'm not sure what kind of attacker would leave such a foot print :?, maybe an user needs help?

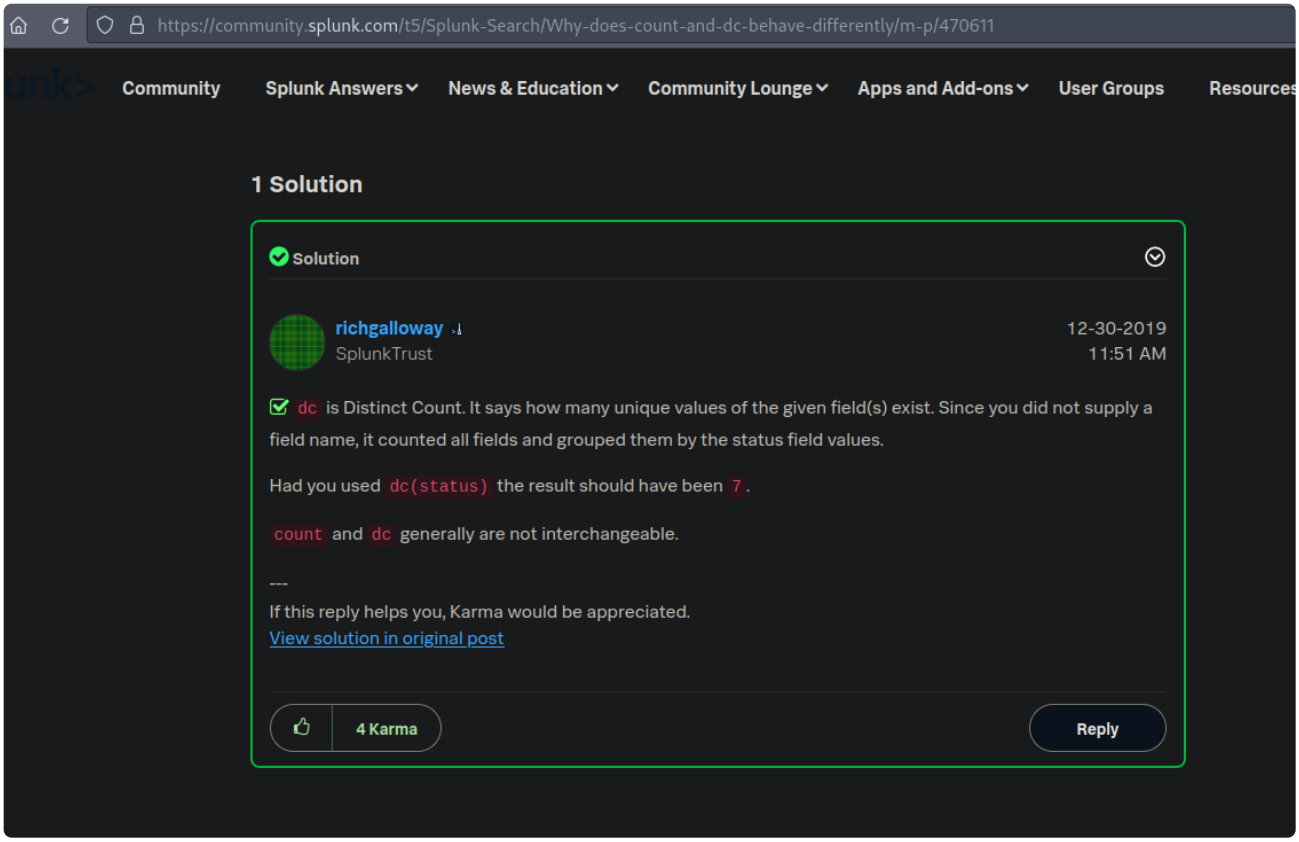
- Navigate to http://[Target IP]:8000, open the "Search & Reporting" application, and find through an SPL search against all 4624 events the count of distinct computers accessed by the account name SYSTEM. Enter it as your answer.
  -> This one is more straight forward (no need to research on event id), just make sure to have account name as SYSTEM as the spl query and filter for event code 4624, then pipe the output to count function as an initial try:

```
index="main" EventCode=4624 and Account_Name="SYSTEM" | stats count by ComputerName
```



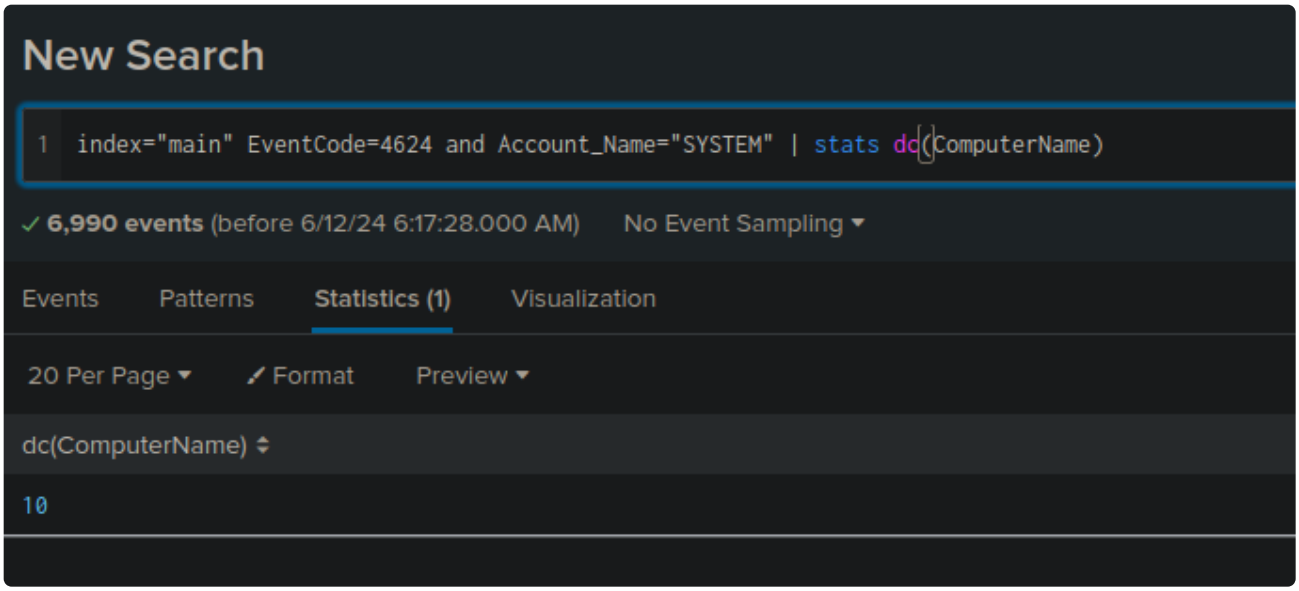
-> We know from the above that there are 10 distinct computers accessed, but the thing is how can we further improve our results.

-> Doing some googling online, we see from the following splunk post:



-> Hence, we can take use the dc() command as recommended in our SPL query:

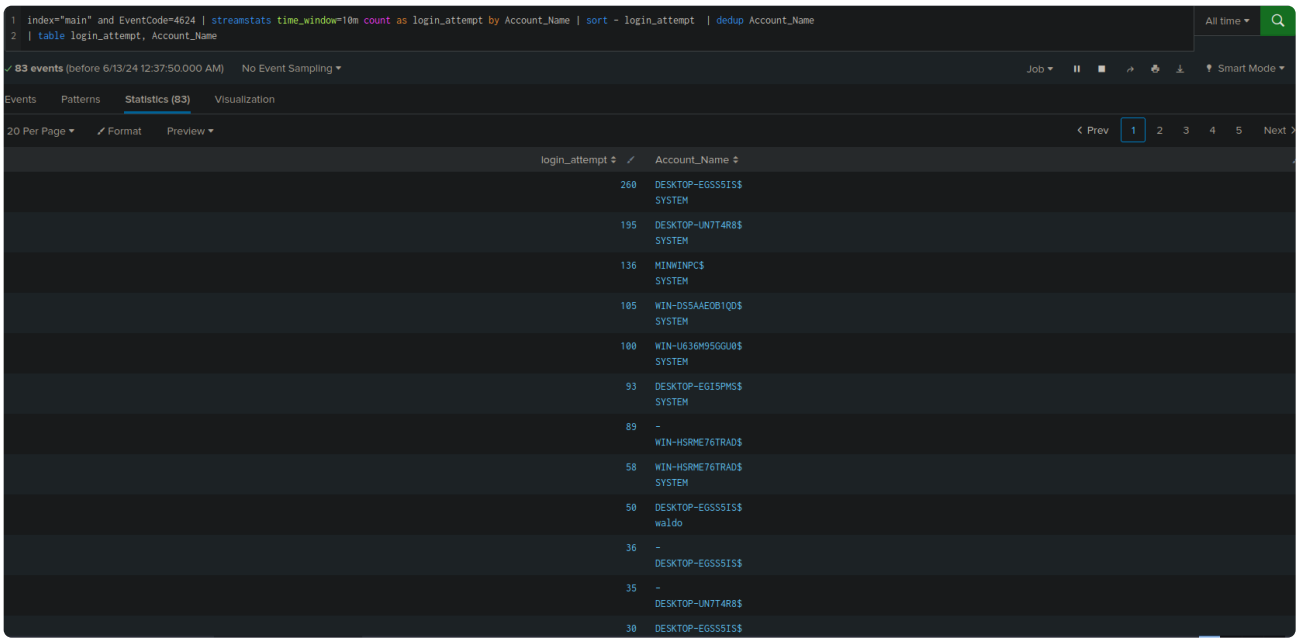
```
index="main" EventCode=4624 and Account_Name="SYSTEM" | stats
dc(ComputerName)
```



-> And we've just optimised our search query.

- Navigate to http://[Target IP]:8000, open the "Search & Reporting" application, and find through an SPL search against all 4624 events the account name that made the most login attempts within a span of 10 minutes. Enter it as your answer.
  -> It's similar to the previous question, but we have to take in account of login attempts within a span of 10 minutes.
  -> We would first count by account_name for login and give it a sliding window of 10 minutes, sort by login attempt, reduce the duplicates (just keep the largest value), then log it in a table, as follows:

```
index="main" and EventCode=4624 | streamstats time_window=10m count as
login_attempt by Account_Name | sort - login_attempt  | dedup
Account_Name | table login_attempt, Account_Name
```



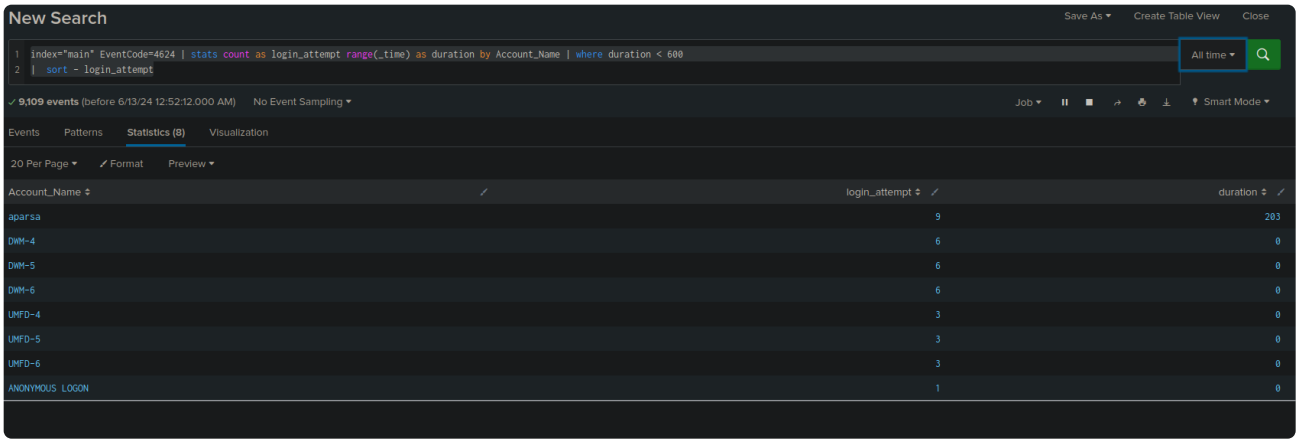
-> And we would have System or the Machine account as the login.

-> Also a quick note, this might not be the "most precise way", as the most precise way to look at the every 10 minute frame of user account logging in, which would be very inefficient for computation(? not too sure but is my guess).

-> However, the question actually meant that the account only logged on within a span of 10 minutes and never logged on (which demonstrates suspicious behaviour), but this information was not explicitly mentioned.

-> Hence, this means we have the construct the query as follows (first query for event 4624, then count for accounts that only logged in within a 10 minute time span and never logged in, using count, range and where filtering).

```
index="main" EventCode=4624 | stats count as login_attempt range(_time)
as duration by Account_Name | where duration < 600
| sort - login_attempt
```



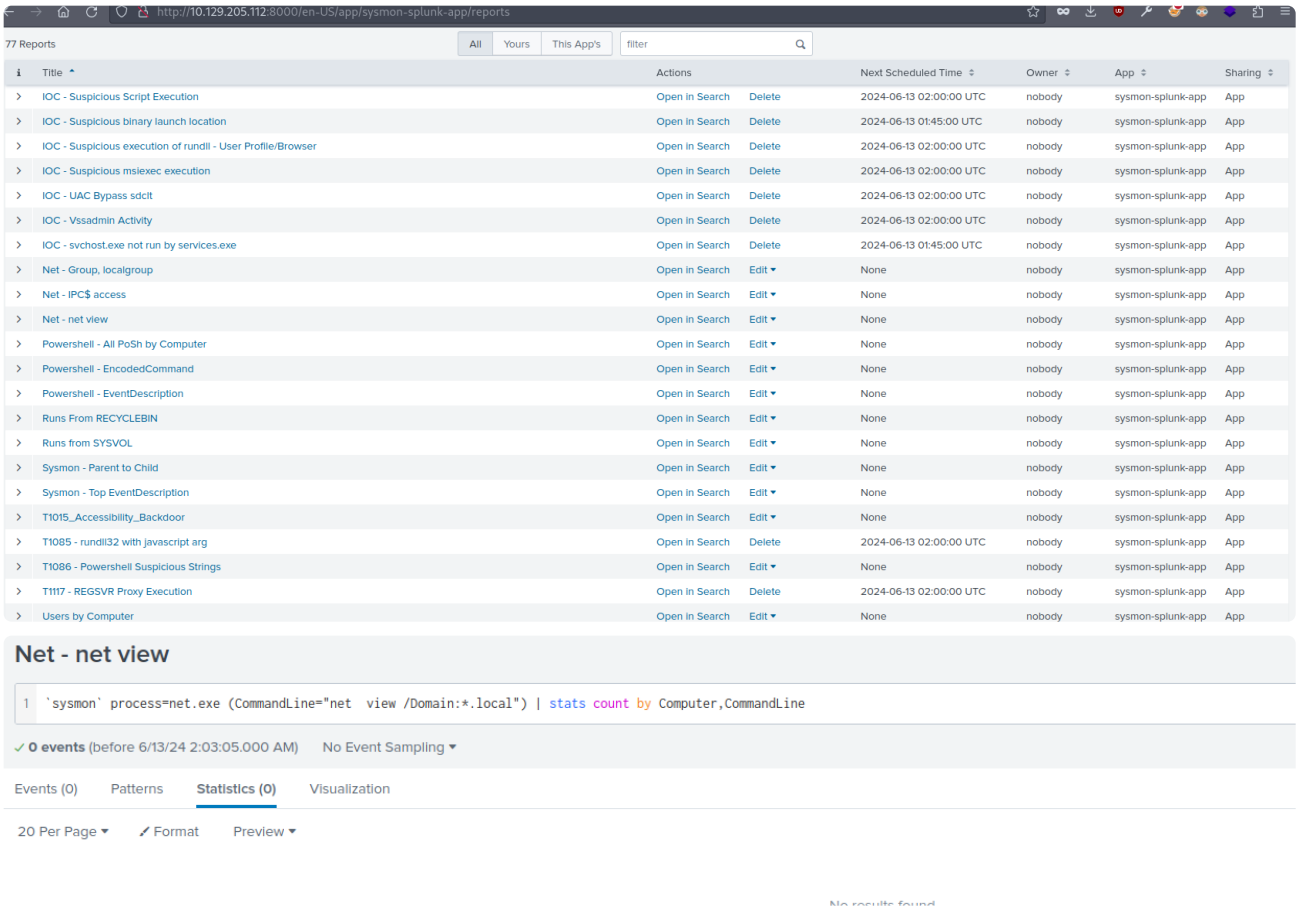
-> As such, our answer here would be aparsa.

