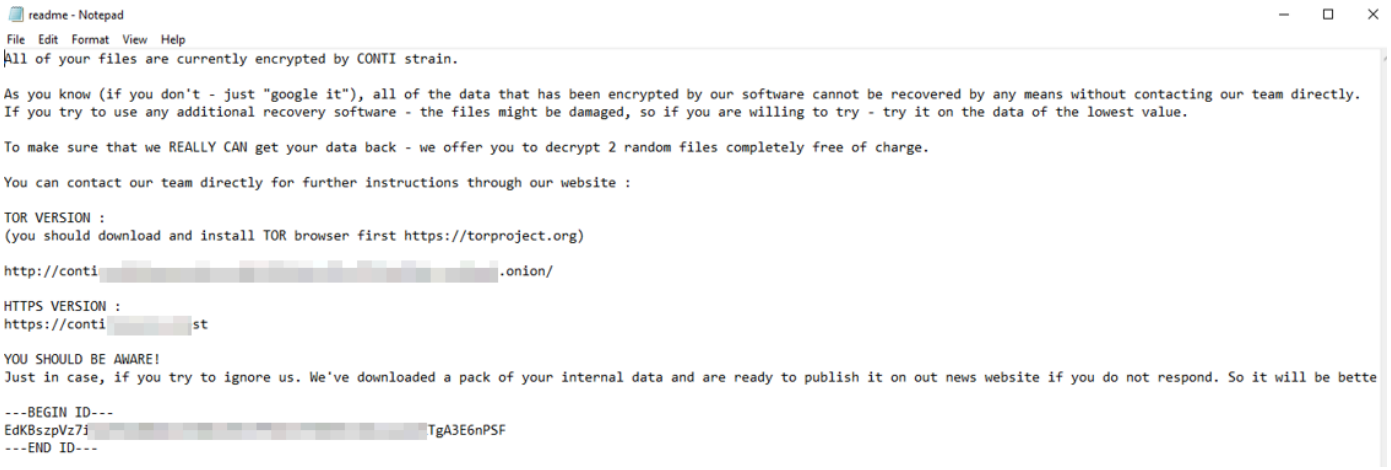
Conti

https://tryhackme.com/room/contiransomwarehgh

Some employees from your company reported that they can't log into Outlook. The Exchange system admin also reported that he can't log in to the Exchange Admin Center. After initial triage, they discovered some weird readme files settled on the Exchange server.

Below is a copy of the ransomware note.



Warning: Do **NOT** attempt to visit and/or interact with any URLs displayed in the ransom note.

Read the latest on the Conti ransomware here.

Connect to OpenVPN or use the AttackBox to access the attached Splunk instance.

Splunk Interface Credentials:

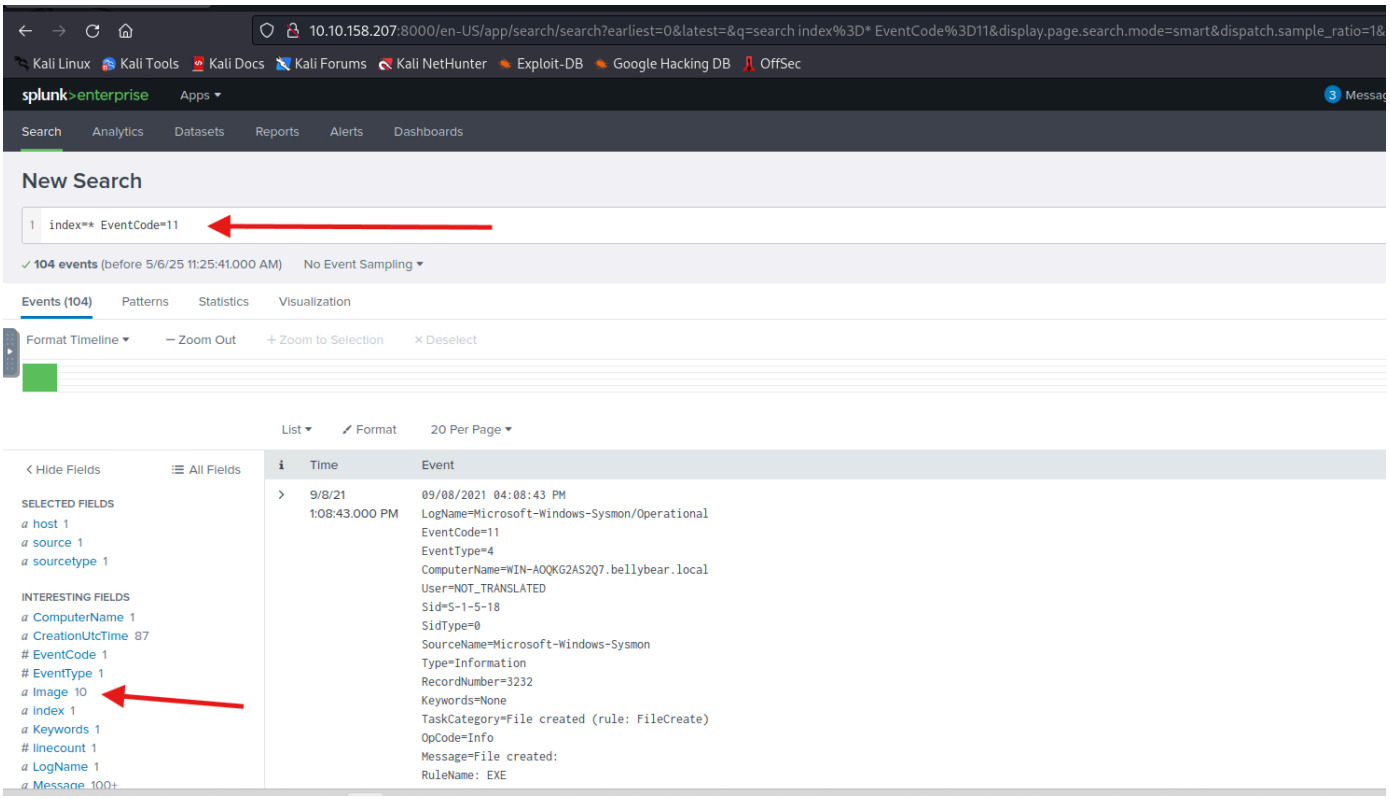
Username: `bellybear`

Password: `password!!!`

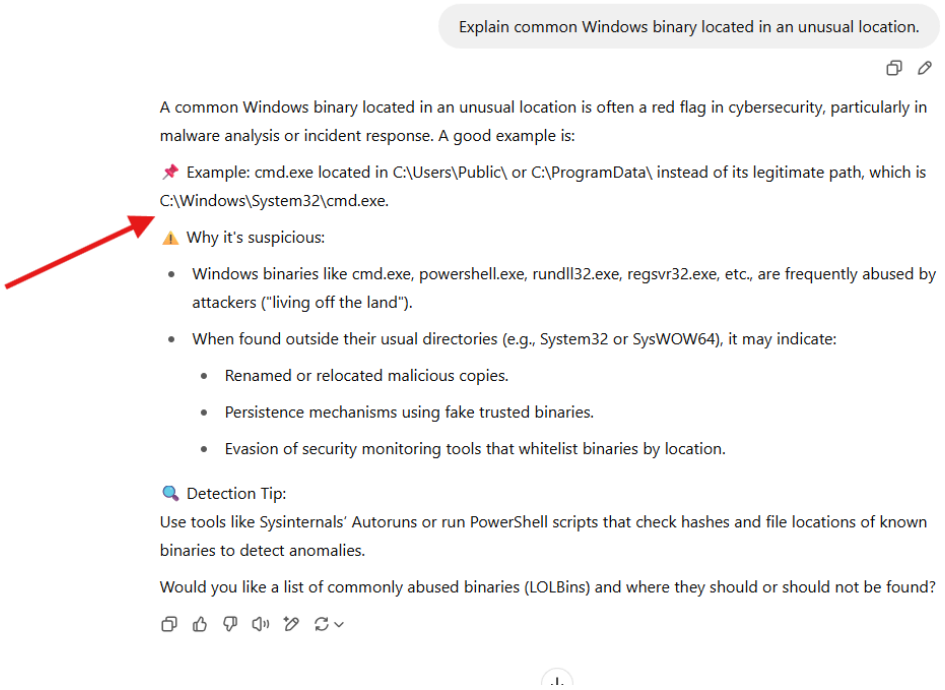
Splunk URL: `http://MACHINE_IP:8000`

Can you identify the location of the ransomware?

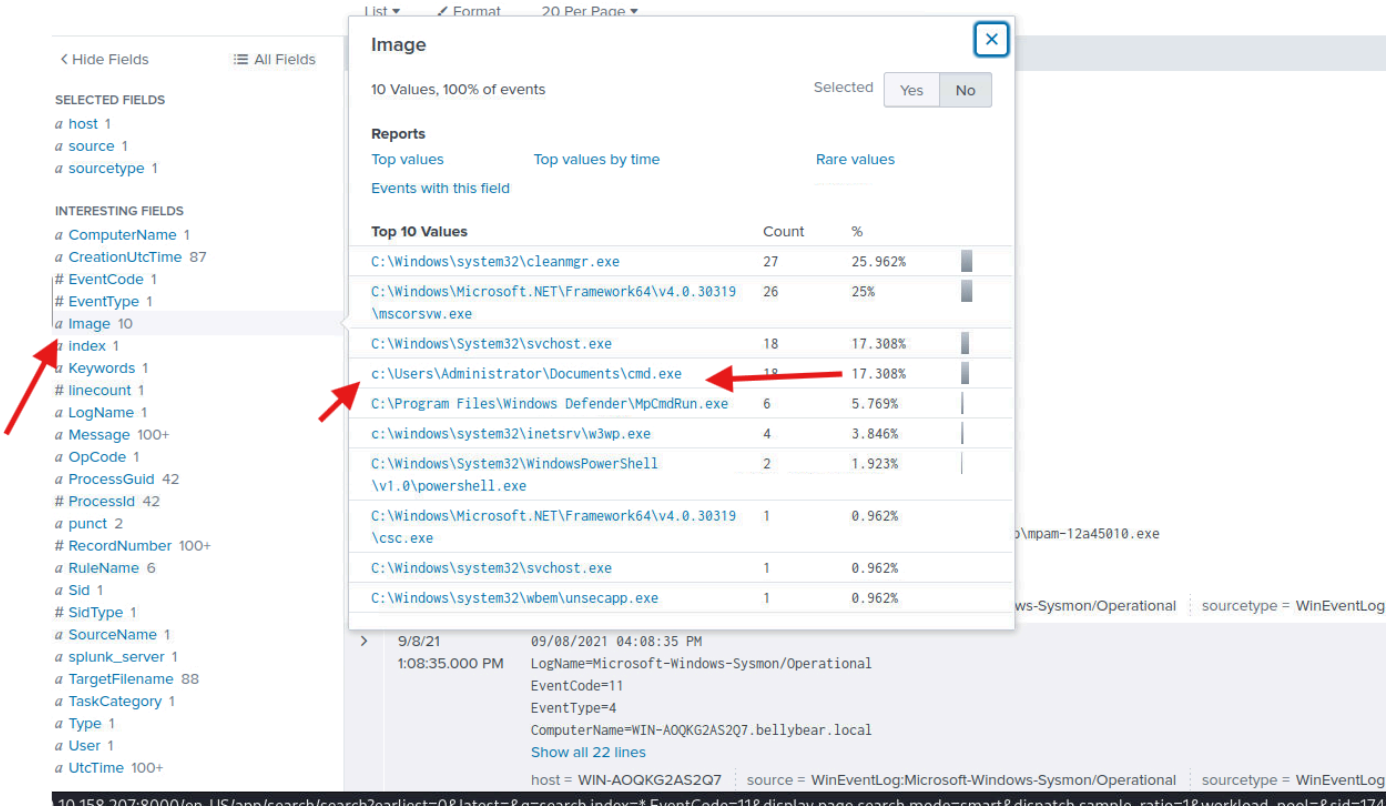
Check the Q2 and it mentioned that Sysmon event created and code is 11. write your query



for our search for the tip we got something



We need to check images if we can see anything.



In Splunk, the "Image" section often refers to the image path of a process or executable as recorded in event logs—particularly in Windows Event Logs or Sysmon logs. It's not about visual images, but rather executable file paths.