Using Splunk Applications

Question

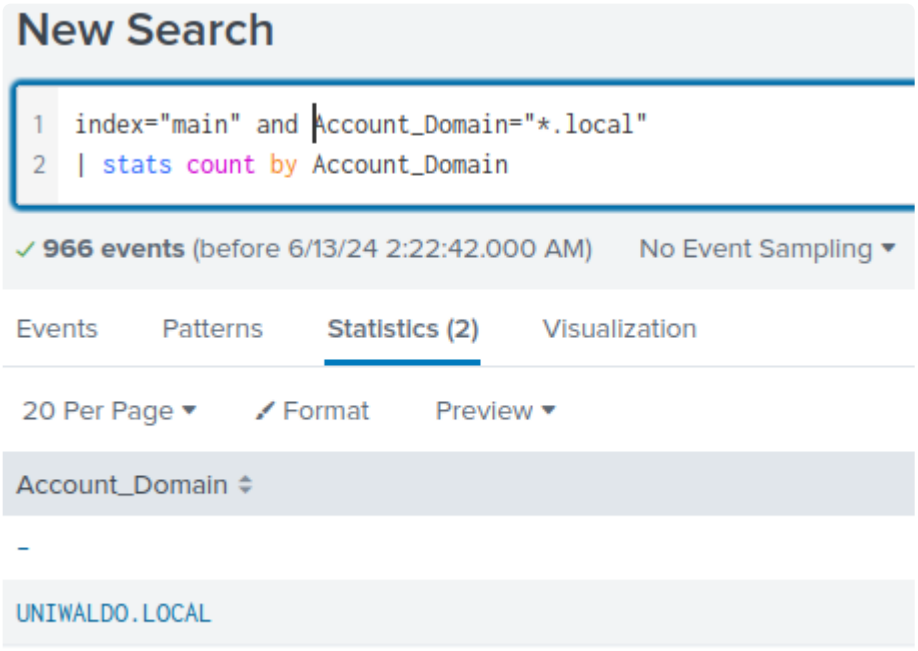
- Access the Sysmon App for Splunk and go to the "Reports" tab. Fix the search associated with the "Net - net view" report and provide the complete executed command as your answer. Answer format: net view /Domain:_.local

-> We look into the the search we need to change:



- To check the domain name, we query for the domain name, as follows:

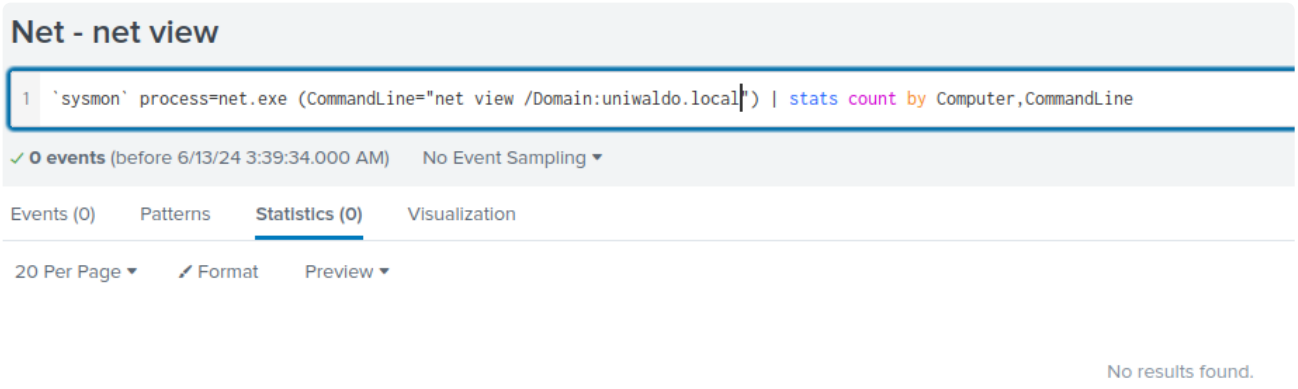
```
index="main" and Account_Domain="*.local"
| stats count by Account_Domain
```



-> Hence, the command we should execute is

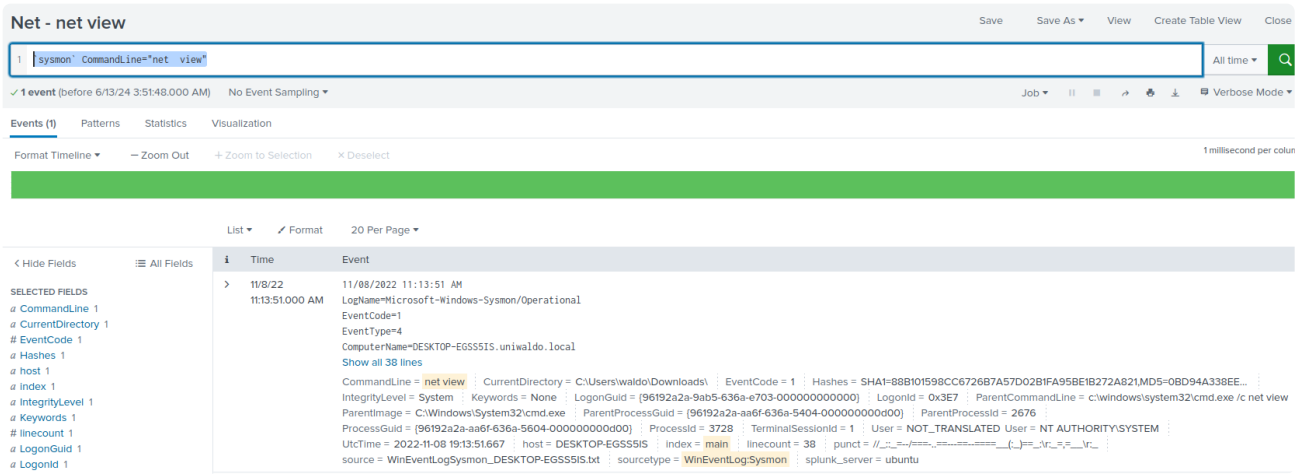
```
net view /Domain:uniwaldo.local
```

-> However, when we input the command, we don't get any result.



-> We know from within section that Sysmon Events with ID 11 does not have a field named Computer, but they do include a field called ComputerName.
-> We could also make the query simpler to being debug it (this is a common tactic, if something complicated doesn't work, try a simpler one command).
-> Note that we also need two spaces for "net view"
-> This is shown below:

```
`sysmon` CommandLine="net  view"
```



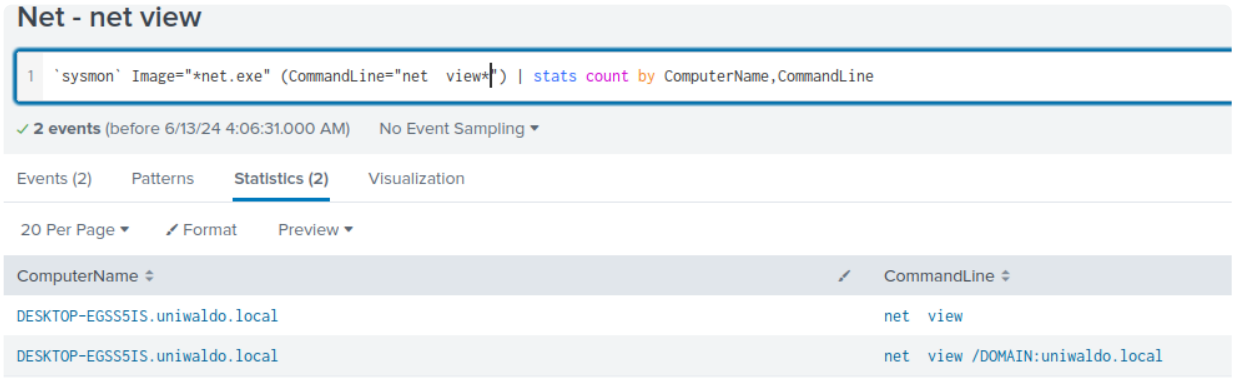
-> Now, we can analyse the log more in depth

```
11/8/22
11:13:51.000 AM    11/08/2022 11:13:51 AM
                   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=49550
                   Keywords=None
                   TaskCategory=Process Create (rule: ProcessCreate)
                   OpCode=Info
                   Message=Process Create:
                   RuleName: technique_id=T1018,technique_name=Remote System Discovery
                   UtcTime: 2022-11-08 19:13:51.667
                   ProcessGuid: {96192a2a-aa6f-636a-5604-000000000d00}
                   ProcessId: 3728
                   Image: C:\Windows\System32\net.exe
                   FileVersion: 10.0.19041.1 (WinBuild.160101.0800)
                   Description: Net Command
                   Product: Microsoft® Windows® Operating System
                   Company: Microsoft Corporation
                   OriginalFileName: net.exe
                   CommandLine: net  view
                   CurrentDirectory: C:\Users\waldo\Downloads\
                   User: NT AUTHORITY\SYSTEM
                   LogonGuid: {96192a2a-9ab5-636a-e703-000000000000}
                   LogonId: 0x3E7
                   TerminalSessionId: 1
                   IntegrityLevel: System
```

-> We see that the log should be calling "Process" instead of "Image" and Computer should be named ComputerName, like the below

```
`sysmon` Image="*net.exe" (CommandLine="net  view*") | stats count by ComputerName,CommandLine
```

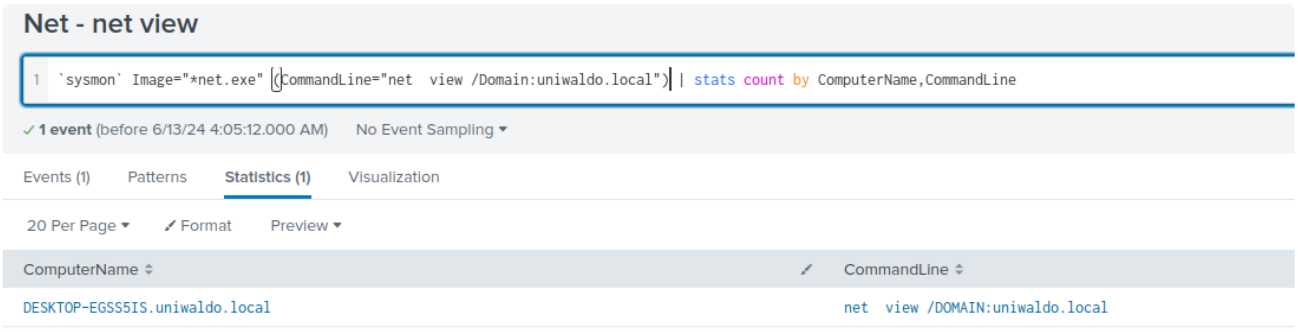
- Changing the query to the above, we see the following:



-> Now, we can change our query to the required format of querying the domain, as follows:

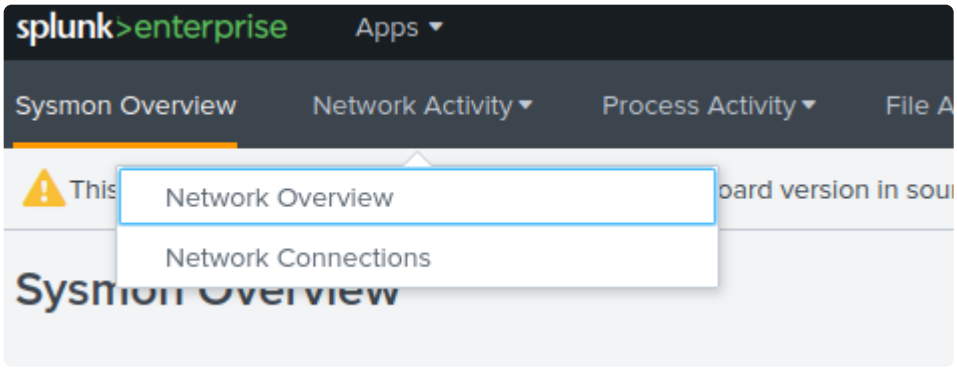
```
`sysmon` Image="*net.exe" (CommandLine="net  view /Domain:uniwaldo.local")
```

-> And we see the following:

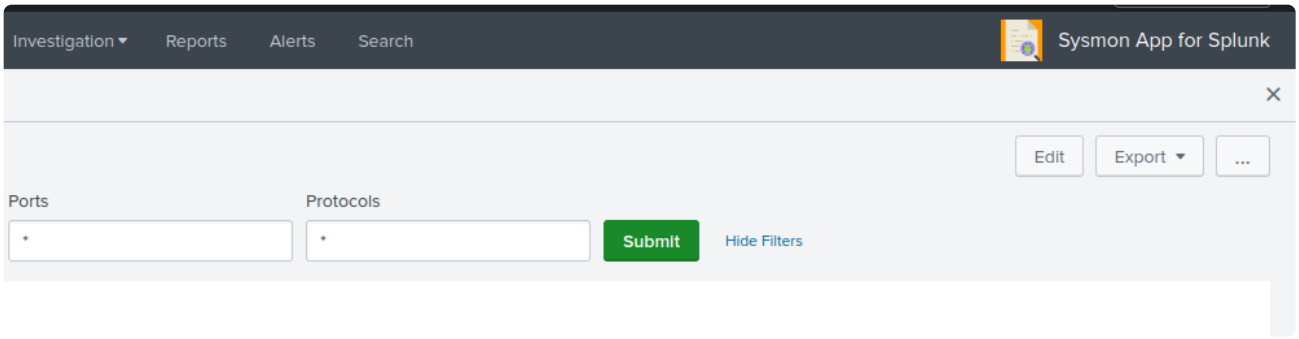


- Access the Sysmon App for Splunk, go to the "Network Activity" tab, and choose "Network Connections". Fix the search and provide the number of connections that SharpHound.exe has initiated as your answer.

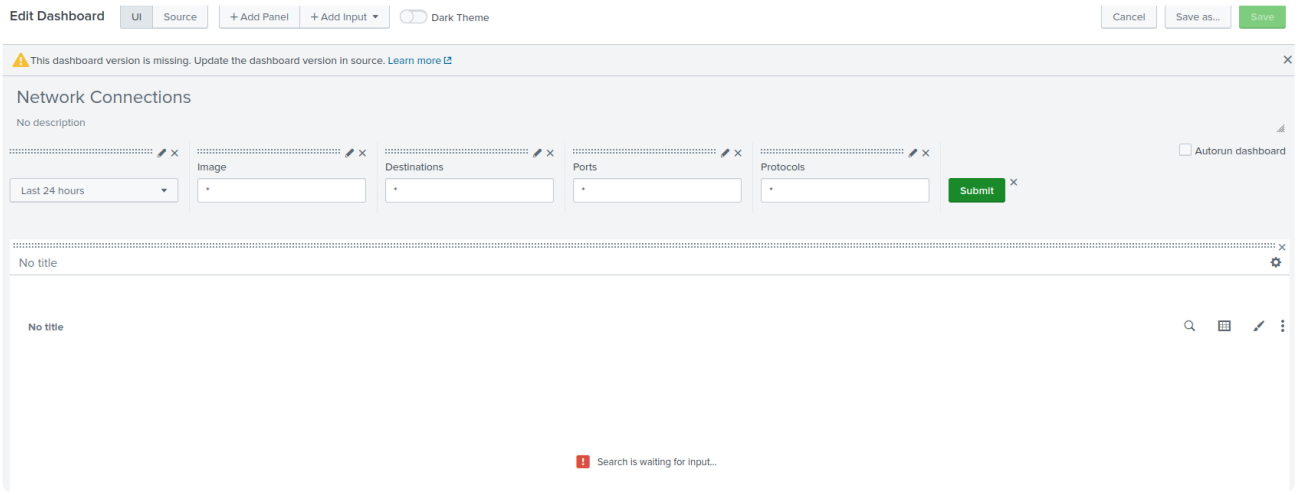
->We first go to the Network Connections page:



-> We click on edit

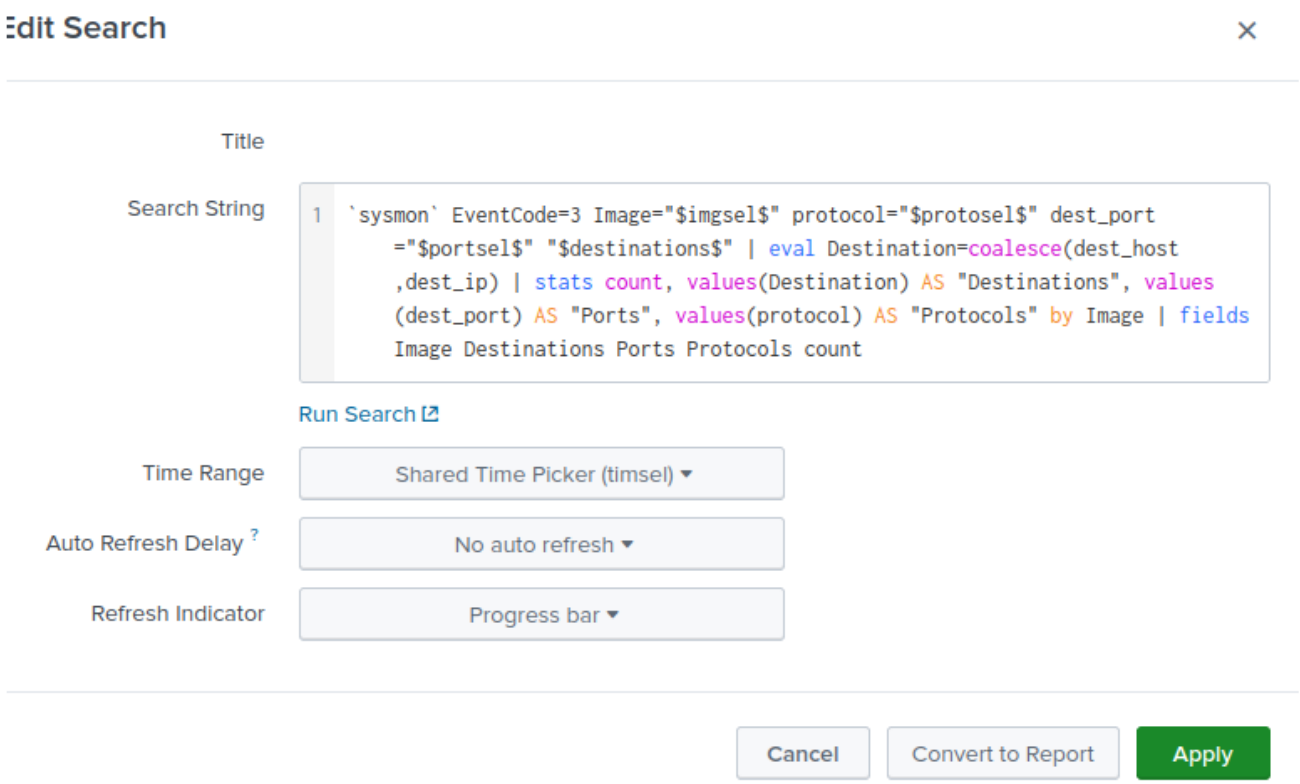


-> And we edit the search

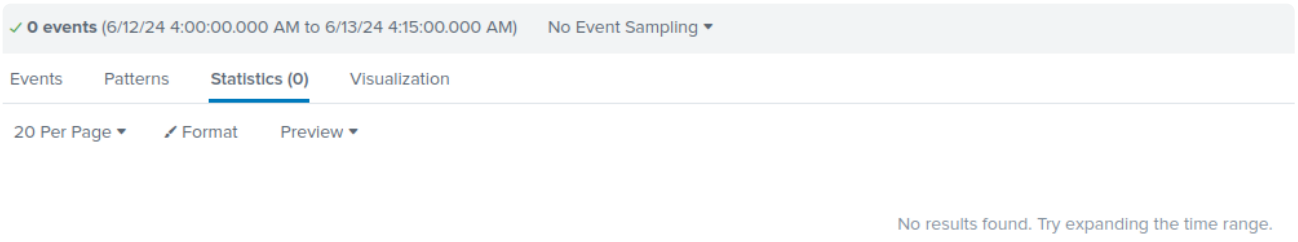


-> When we edit out search, we see the following:

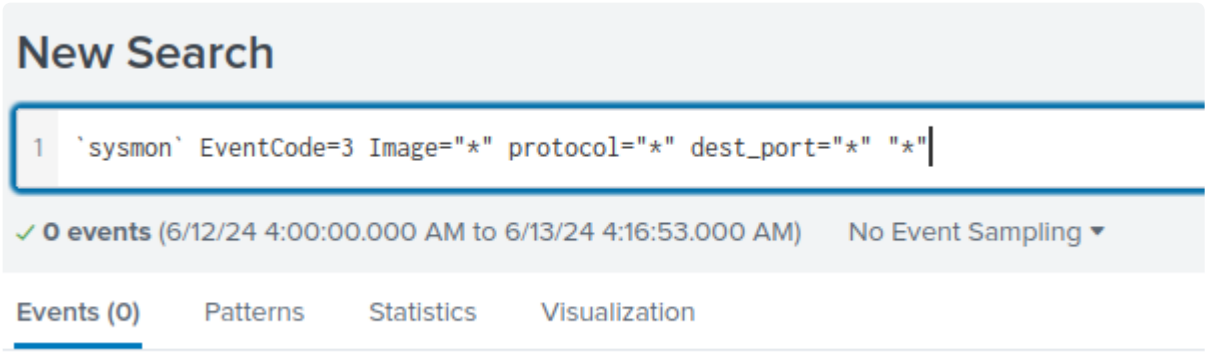
```
`sysmon` EventCode=3 Image="*" protocol="*" dest_port="*" "*" | eval
Destination=coalesce(dest_host,dest_ip) | stats count,
values(Destination) AS "Destinations", values(dest_port) AS "Ports",
values(protocol) AS "Protocols" by Image | fields Image Destinations
Ports Protocols count
```



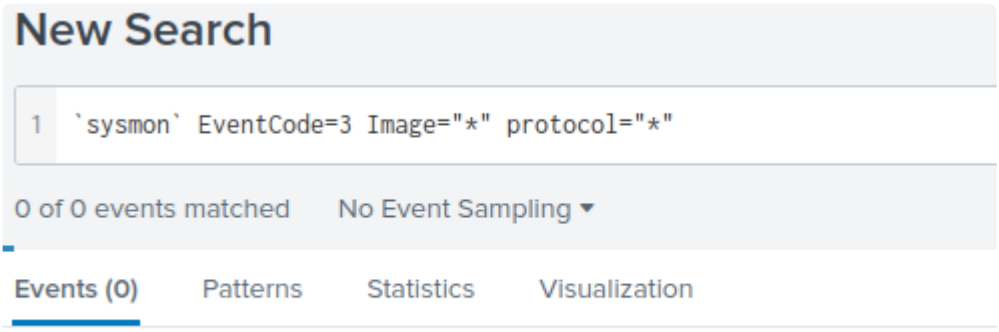
-> Running the search we get nothing:



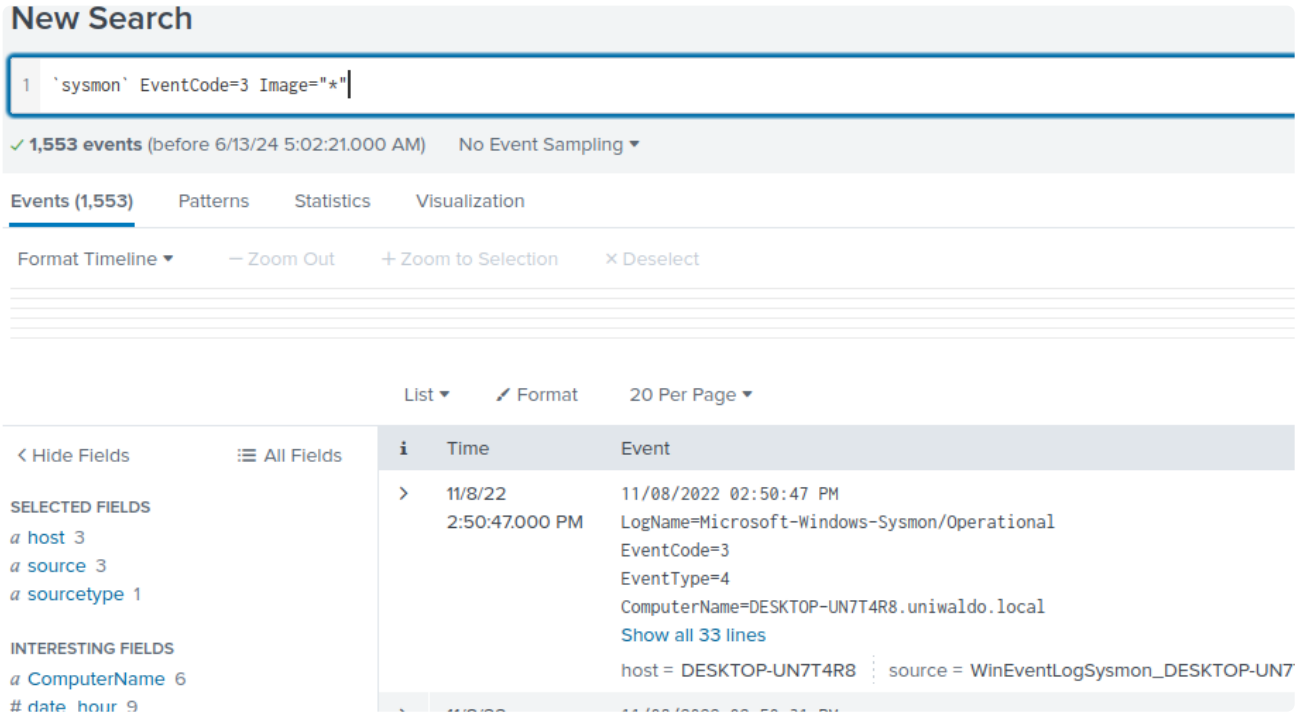
-> Again we utilise the same tactic, we trim down the query:



-> We still get nothing, so we trim down further



-> And we still get nothing, so we trim down further:



-> We finally get something,

-> And we get some search result.

-> Examining the field, we see that the destination fields looks like this:

```
SourceIsIpv6: false
SourceIp: 10.0.0.47
SourceHostname: -
SourcePort: 49789
SourcePortName: -
DestinationIsIpv6: false
DestinationIp: 20.42.65.85
DestinationHostname: -
DestinationPort: 443
DestinationPortName: -
Collapse
```

-> And the protocol is defined as "Protocol" instead:

Event

```
RecordNumber=30837
Keywords=None
TaskCategory=Network connection detected (rule: NetworkConnect)
OpCode=Info
Message=Network connection detected:
RuleName: technique_id=T1036,technique_name=Masquerading
UtcTime: 2022-11-08 22:50:45.961
ProcessGuid: {1cb7ffb5-dd03-636a-fd00-000000000d00}
ProcessId: 7192
Image: C:\Users\waldo\AppData\Local\Microsoft\Teams\current\Teams.exe
User: DESKTOP-UN7T4R8\waldo
Protocol: tcp
Initiated: true
SourceIsIpv6: false
SourceIp: 10.0.0.47
SourceHostname: -
SourcePort: 49789
SourcePortName: -
DestinationIsIpv6: false
DestinationIp: 20.42.65.85
DestinationHostname: -
DestinationPort: 443
DestinationPortName: -
```

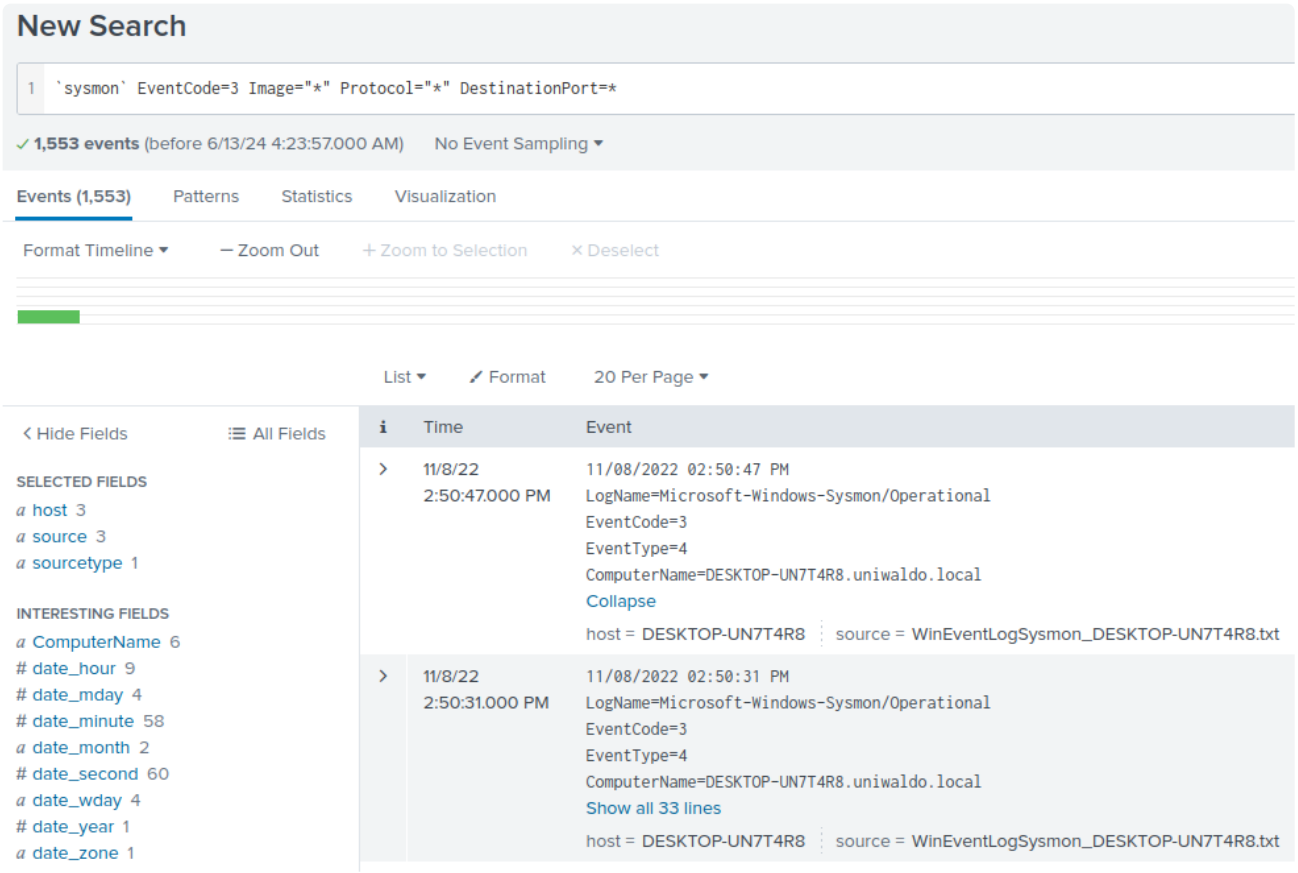
Collapse

host = DESKTOP-UN7T4R8 | source = WinEventLogSysmon_DESKTOP-UN7T4R8.txt

-> This shws that we should change the original dest_port to DestinationPort.