🧠 Here's what it means:

- **Image** = Full path to the executable file (.exe) that started a process or was involved in an event.
- This field is most common in:
  - Windows Security Logs (Event ID 4688: A new process has been created)
  - Sysmon Logs (Event ID 1: Process Create)
  - EDR or SIEM data sources

🔍 Example:
In a Sysmon event:

json

CopyEdit

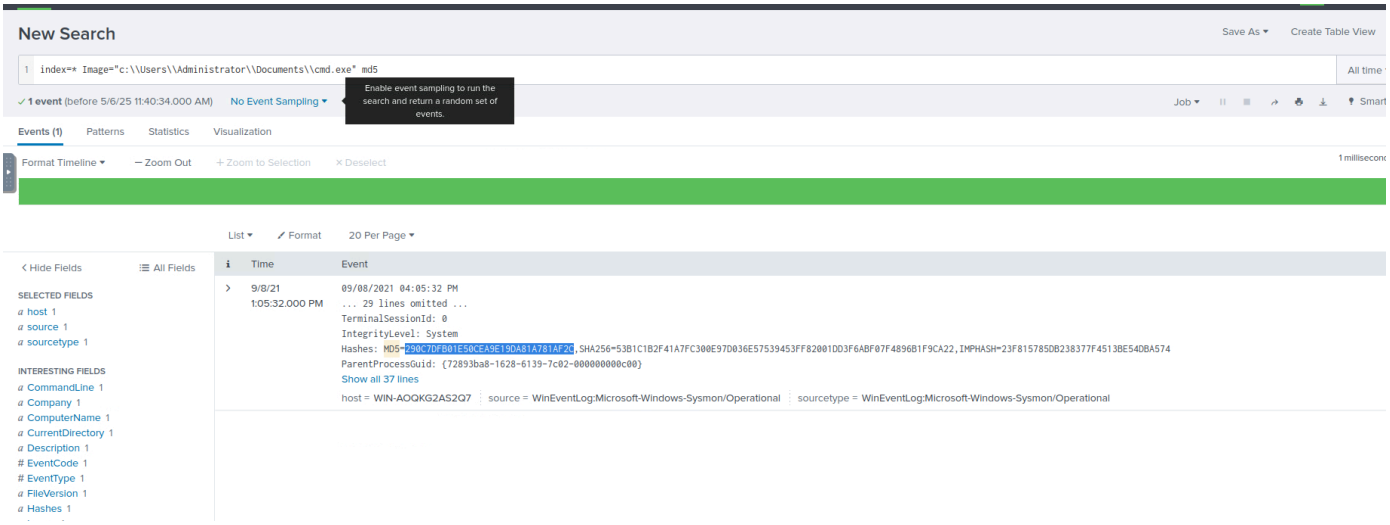
```
"Image": "C:\Windows\System32\cmd.exe"
```

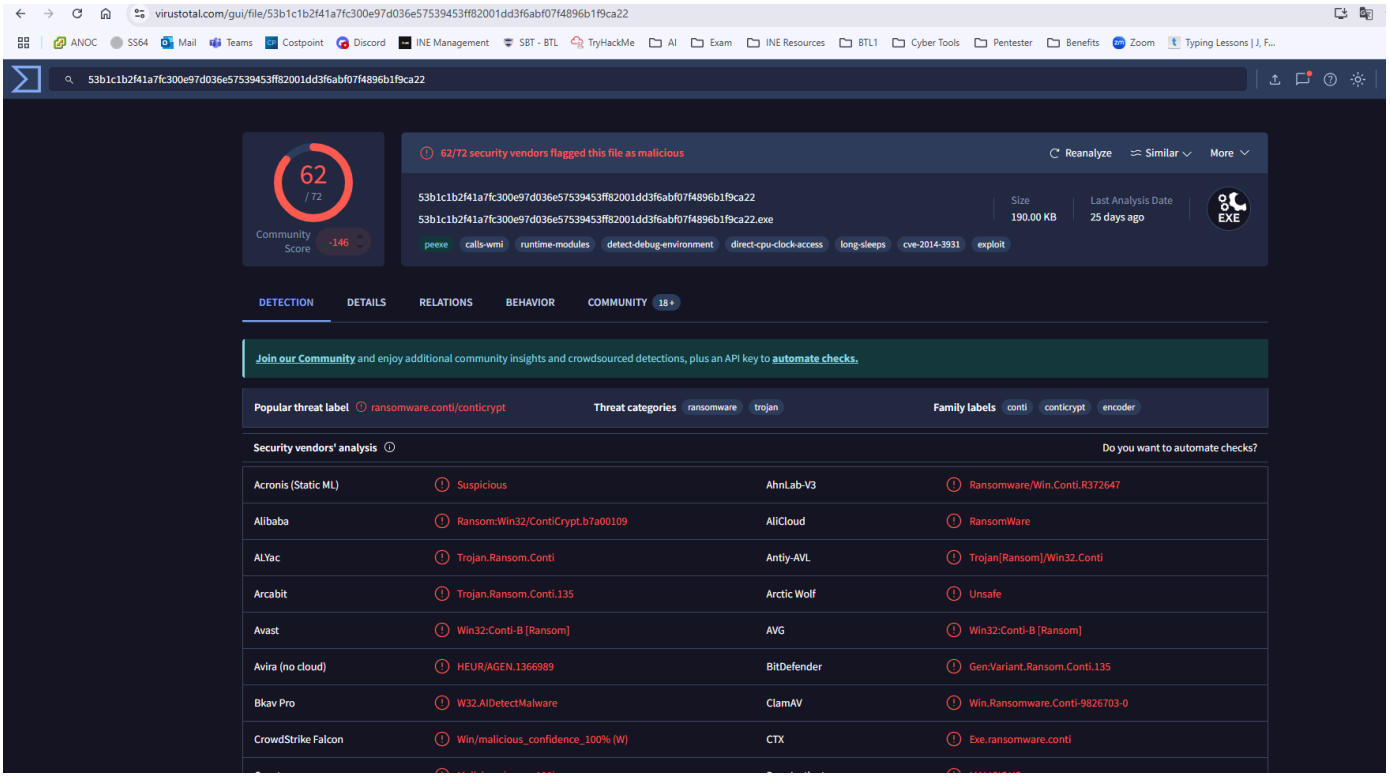
What is the Sysmon event ID for the related file creation event?

11

Can you find the MD5 hash of the ransomware?

Basicly remove the EcentCode =11 o and add md5 in the search bar





What file was saved to multiple folder locations?

“Image=”c:\\Users\\Administrator\\Documents\\cmd.exe” EventCode=11”, and after that click on TargetFileName on the interesting fields section, we can see below result.