-> Let's give that a try:

```
`sysmon` EventCode=3 Image="*" Protocol="*" DestinationPort=*
```



-> It's a success, now we can fix the statement `eval Destination=coalesce(dest_host,dest_ip)` accordingly as well, changing dest_host to DestinationHostName and DestinationIp

```
`sysmon` EventCode=3 Image="*" Protocol="*" DestinationPort=* | eval
Destination=coalesce(DestinationHostname,DestinationIp) | stats count,
values(Destination) AS "Destinations", values(DestinationPort) AS
"Ports", values(protocol) AS "Protocols" by Image | fields Image
Destinations Ports Protocols count
```

-> Note that Coalesce is the returning of the first non-NULL value you give to it:

Solution

micahkemp
Champion

02-05-2018
02:57 PM

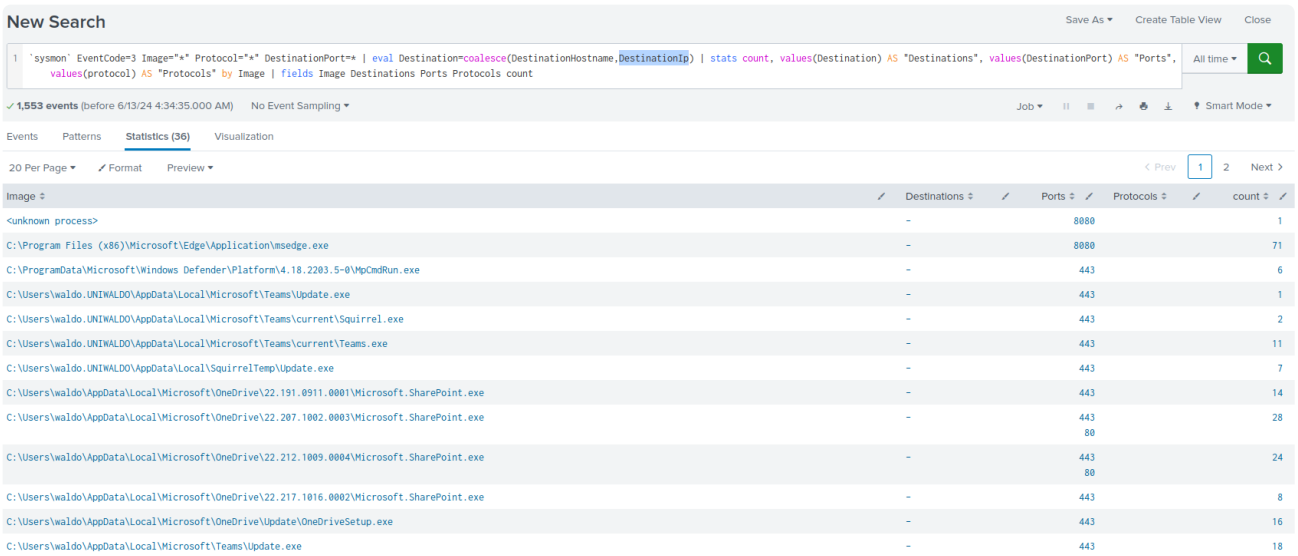
It sounds like `coalesce` is doing exactly what it's supposed to do: return the first non-NULL value you give it.

Perhaps you are looking for `mvappend`, which will put all of the values passed to it into the result:

```
| eval allvalues=mvappend(value1, value2)
```

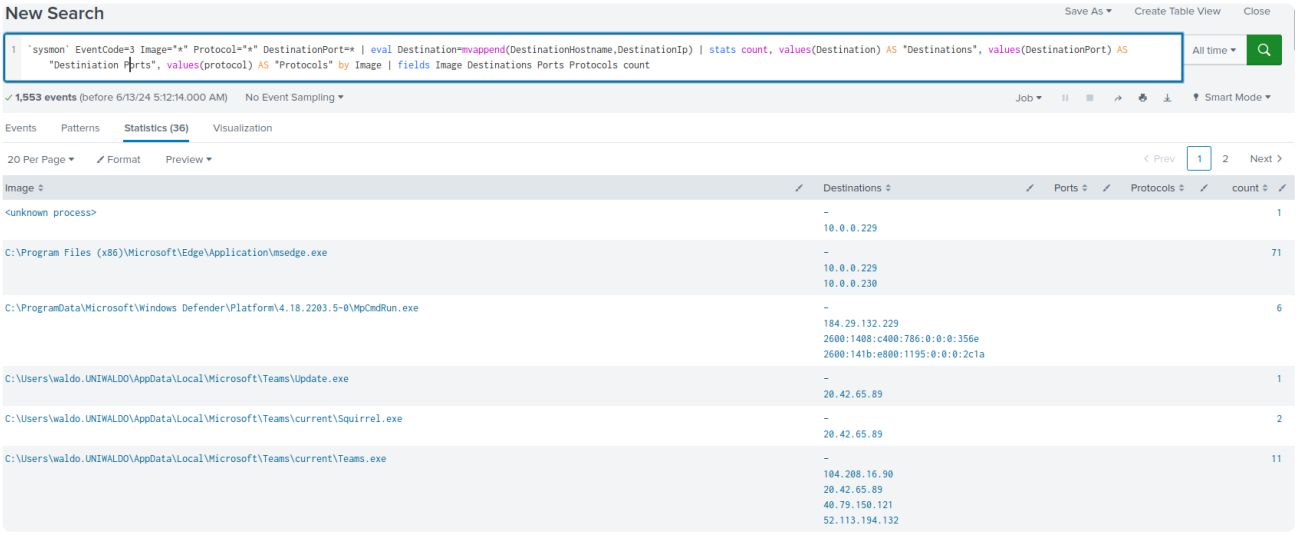
View solution in original post

-> now we obtain the following:



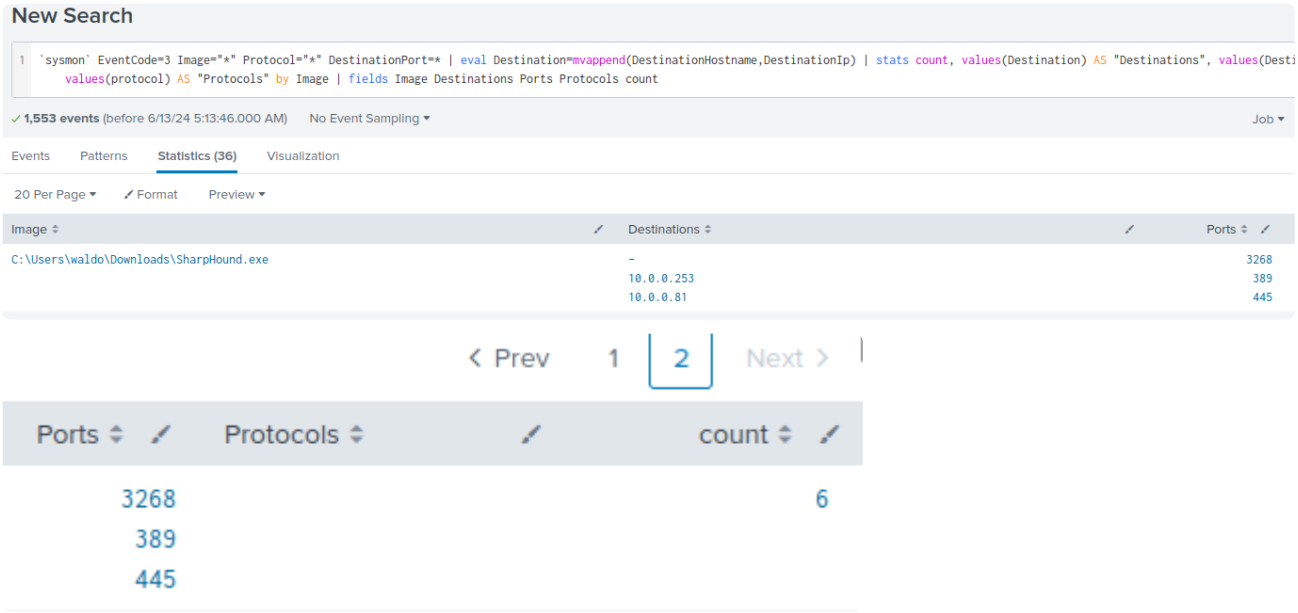
-> we see that we do get some results, but we are getting empty values (represented as -) in Destination. This most likely happened as we have lots of empty values for it.
-> To resolve this, we'll use the mvappend function as suggested by the post above, with some slight tweak in wording:

```
`sysmon` EventCode=3 Image="*" Protocol="*" DestinationPort=* | eval
Destination=mvappend(DestinationHostname,DestinationIp) | stats count,
values(Destination) AS "Destinations", values(DestinationPort) AS
"Ports", values(protocol) AS "Protocols" by Image | fields Image
Destinations Ports Protocols count
```



-> We see that we are content with the following result above.

-> Looking at the next page, we see there is an SharpHound process.



-> With 6 connection count.

-> Note: While changing the value for Destination port from = `"*" "*" to "*"` is alright, we should keep in mind that an more ideal approach would be to just change the name and see if anything occurs first, i.e.:

```
`sysmon` EventCode=3 Image="*" Protocol="*" DestinationPort="*" "*"
```

-> If the above doesn't work, then change it to:

```
`sysmon` EventCode=3 Image="*" Protocol="*" DestinationPort="*"
```

Investigating With Splunk

Intrusion Detection With Splunk (Real-world Scenario)

Question

- Navigate to http://[Target IP]:8000, open the "Search & Reporting" application, and find through an SPL search against all data the other process that dumped lsass. Enter its name as your answer. Answer format: _.exe
  ->If we use the query given in the section:

```
index="main" CallTrace="*UNKNOWN*" SourceImage!="*Microsoft.NET*"
CallTrace!=*ni.dll* CallTrace!=*clr.dll CallTrace!=*wow64*
SourceImage!="C:\\Windows\\Explorer.EXE" | where
SourceImage!=TargetImage | stats count by SourceImage, TargetImage,
CallTrace
```

SourceImage	TargetImage	CallTrace
C:\Windows\System32\notepad.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\SYSTEM32\ntdll.dll+9e664|C:\Windows\System32\KERNELBASE.dll+8\KERNELBASE.dll+7226|C:\Windows\System32\KERNEL32.DLL+1c7b4|UNKNOWN(0000
C:\Windows\System32\notepad.exe	C:\Windows\system32\lsass.exe	C:\Windows\SYSTEM32\ntdll.dll+9d4c4|UNKNOWN(00000288CF8F5445)
C:\Windows\System32\rundll32.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\SYSTEM32\ntdll.dll+9e8f4|C:\Windows\System32\KERNELBASE.dll+8\KERNELBASE.dll+7226|C:\Windows\System32\KERNEL32.DLL+1c7b4|UNKNOWN(0000
C:\Windows\System32\rundll32.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\SYSTEM32\ntdll.dll+9e8f4|C:\Windows\System32\KERNELBASE.dll+8\KERNELBASE.dll+7226|C:\Windows\System32\KERNEL32.DLL+1c7b4|UNKNOWN(0000
C:\Windows\System32\rundll32.exe	C:\Windows\System32\notepad.exe	C:\Windows\SYSTEM32\ntdll.dll+9e8f4|C:\Windows\System32\KERNELBASE.dll+8\KERNELBASE.dll+7226|C:\Windows\System32\KERNEL32.DLL+1c7b4|UNKNOWN(0000
C:\Windows\System32\rundll32.exe	C:\Windows\System32\notepad.exe	C:\Windows\SYSTEM32\ntdll.dll+9e8f4|C:\Windows\System32\KERNELBASE.dll+8\KERNELBASE.dll+7226|C:\Windows\System32\KERNEL32.DLL+1c7b4|UNKNOWN(0000
C:\Windows\System32\rundll32.exe	C:\Windows\system32\lsass.exe	C:\Windows\SYSTEM32\ntdll.dll+9dd34|UNKNOWN(000002E53982549A)

-> We see that the other process that dumped LSASS is `rundll32.exe`
-> Further investigation with the following spl shows that

```
index="main" CallTrace="*UNKNOWN*" SourceImage!="*Microsoft.NET*"
CallTrace!=*ni.dll* CallTrace!=*clr.dll CallTrace!=*wow64*
SourceImage!="C:\\Windows\\Explorer.EXE" | where
SourceImage!=TargetImage  | search
SourceImage="C:\\Windows\\System32\\rundll32.exe"
```

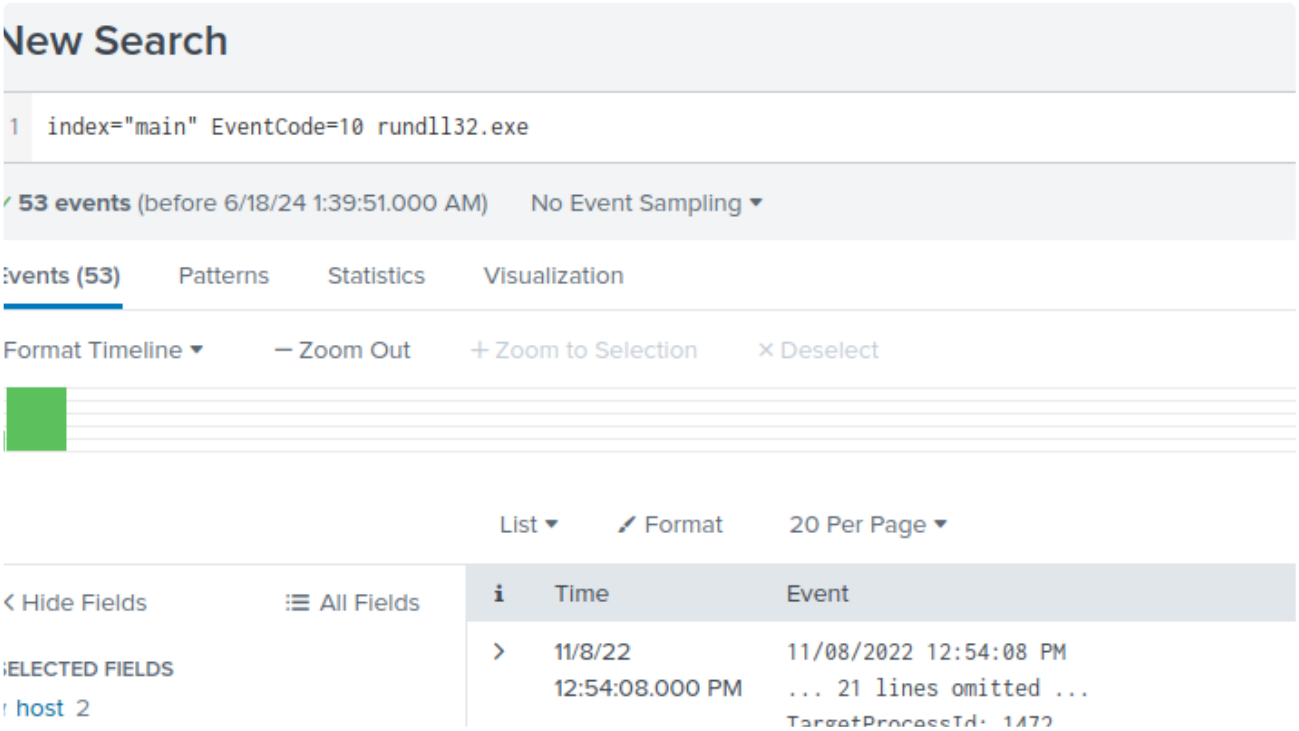
i	Time	Event
>	11/6/22 11:52:33.000 AM	11/06/2022 11:52:33 AM LogName=Microsoft-Windows-Sysmon/Operational EventCode=10 EventType=4 ComputerName=DESKTOP-EGSS5IS User=NOT_TRANSLATED Sid=S-1-5-18 SidType=0 SourceName=Microsoft-Windows-Sysmon Type=Information RecordNumber=40442 Keywords=None TaskCategory=Process accessed (rule: ProcessAccess) OpCode=Info Message=Process accessed: RuleName: technique_id=T1003,technique_name=Credential Dumping UtcTime: 2022-11-06 19:52:33.116 SourceProcessGUID: {96192a2a-09d5-6368-3b05-000000000900} SourceProcessId: 2964 SourceThreadId: 7468 SourceImage: C:\Windows\System32\rundll32.exe TargetProcessGUID: {96192a2a-f6ae-6367-0c00-000000000900} TargetProcessId: 656 TargetImage: C:\Windows\system32\lsass.exe GrantedAccess: 0x1FFFFF CallTrace: C:\Windows\SYSTEM32\ntdll.dll+9dd34|UNKNOWN(000002E53982549A) SourceUser: DESKTOP-EGSS5IS\waldo TargetUser: NT AUTHORITY\SYSTEM Collapse host = DESKTOP-EGSS5IS | source = WinEventLogSysmon_DESKTOP-EGSS5IS.txt | sourcetype = WinEventLog:Sysmon

-> It is indeed a Credential Dumping techniques.

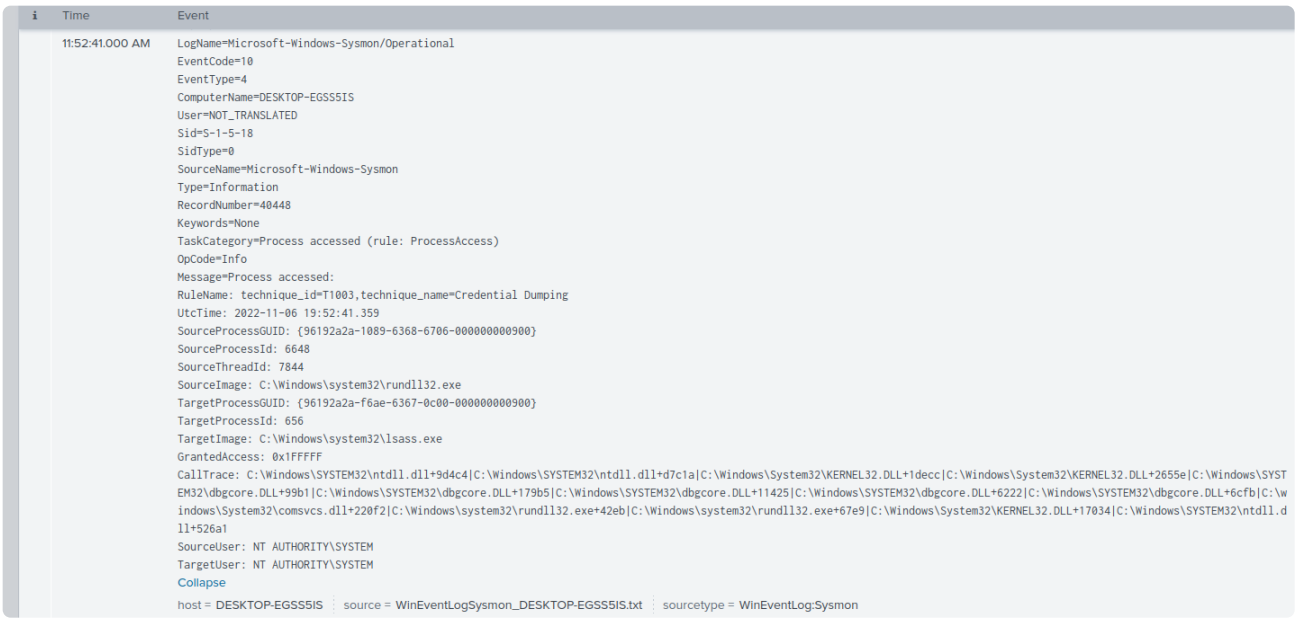
- Navigate to http://[Target IP]:8000, open the "Search & Reporting" application, and find through SPL searches against all data the method through which the other process dumped lsass. Enter the misused DLL's name as your answer. Answer format: _.dll
  -> We create following spl (focus on dll's that were being used)

```
index="main" EventCode=10 rundll32.exe
```

-> We have 53 events:



-> We have some queries like this:



-> We want to refine our queries to specifically focus on Source as rundll32.exe and Targeet as lsass.exe

```
index="main" EventCode=10 SourceImage="*rundll32.exe"
TargetImage="*lsass.exe"
```

-> We now have 7 queries and we got results like this, with the first event happening first then the second happening:

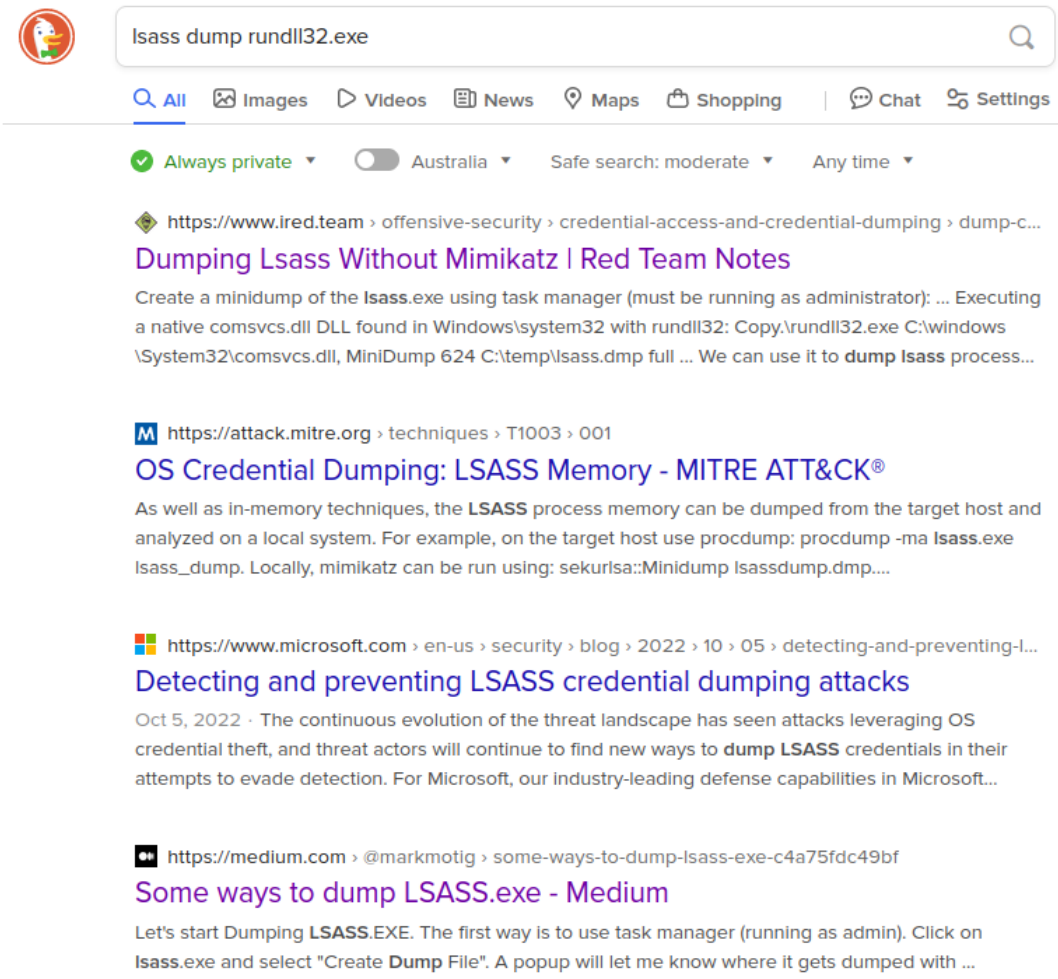
```
11/06/2022 11:52:33 AM
LogName=Microsoft-Windows-Sysmon/Operational
EventCode=10
EventType=4
ComputerName=DESKTOP-EGSS5IS
User=NOT_TRANSLATED
Sid=S-1-5-18
SidType=0
SourceName=Microsoft-Windows-Sysmon
Type=Information
RecordNumber=40442
Keywords=None
TaskCategory=Process accessed (rule: ProcessAccess)
OpCode=Info
Message=Process accessed:
RuleName: technique_id=T1003,technique_name=Credential Dumping
UtcTime: 2022-11-06 19:52:33.116
SourceProcessGUID: {96192a2a-09d5-6368-3b05-000000000900}
SourceProcessId: 2964
SourceThreadId: 7468
SourceImage: C:\Windows\System32\rundll32.exe
TargetProcessGUID: {96192a2a-f6ae-6367-0c00-000000000900}
TargetProcessId: 656
TargetImage: C:\Windows\system32\lsass.exe
GrantedAccess: 0x1FFFFF
CallTrace: C:\Windows\SYSTEM32\ntdll.dll+9dd34|UNKNOWN(000002E53982549A)
SourceUser: DESKTOP-EGSS5IS\waldo
TargetUser: NT AUTHORITY\SYSTEM
Collapse
host = DESKTOP-EGSS5IS | source = WinEventLogSysmon_DESKTOP-EGSS5IS.txt | sourcetype = WinEventLog:Sysmon
```

```
EventType=4
ComputerName=DESKTOP-EGSS5IS.uniwaldo.local
User=NOT_TRANSLATED
Sid=S-1-5-18
SidType=0
SourceName=Microsoft-Windows-Sysmon
Type=Information
RecordNumber=51579
Keywords=None
TaskCategory=Process accessed (rule: ProcessAccess)
OpCode=Info
Message=Process accessed:
RuleName: technique_id=T1003,technique_name=Credential Dumping
UtcTime: 2022-11-08 19:46:07.171
SourceProcessGUID: {96192a2a-b1ff-636a-d805-000000000d00}
SourceProcessId: 1624
SourceThreadId: 8340
SourceImage: C:\Windows\system32\rundll32.exe
TargetProcessGUID: {96192a2a-9ab5-636a-0c00-000000000d00}
TargetProcessId: 640
TargetImage: C:\Windows\system32\lsass.exe
GrantedAccess: 0x1FFFFF
CallTrace: C:\Windows\SYSTEM32\ntdll.dll+9d4c4|C:\Windows\SYSTEM32\ntdll.dll+d7c1a|C:\Windows\System32\KERNEL32.DLL+1decc|C:\Windows\System32\KERNEL32.DLL+2655e|C:\Windows\SYSTEM32\dbgcore.DLL+99b1|C:\Windows\SYSTEM32\dbgcore.DLL+179b5|C:\Windows\SYSTEM32\dbgcore.DLL+11425|C:\Windows\SYSTEM32\dbgcore.DLL+6222|C:\Windows\SYSTEM32\dbgcore.DLL+6cfb|C:\Windows\System32\comsvcs.dll+220f2|C:\Windows\system32\rundll32.exe+42eb|C:\Windows\system32\rundll32.exe+67e9|C:\Windows\System32\KERNEL32.DLL+17034|C:\Windows\SYSTEM32\ntdll.dll+526a1
SourceUser: NT AUTHORITY\SYSTEM
TargetUser: NT AUTHORITY\SYSTEM
Collapse
host = DESKTOP-EGSS5IS | source = WinEventLogSysmon_DESKTOP-EGSS5IS.txt | sourcetype = WinEventLog:Sysmon
```

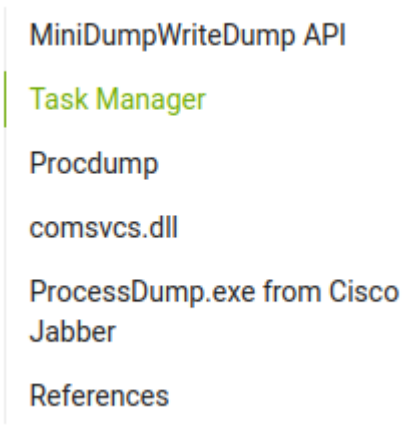
-> This meas that firstly, ntdll.dll calls an unknown memory segment into it, which in turns loads a bunch of dll into it.

-> Now, thinking from the attacker's POV, their tactics (unless its a zero day novel technique, which says alot of the complexity of the attacker's technique) would likely to be documented online.

-> Hence, we would be able to search something like "lsass dump rundll32" and get some result of what techniques that attackers used:



-> We look into the first result and we see that it talks about dumping lsass using various techniques:

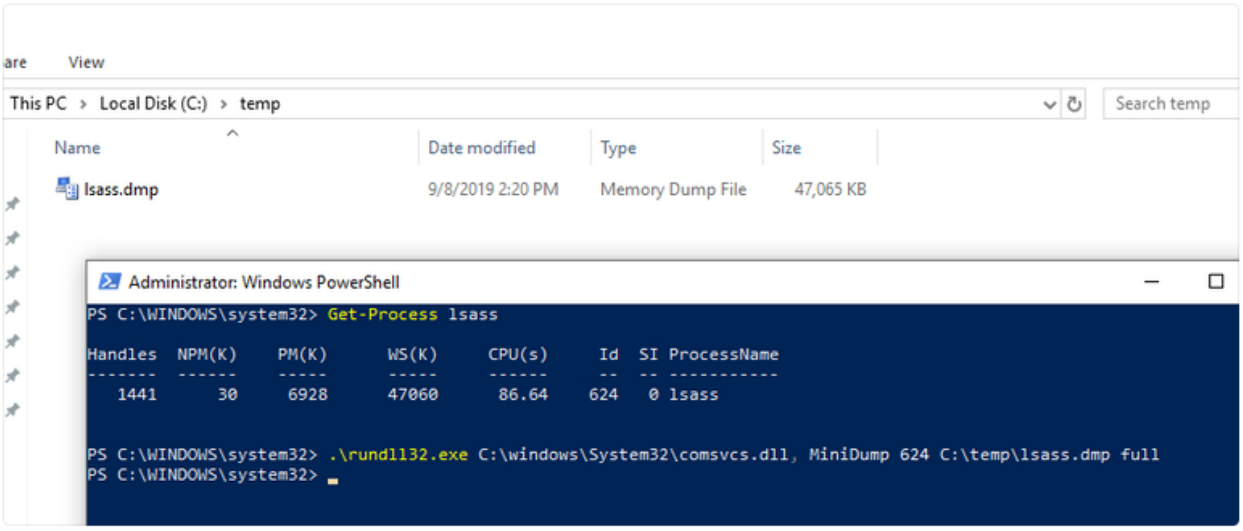


-> the comsvcs.dll looks interesting and relevant to us here:

comsvcs.dll

Executing a native comsvcs.dll DLL found in Windows\system32 with rundll32:

```
.\rundll32.exe C:\windows\System32\comsvcs.dll, MiniDump 624 C:\temp\lsass.dmp full
```



-> Hence, we can see that comsvc.dll is likely the dll being misused.

- Navigate to http://[Target IP]:8000, open the "Search & Reporting" application, and find through an SPL search against all data any suspicious loads of clr.dll that could indicate a C# injection/execute-assembly attack. Then, again through SPL searches, find if any of the suspicious processes that were returned in the first place were used to temporarily execute code. Enter its name as your answer. Answer format: _.exe
- We know from windows event log that C# injection requires the detection of loading "clr.dll" or "clrjit.dll", which has an sysmon event id of 7.