In Splunk, the field TargetFileName typically refers to the full path and name of a file involved in a file-related event—most commonly from Windows logs, Sysmon (System Monitor), or endpoint detection tools.

Why it’s important in security:

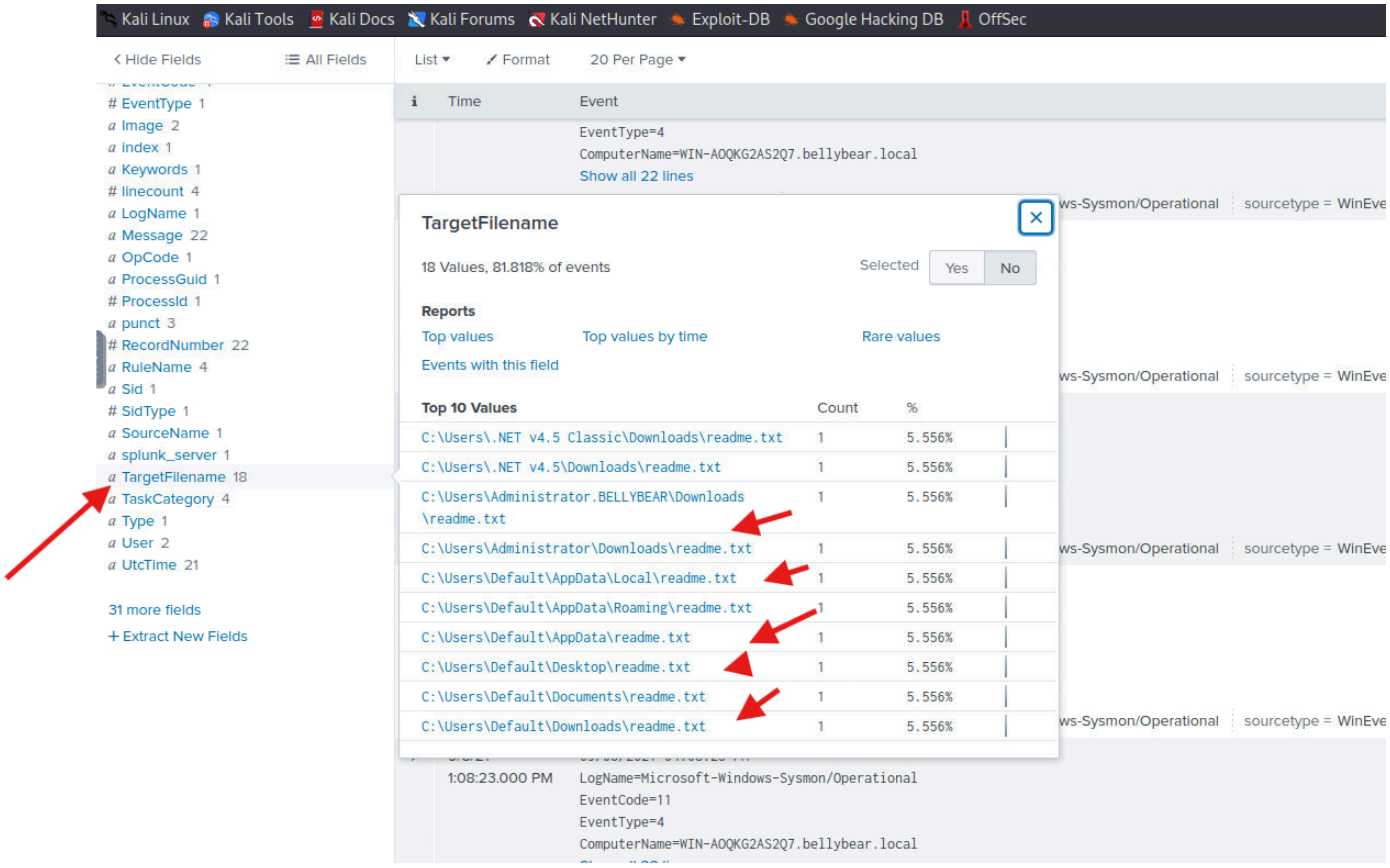
- Used to detect:
  - File creation in suspicious locations (e.g., TEMP, Downloads, ProgramData)
  - Known malware filenames or hashes
  - Unexpected file writes by sensitive processes

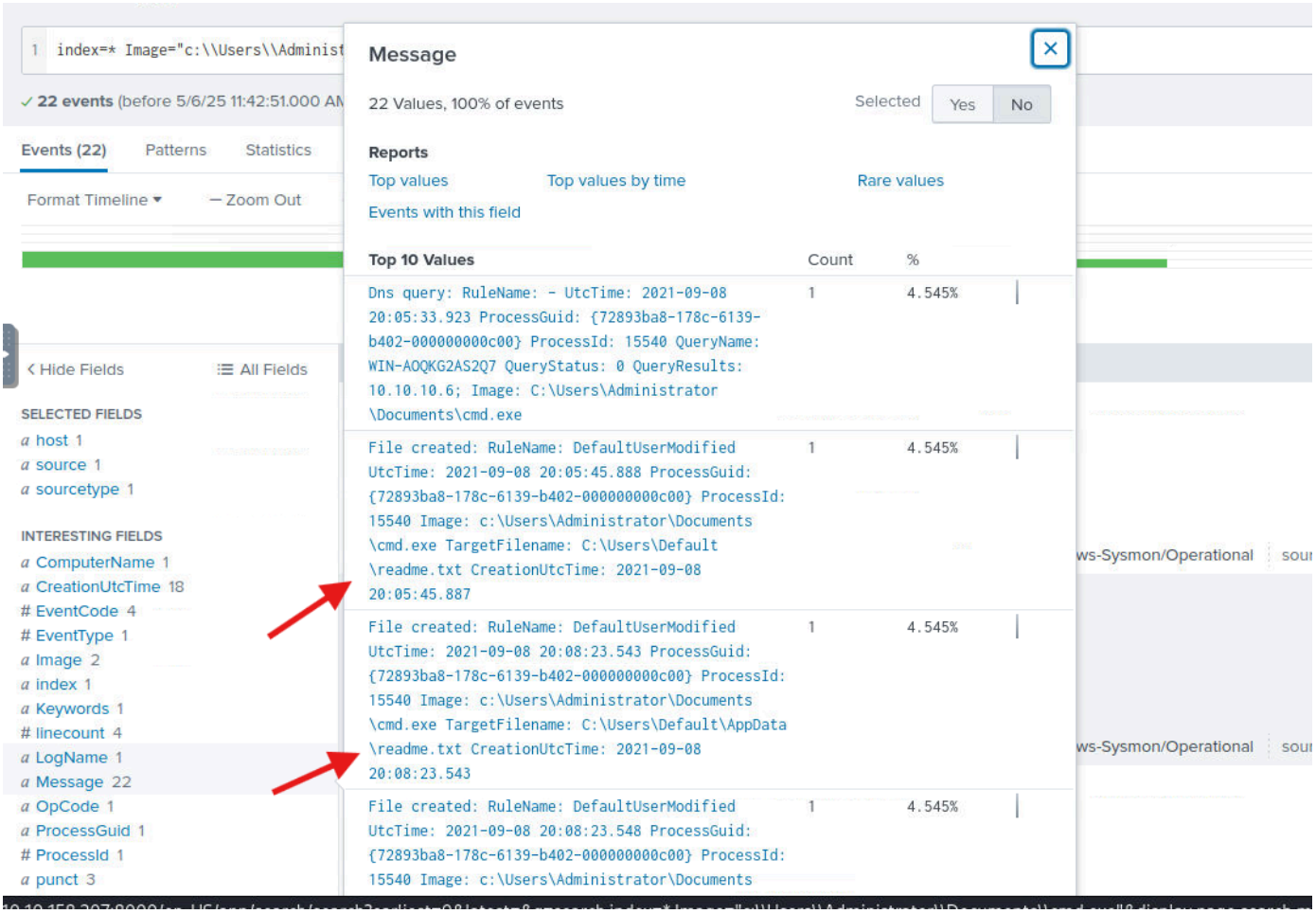
🛠️ Example Splunk search:

spl

CopyEdit

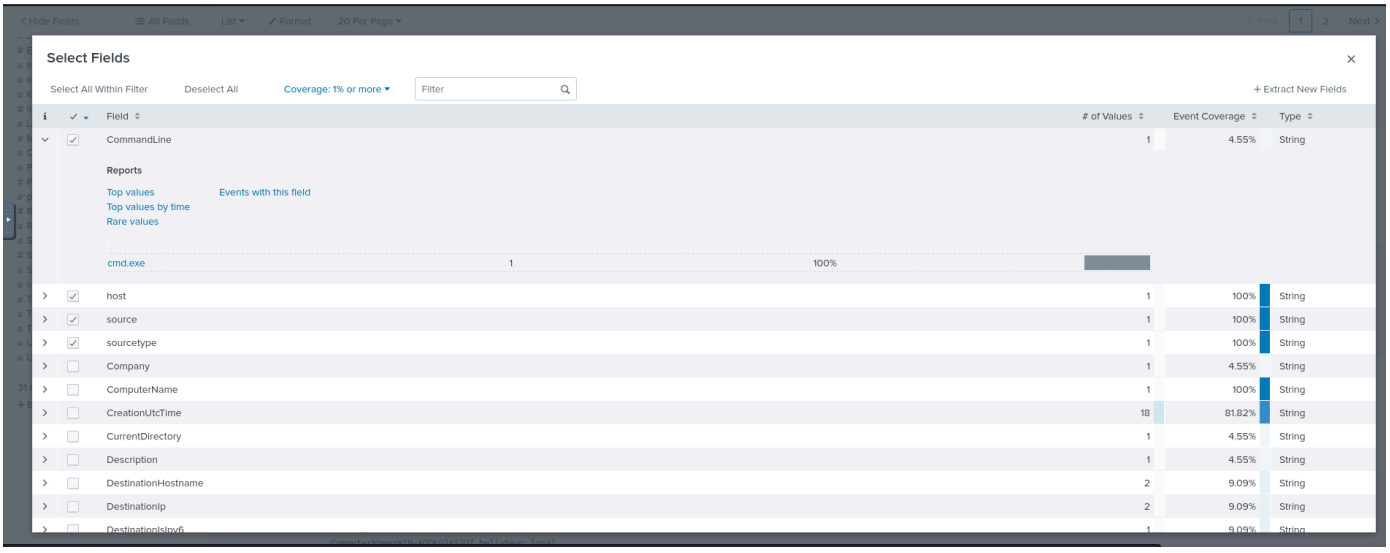
```
index=sysmon EventCode=11 TargetFileName="*\\Temp\\*"
```





What was the command the attacker used to add a new user to the compromised system?

Filter out using CommandLine and wildcard on add to check all commands containing *add*. Below is the result command.



CommandLine="*/add*"