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=7
ImageLoaded="*clr.dll" | stats count by Image | sort - count
```

C:\Users\waldo\AppData\Local\Microsoft\Teams\Update.exe	15
C:\Windows\System32\notepad.exe	12
C:\Program Files\Corsair\CORSAIR iCUE 4 Software\Corsair.Service.CpuIdRemote64.exe	9
C:\Program Files\Corsair\CORSAIR iCUE 4 Software\Corsair.Service.exe	9
C:\Windows\System32\rundll32.exe	4
C:\Users\waldo\AppData\Local\Temp\pxxebibs.0cx\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe	3
C:\Users\waldo\AppData\Local\Temp\2l14lwvz.xmw\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe	2
C:\Users\waldo\AppData\Local\Temp\5z2nzjvj.vjn\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe	2
C:\Users\waldo\AppData\Local\Temp\5zxxbqih.b0p\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe	2
C:\Users\waldo\AppData\Local\Temp\y0gfm1ry.auq\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe	2
C:\Users\waldo\AppData\Local\Temp\yoz0otta.trj\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe	2
C:\Users\waldo\Downloads\randomfile.exe	2

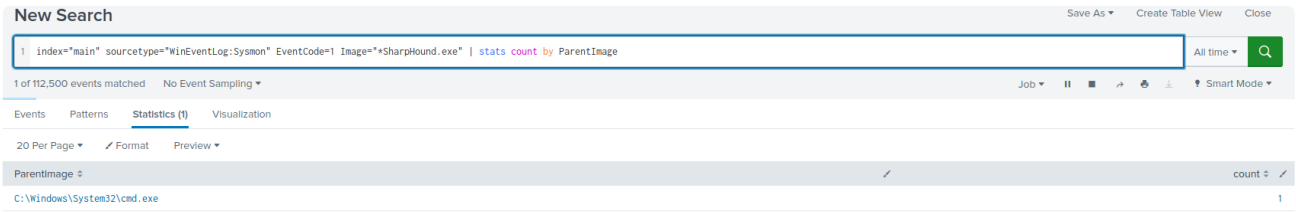
Image	count
C:\Users\waldo.UNIWALDO\AppData\Local\Microsoft\Teams\Update.exe	1
C:\Users\waldo.UNIWALDO\AppData\Local\Microsoft\Teams\current\Squirrel.exe	1
C:\Users\waldo.UNIWALDO\AppData\Local\SquirrelTemp\Update.exe	1
C:\Users\waldo\Downloads\SharpHound.exe	1

-> We see some suspicious process including sharphound.exe, notepad.exe, rundll32.exe and randomfile.exe that has very little occurrence, as a large occurrence usually indicates a normal working process.

-> Now, we look for unusual parent-child trees relationships, as unusual parent-child trees are always suspicious (event generation by parent)

-> Analysing sharphound

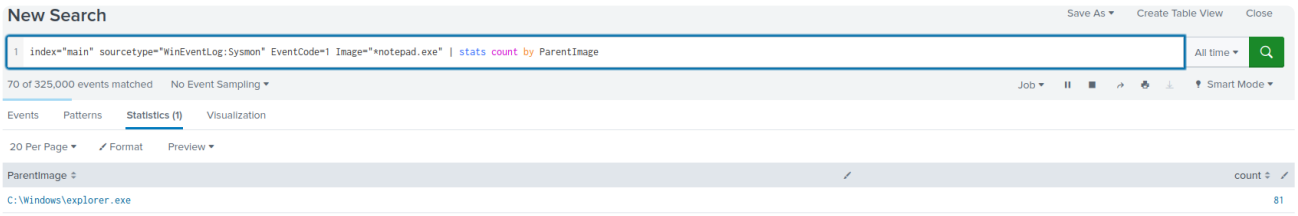
```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1
Image="*SharpHound.exe" | stats count by ParentImage
```



-> Looks standard, looks like some recon done by the attacker using cmd.

-> Analysing notepad

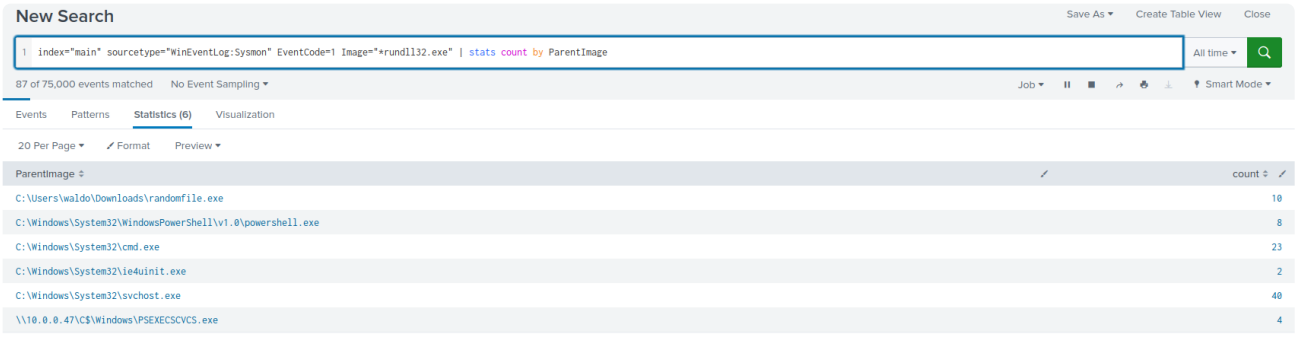
```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1
Image="*notepad.exe" | stats count by ParentImage
```



-> Explorer running notepad, not too sure but seems ok?

-> Analysing rundll32

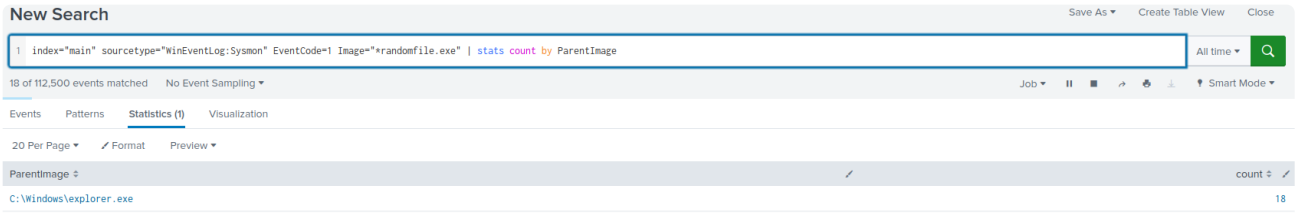
```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1
Image="*rundll32.exe" | stats count by ParentImage
```



-> Very weird, have mulitple weird files executing it, with the standout being randomfile being a parent of rundll32 definitely very weird parent-child relationships.

-> Analysing randomfile

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1
Image="*randomfile.exe" | stats count by ParentImage
```

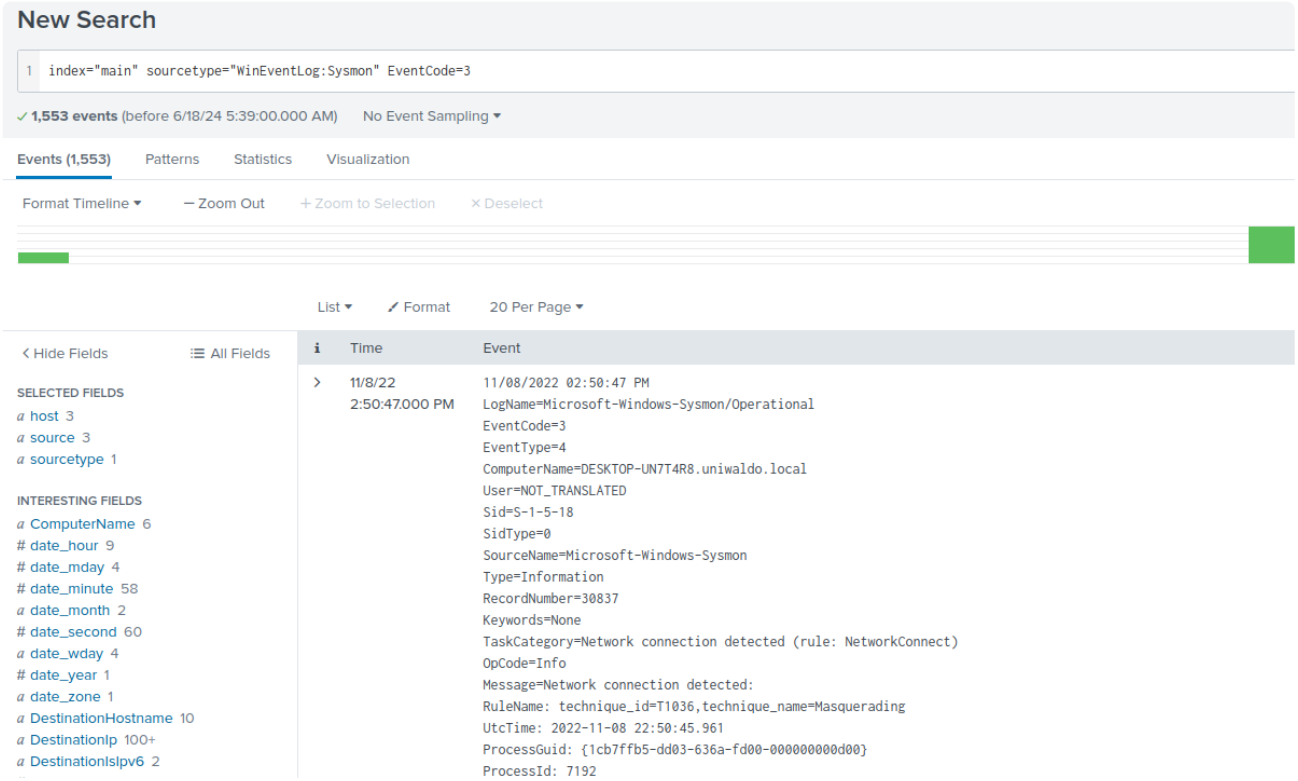


-> Overall, we can say that rundll32 is the most suspicious process executing code because it has multiple unusual parent-child relations.
-> We may also say that it is used as an sacrificial process (creates a new logon session an passes tickets to that session and does work in that logon session hence "sacrificial" .

Failure to create an sacrificial process may result in the service being taken down, e.g. overwriting of an Kerberos ticket of the local machine ticket account).

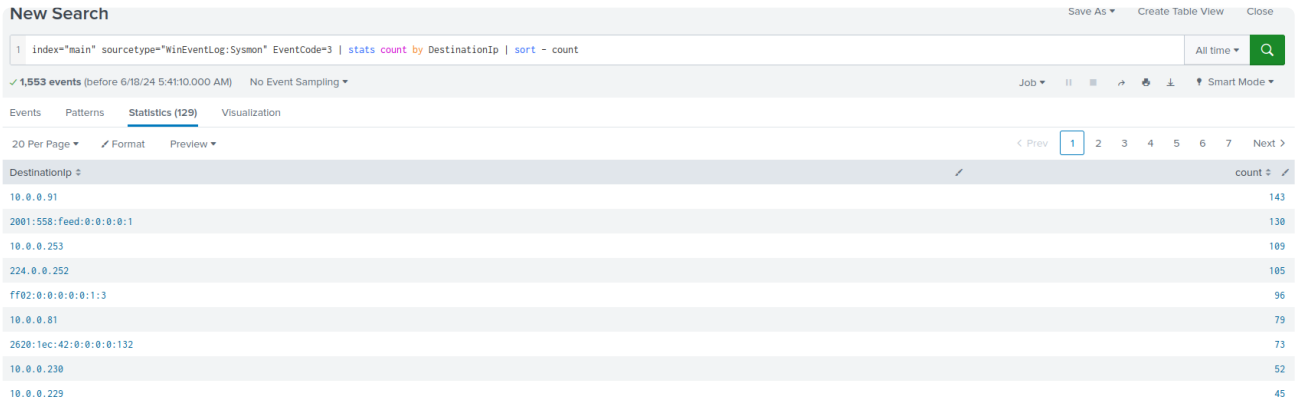
- Navigate to http://[Target IP]:8000, open the "Search & Reporting" application, and find through SPL searches against all data the two IP addresses of the C2 callback server. Answer format: 10.0.0.1XX and 10.0.0.XX
  -> Search for sysmon eventcode 3

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=3
```



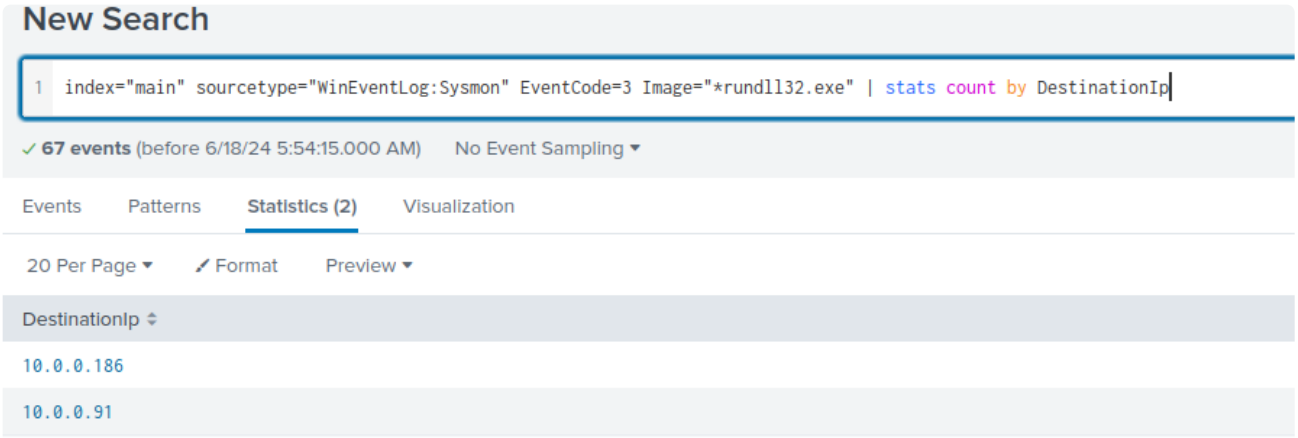
-> Alot of events, let search by count of IP addresses (look for unusual connections)

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=3 | stats count
by DestinationIp | sort - count
```



-> We have alot of results, so instead we take a step back now and analyse what we obtained in the previous question.
-> We know that rundll32.exe is running as an sacrifical process from the previous question, so let's test that with event code 3.
-> Hence, let's look at rundll32.exe with eventcode 3:

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=3
Image="*rundll32.exe" | stats count by DestinationIp
```



-> We have destination IP's 10.0.0.91 and 10.0.0.186 and is precisely what we want.

- Navigate to http://[Target IP]:8000, open the "Search & Reporting" application, and find through SPL searches against all data the port that one of the two C2 callback server IPs used to connect to one of the compromised machines. Enter it as your answer.

-> We look for the ports that the c2 server used to connected to the compromised host (destination host)

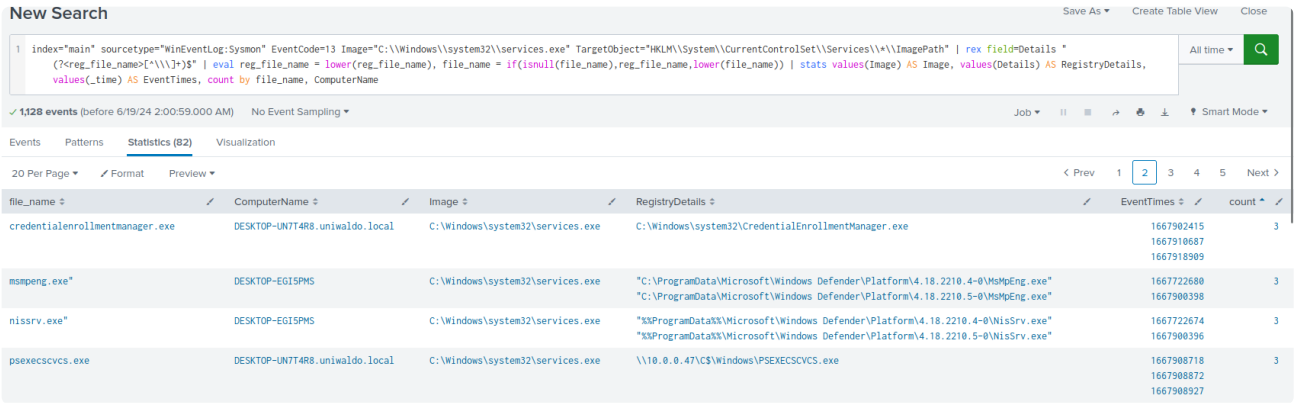
```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=3
(SourceIp=10.0.0.186 OR SourceIp=10.0.0.91) | stats count by
```

```
DestinationIp, DestinationPort
```

-> We see that it connects to 3389, the rdp port.

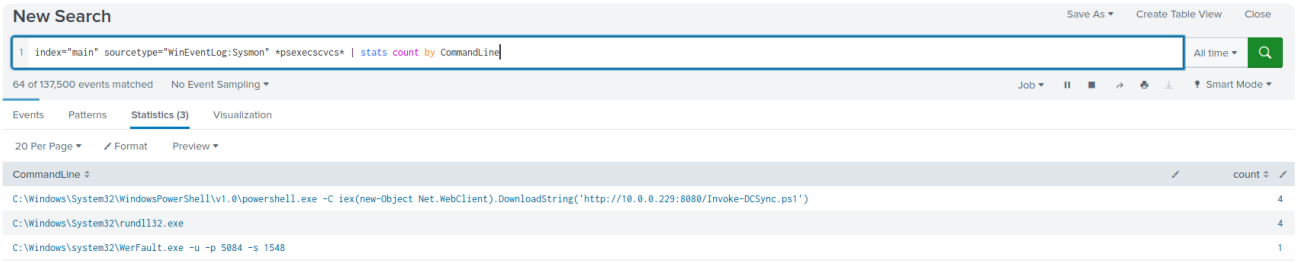
- Navigate to http://[Target IP]:8000, open the "Search & Reporting" application, and find through SPL searches against all data the password utilized during the PsExec activity. Enter it as your answer.
  ->We first look for psexec activities, first leveraging Sysmon Event ID 13, using the spl queries as follows:

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=13
Image="C:\\Windows\\system32\\services.exe"
TargetObject="HKLM\\System\\CurrentControlSet\\Services\\*\\ImagePath" |
rex field=Details "(?<reg_file_name>[^\\\]+)$" | eval reg_file_name =
lower(reg_file_name), file_name =
if(isnull(file_name),reg_file_name,lower(file_name)) | stats
values(Image) AS Image, values(Details) AS RegistryDetails,
values(_time) AS EventTimes, count by file_name, ComputerName
```



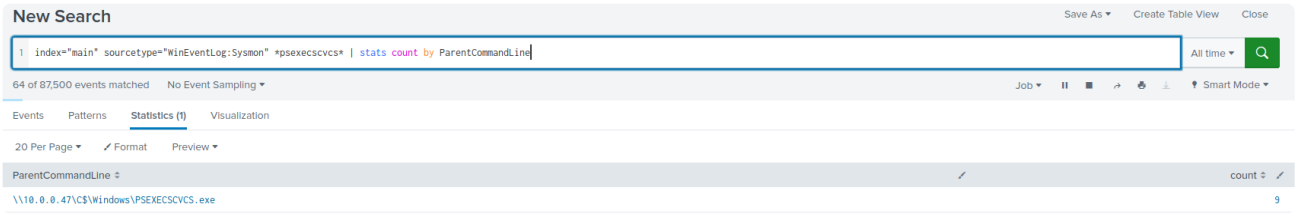
-> With some less frequent registry value set events, we see how there seems to be indications of resembling PsExec.
-> Hence, if we look into the command line:

```
index="main" sourcetype="WinEventLog:Sysmon" *psexecscvcs* | stats count
by CommandLine
```



-> We don't really see the password we want. So, we look for parent command line here:

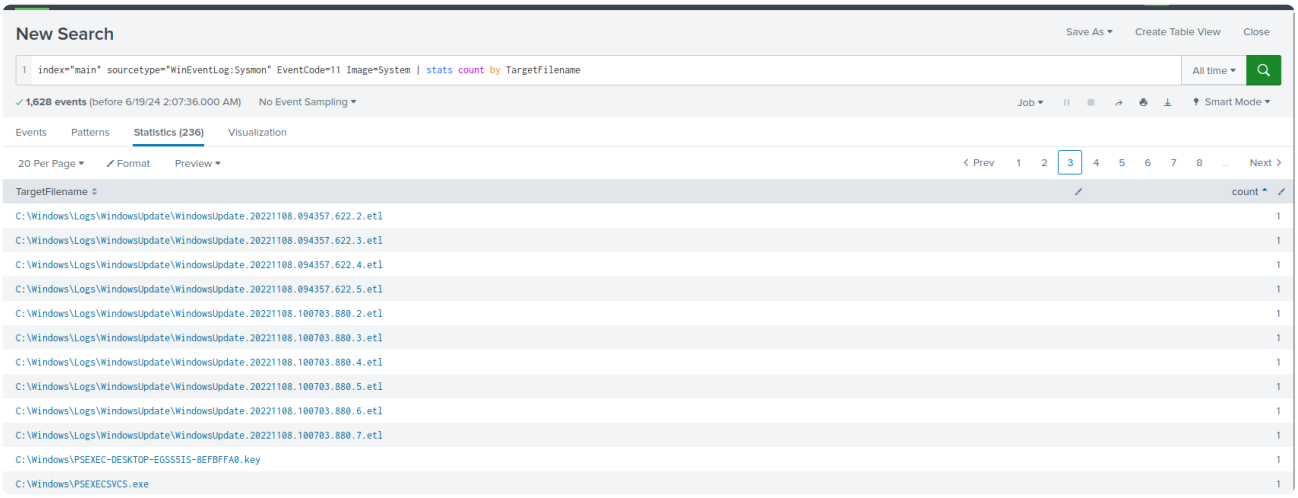
```
index="main" sourcetype="WinEventLog:Sysmon" *psexecscvcs* | stats count
by ParentCommandLine
```



-> Again, we don't see much, so we continue with our detection path. Maybe this is not the PsExec we are looking for.

-> Next, we look for leveraging Sysmon Event ID 11

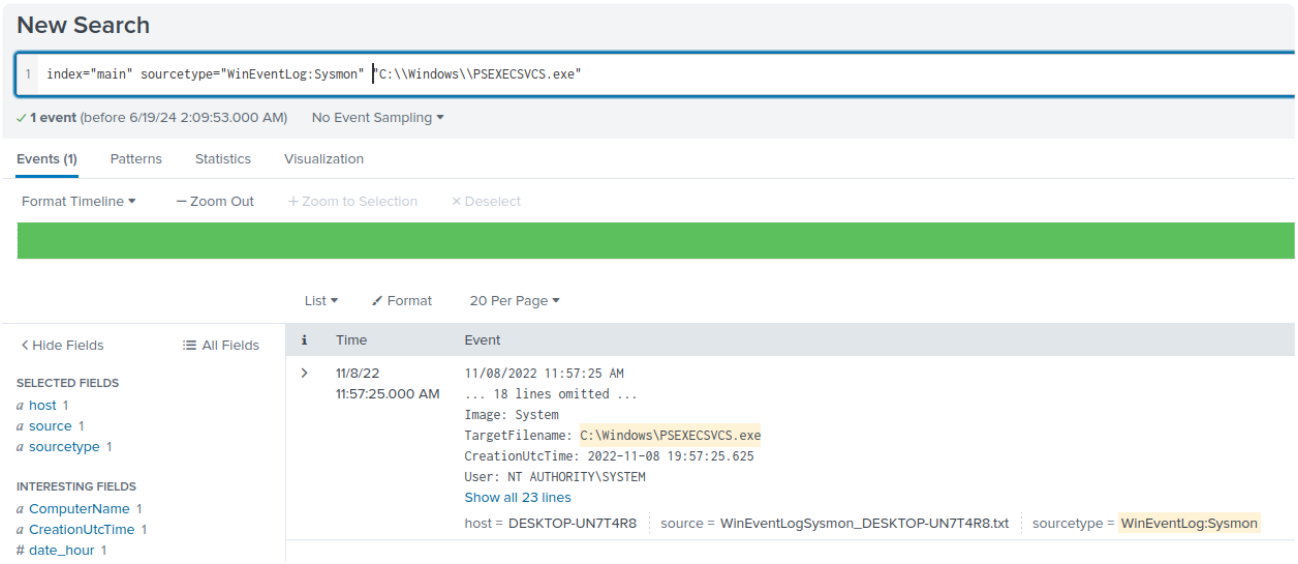
```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=11 Image=System |
stats count by TargetFilename
```



-> Again, we see the same thing, but the file we found is the same thing, that is, "PSEXECSVC.exe".

-> Now, looking into the event in an general manner, we see that:

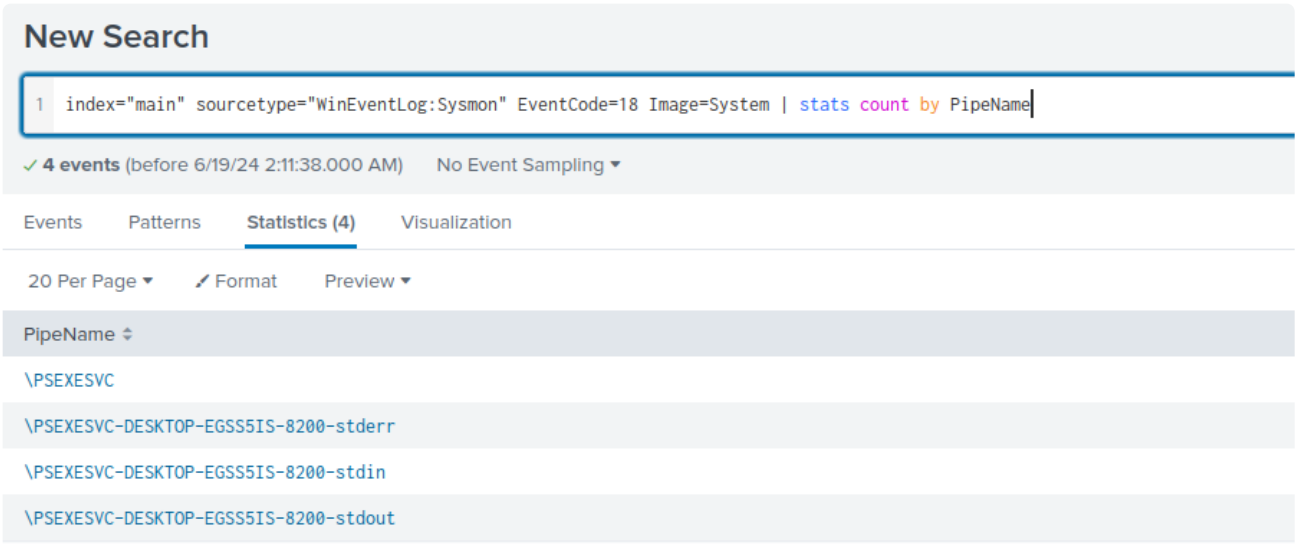
```
index="main" sourcetype="WinEventLog:Sysmon"
"C:\\Windows\\PSEXECSVCS.exe"
```



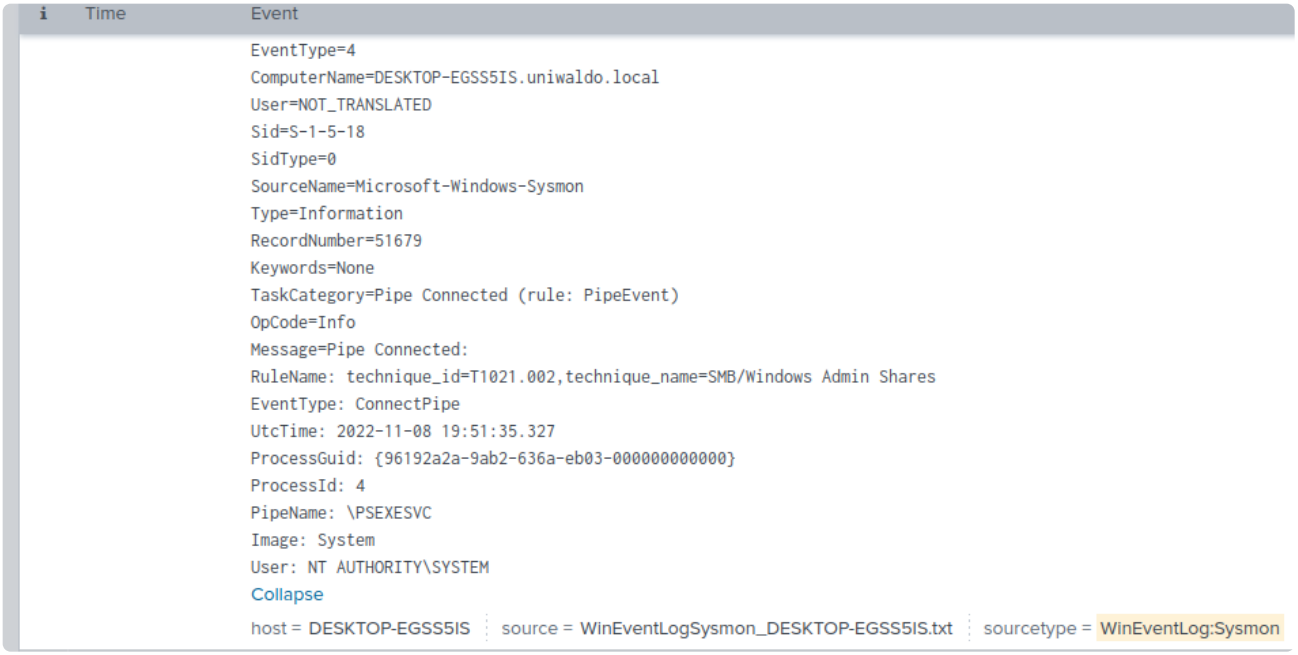
-> We didn't achieve much thing (see the password) and this is not the PsExec we are looking for.

-> Now, we look into the events for pipe connection event, leveraging Sysmon Event ID 18:

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=18 Image=System |
stats count by PipeName
```



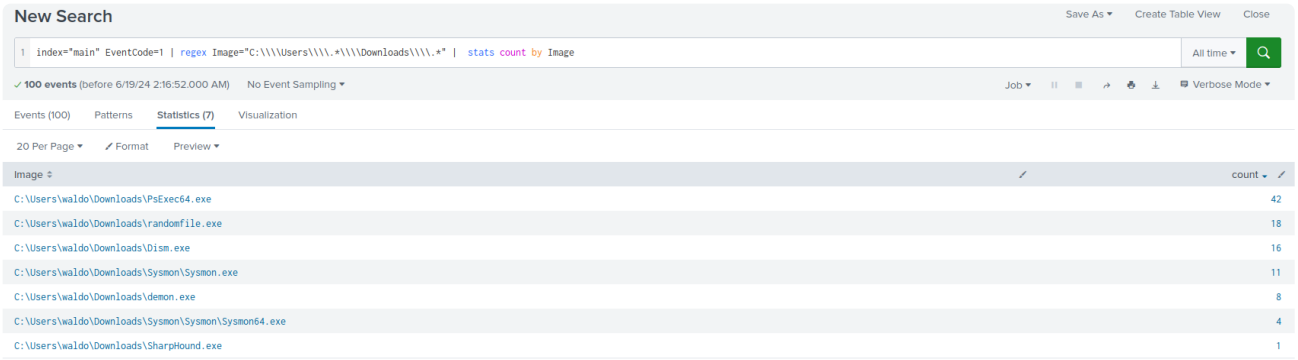
-> Again, we see some suspicious pipes being created which could resemble PsExec execution. However, we still haven't see any password yet.
-> Looking at the logs in verbose mode, we still don't capture any password related command lines.



-> Thus, we look for other techniques.

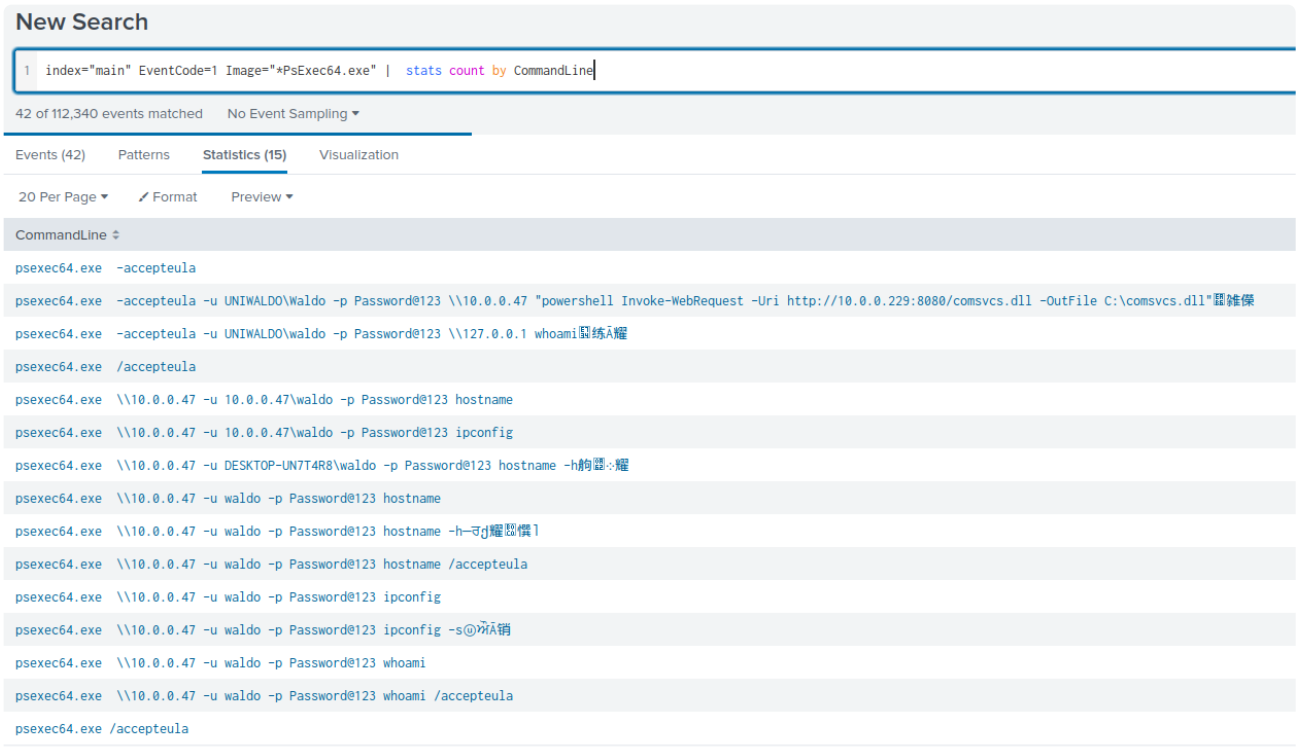
- Now, we look for suspicious process creation arising from the suspicious locations (e.g. User's download folder):