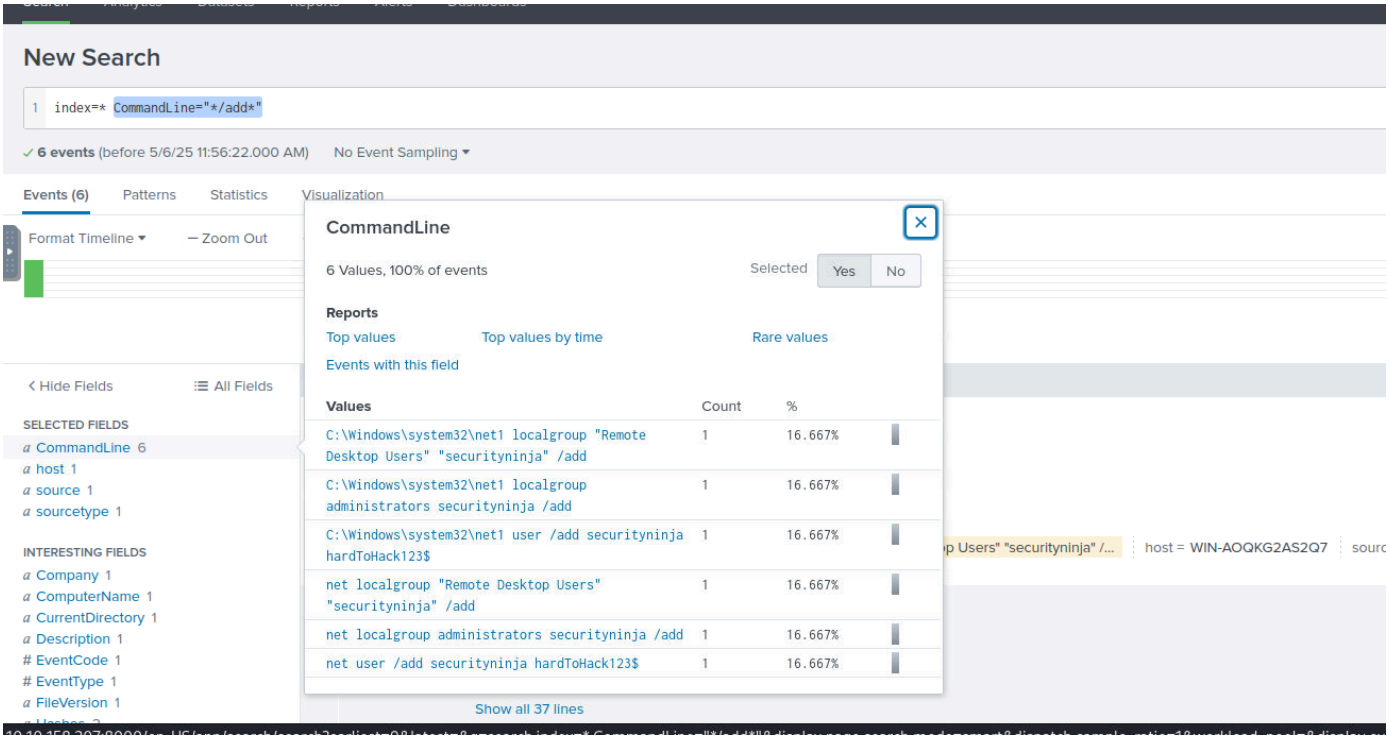
The query snippet CommandLine="*/add*" is filtering events in Splunk where the CommandLine field contains the string `/add`

"*/add*":

- The asterisks (*) act as wildcards in Splunk (they represent "zero or more characters").
- So this means: **the CommandLine must contain the string `/add` anywhere in it**.

Why this is often monitored:

- The `/add` switch is commonly used with the `net` command to add user accounts or groups.
- This pattern is often monitored in detection rules to catch potential privilege escalation or unauthorized account creation.



net user /add securityninja hardToHack123$
✅ This is the most direct command that:

Adds a new user account named securityninja

Sets the password to hardToHack123$

The other commands extend privileges or add the user to groups, but this one is the initial creation of the user account.

Breakdown of Each Command:

1. ✅ **net user /add securityninja hardToHack123$**

- **Purpose**: Create a new local user account named securityninja with the password hardToHack123$
- **Explanation**:
  This is the core action that introduces the new user into the system.

- **Impact**: Account now exists locally on the system.

2. 🛠 net localgroup administrators securityninja /add

- **Purpose**: Add the new user to the local Administrators group.
- **Explanation**:
  Grants the user full administrative privileges.
- **Impact**: The attacker now has full control over the system using this account.

3. 🌐 net localgroup "Remote Desktop Users" "securityninja" /add

- **Purpose**: Allow the user to connect to the system via Remote Desktop Protocol (RDP).
- **Explanation**:
  This is required if the attacker wants to log in graphically via RDP.
- **Impact**: Enables remote access with GUI.

4. 💉 C:\Windows\system32\net1 user /add securityninja hardToHack123$

- **Purpose**: Same as Command #1 but using net1 instead of net.
- **Explanation**:
  net1 is a backup or alternative binary to net, sometimes used to bypass basic monitoring that watches net.exe.
- **Impact**: Redundant but useful for stealth.

5. 🛠 C:\Windows\system32\net1 localgroup administrators securityninja /add

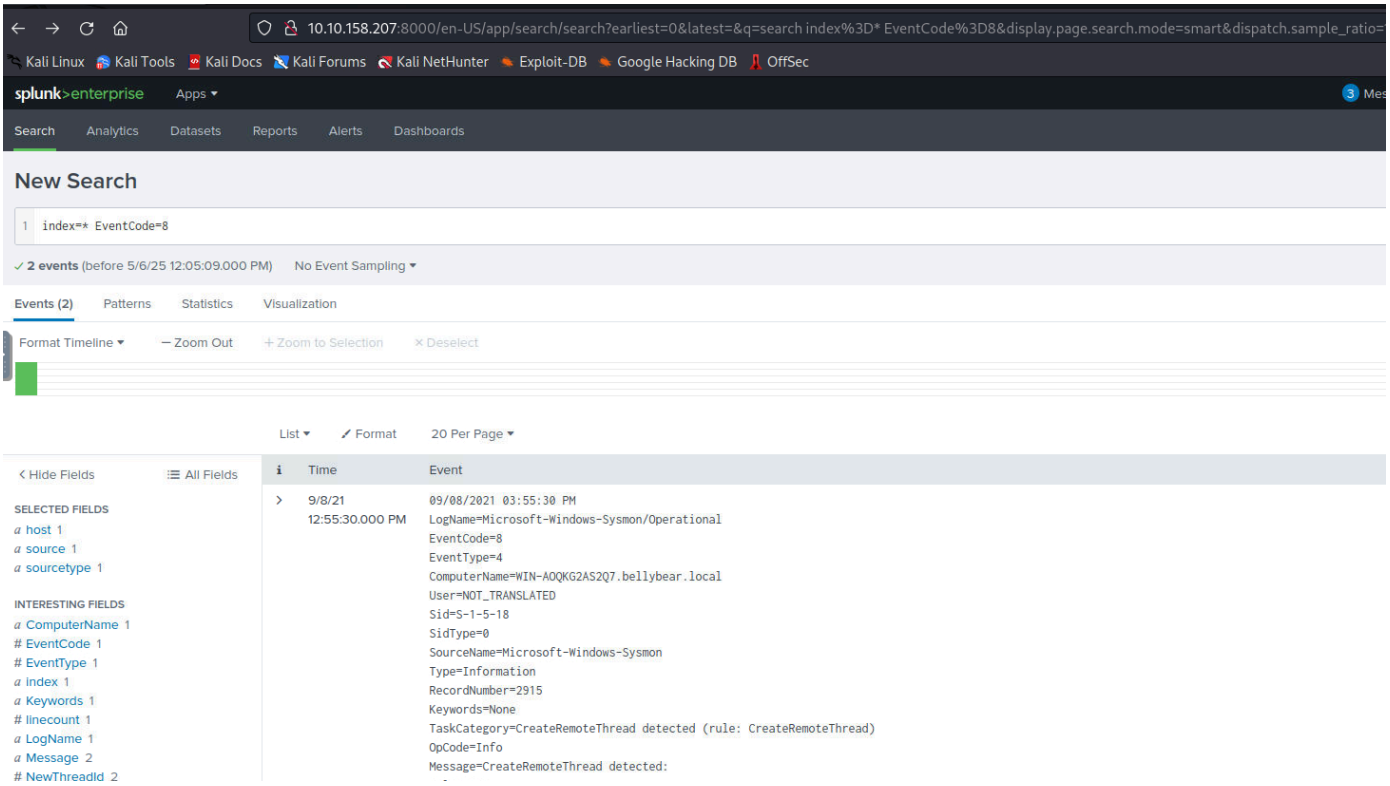
- **Purpose**: Same as Command #2, but using net1.
- **Explanation**:
  Also for stealth or if net.exe is being monitored or restricted.

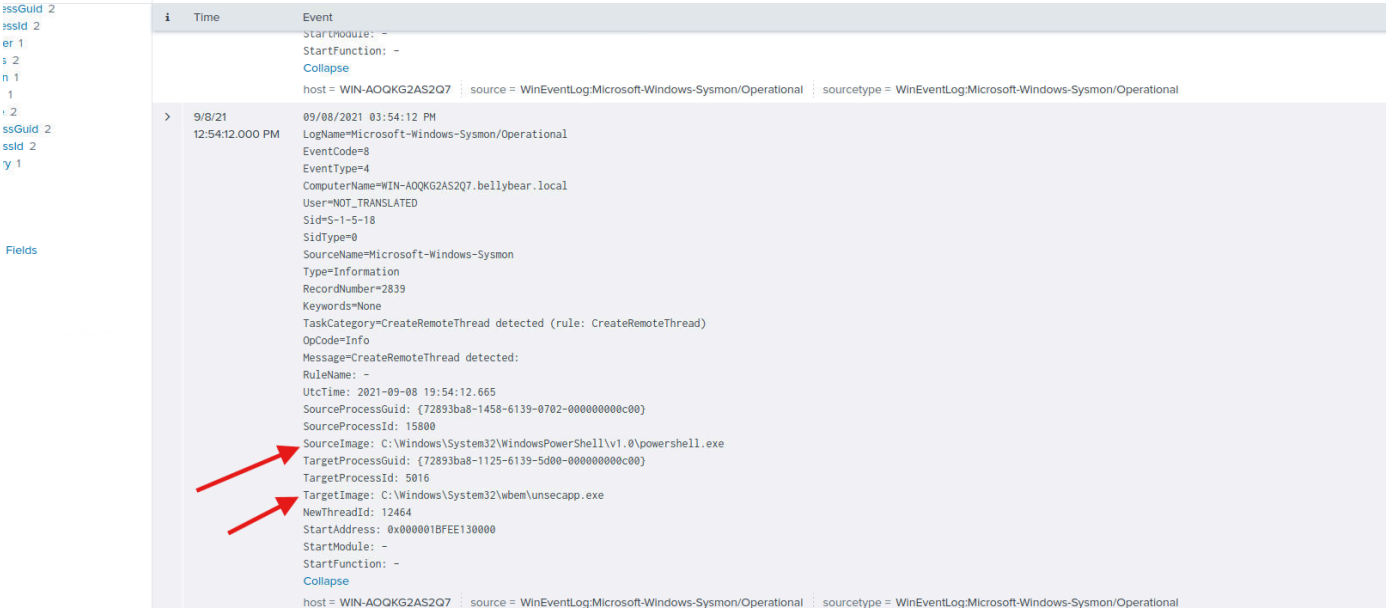
6. 🌐 C:\Windows\system32\net1 localgroup "Remote Desktop Users" "securityninja" /add

- **Purpose**: Same as Command #3, but using net1.
- **Explanation**:
  Again, an alternate way to ensure remote access is granted.

The attacker migrated the process for better persistence. What is the migrated process image (executable), and what is the original process image (executable) when the attacker got on the system?

Add EvendCode =8 in search bar





- **Source Image (Original Process)**: `C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe`
- **Target Image (Migrated Process)**: `C:\Windows\System32\wbem\unsecapp.exe`
- **Details**:
  - **Original Process**: `powershell.exe` is being used by the attacker, which is common in exploitation or post-exploitation activities. Attackers frequently use PowerShell for scripting malicious actions.
  - **Migrated Process**: `unsecapp.exe` is a legitimate Windows process involved in WMI (Windows Management Instrumentation). However, in this case, it is being used by the attacker to move

their malicious code into a more stealthy process that is less likely to be flagged.

The key here is that the attacker migrated from `powershell.exe` to `unsecapp.exe`, and this is typically done for **better persistence** and to evade detection. `powershell.exe` is often closely monitored, and by injecting code into `unsecapp.exe`, the attacker can keep their activities hidden behind a legitimate process.

Answer:

- **Migrated Process Image (Executable)**: `C:\Windows\System32\wbem\unsecapp.exe` (Event 2)
- **Original Process Image (Executable)**: `C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe` (Event 2)