```
index="main" EventCode=1 | regex
Image="C:\\\\Users\\\\.*\\\\Downloads\\\\.*" |  stats count by Image
```



-> Here, we see suspicious files, like PsExec64.exe creating processes.
-> let's look at the command line that it is creating:

```
index="main" EventCode=1 Image="*PsExec64.exe" |  stats count by
CommandLine
```

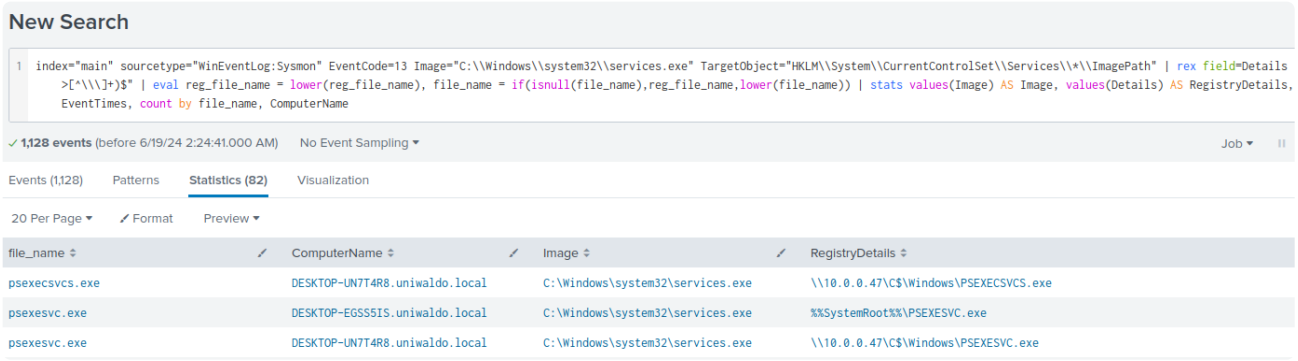


-> At last we found the PsExec execution we wanted, with the password of Password@123.

Alternative solution

-> We first run spl queries, leveraging Sysmon Event ID 13 (registry value set)

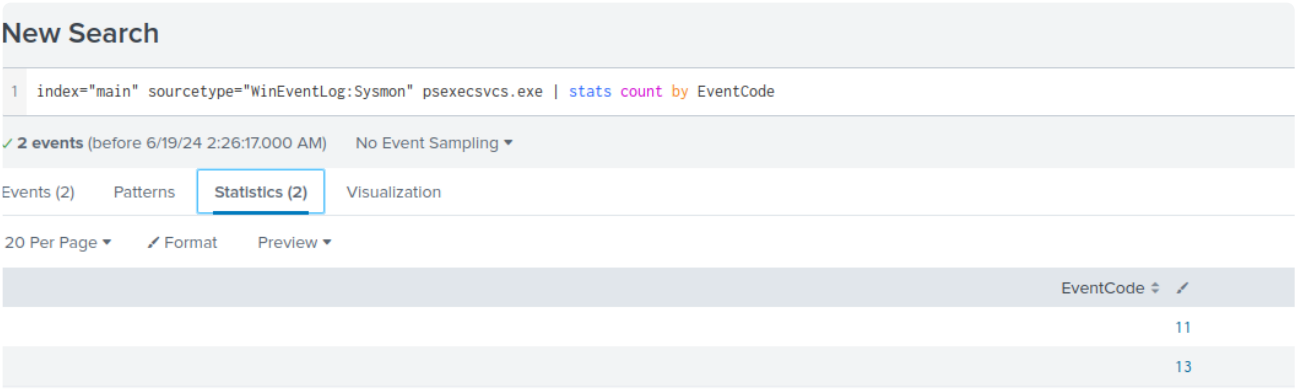
```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=13
Image="C:\\Windows\\system32\\services.exe"
TargetObject="HKLM\\System\\CurrentControlSet\\Services\\*\\ImagePath" |
rex field=Details "(?<reg_file_name>[^\\\]+)$" | eval reg_file_name =
lower(reg_file_name), file_name =
if(isnull(file_name),reg_file_name,lower(file_name)) | stats
values(Image) AS Image, values(Details) AS RegistryDetails,
values(_time) AS EventTimes, count by file_name, ComputerName
```



-> We see some unusual psexecsvc.exe binary.

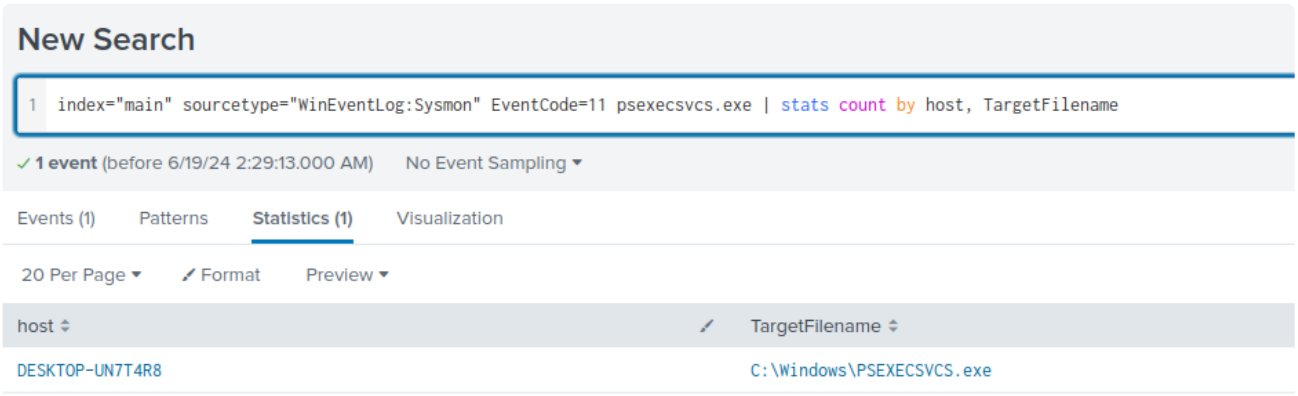
-> Next, we will filter out the binary by event code to see the behaviour of the binary

```
index="main" sourcetype="WinEventLog:Sysmon" psexecsvcs.exe | stats
count by EventCode
```



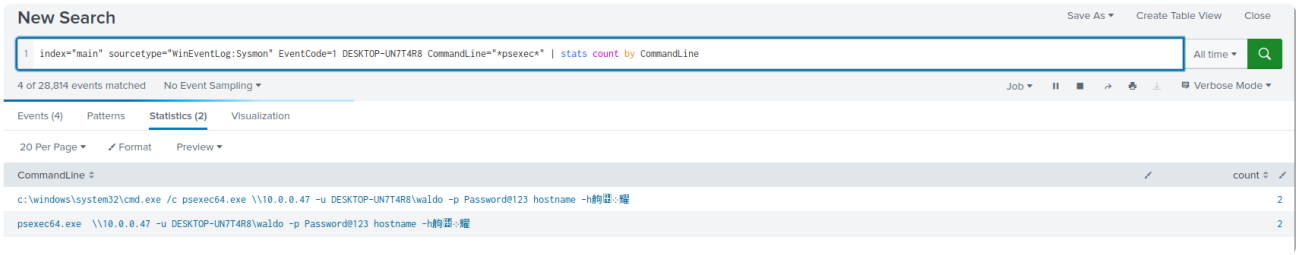
-> We see that it has an event code of 11, so it is created somewhere, let's look at the host that it is created through examining the host and TargetFilename

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=11 psexecsvcs.exe
| stats count by host, TargetFilename
```



-> We see that it the filename is suspicious and the host is "DESKTOP-UN7T4R8"
-> Now, we can focus on process creation events on this host that executes psexec in the commandline with the corresponding hostname.
-> Note that if we don't have any result with the host name, that means the attacker may be using IP addresses instead, so we may remove the DESKTOP-UN7T4R8 and look at the command line accordingly.

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 DESKTOP-UN7T4R8
CommandLine="*psexec*" | stats count by CommandLine
```



-> We obtained the password we want from PsExec execution/
-> Furthermore, we could see where psexec is executed through the host by including host in the stats command:

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 DESKTOP-UN7T4R8
CommandLine="*psexec*" | stats count by CommandLine, host
```

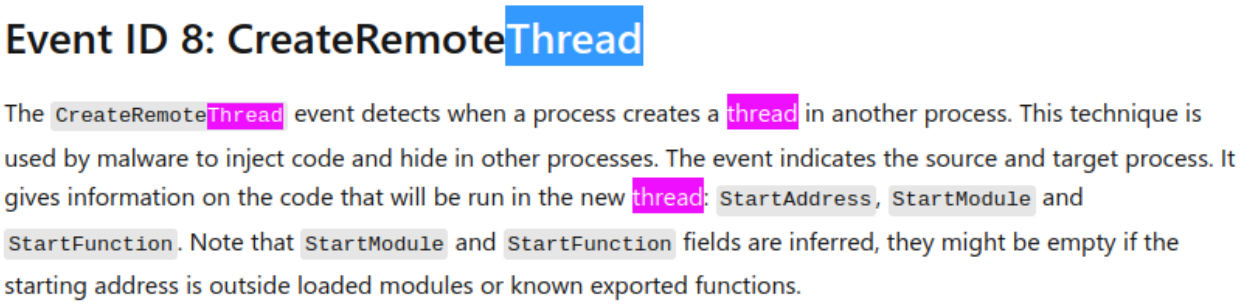
-> Hence, we see that psexec is running from the host DESKTOP-EGSS51S.
-> overall, it can be summarised that it is first identifying suspicious psexec binaries as well as the host being identified, then tracing back through the command line which host executed it.

Detecting Attacker Behavior With Splunk Based On Analytics

Question

- Navigate to http://[Target IP]:8000, open the "Search & Reporting" application, and find through an analytics-driven SPL search against all data the source process images that are creating an unusually high number of threads in other processes. Enter the outlier process name as your answer where the number of injected threads is greater than two standard deviations above the average. Answer format: _.exe

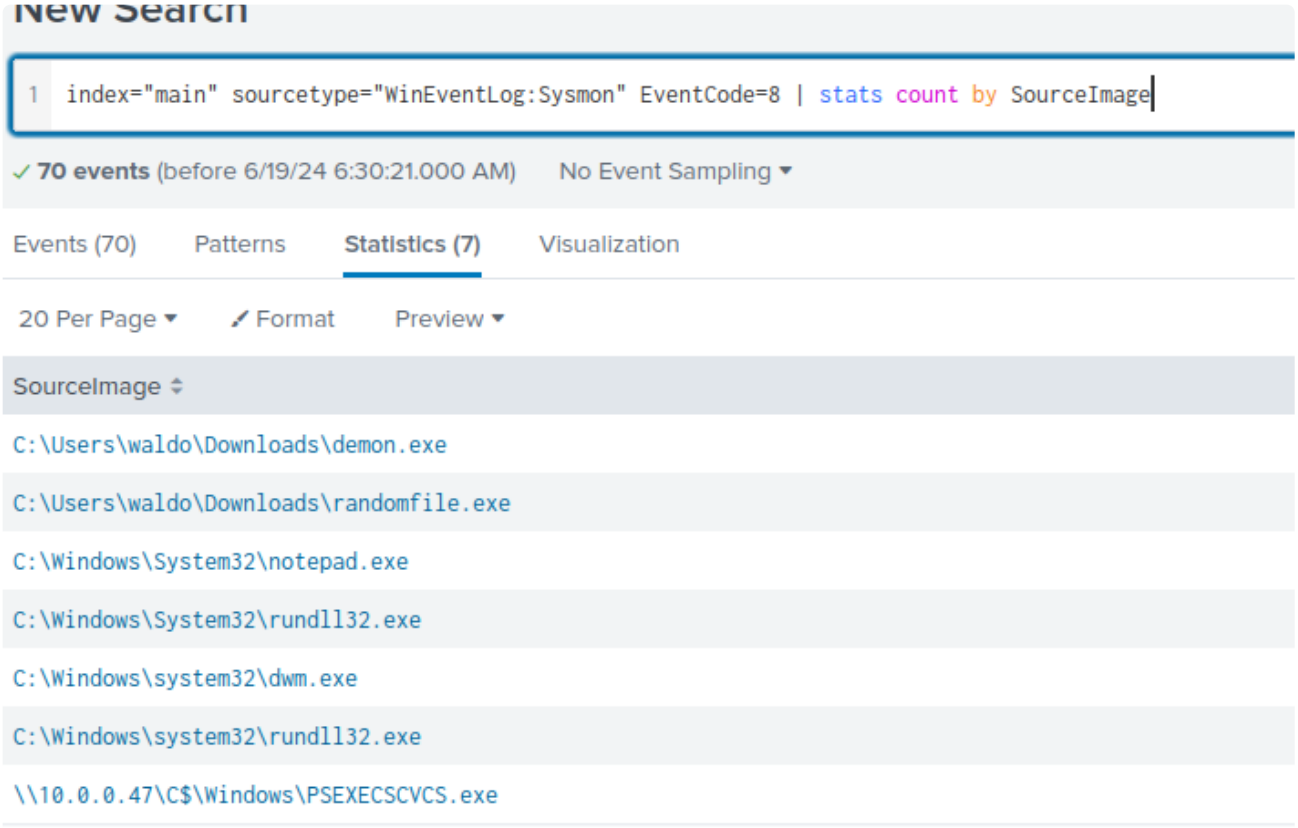
-> Looks for event that create threads



-> Event ID 8 is on the lookout.

-> Base examination:
-> We craft the spl query as follows (filter by event code 8,)

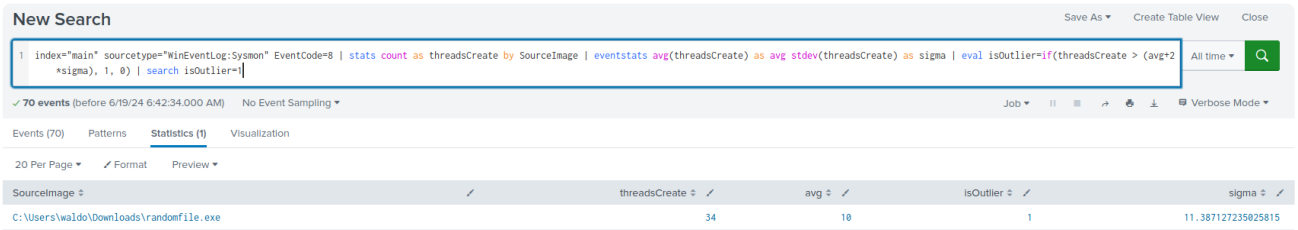
```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=8 | stats count
by SourceImage
```



-> We see some suspicious image, but let's formulate this in a more rigorous way.

- Now using eventstats, we craft the following spl queries (filter by event 8, create statistics on threads create, then evaluating if threads created is an anomaly).

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=8 | stats count as threadsCreate by SourceImage | eventstats avg(threadsCreate) as avg stdev(threadsCreate) as sigma | eval isOutlier=if(threadsCreate > (avg+2*sigma), 1, 0) | search isOutlier=1
```



-> Hence, the suspicious process is the randomfile.exe, if I remember correctly this is also the payload used for reverse shell.