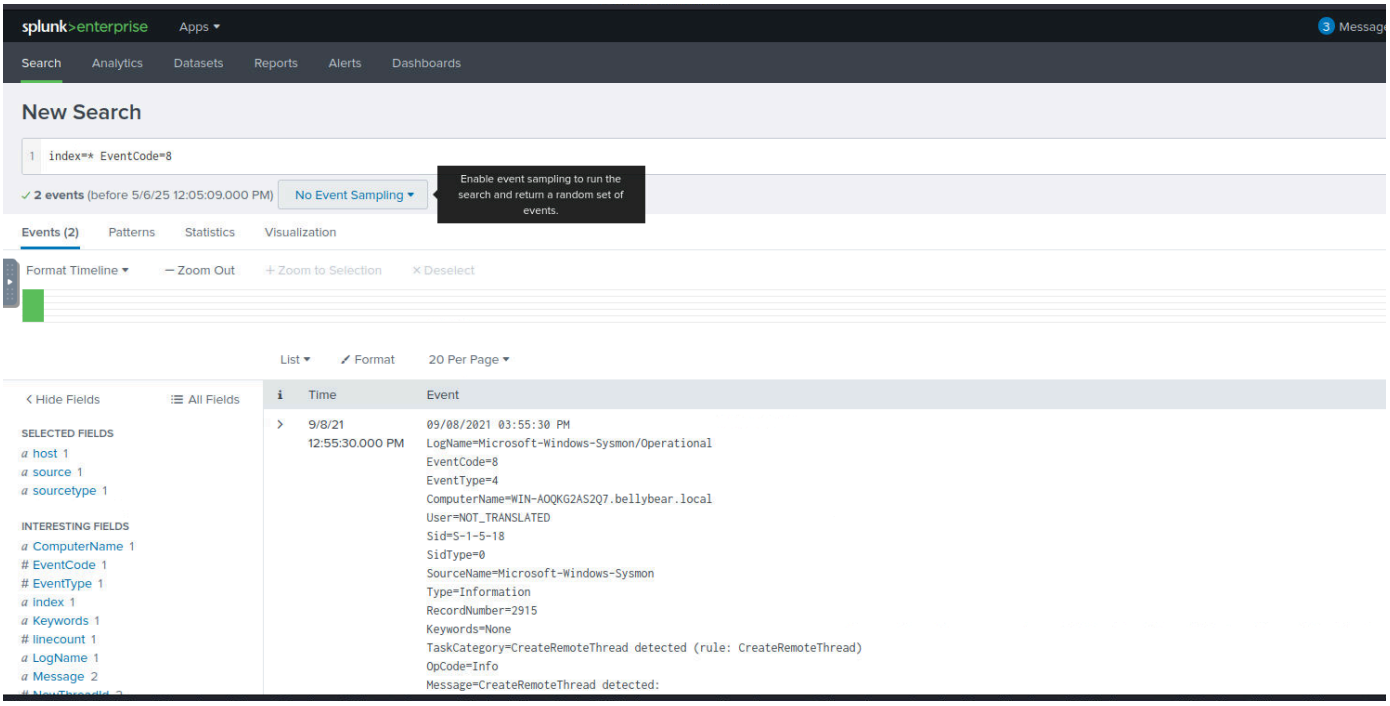
Why This Event is Key:

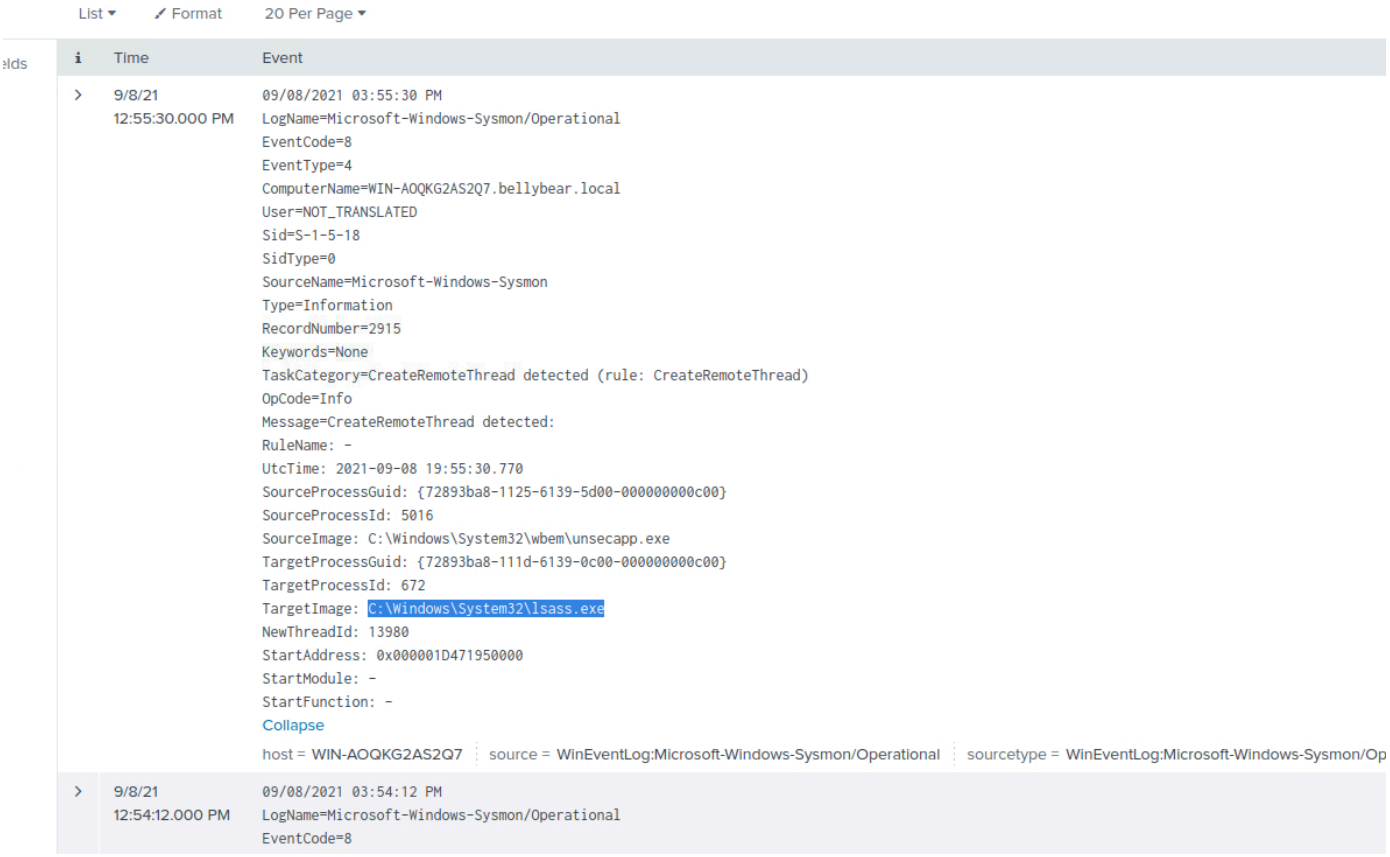
- **Persistence and Stealth**: The attacker migrated from a high-visibility process (`powershell.exe`) to a less suspicious one (`unsecapp.exe`). This is often done to maintain control while avoiding detection by security tools that might flag the more obvious PowerShell activity.

This is the correct event because the **process migration** to `unsecapp.exe` reflects the attacker's intent to use a legitimate system process for persistence, while reducing the likelihood of detection.

The attacker also retrieved the system hashes. What is the process image used for getting the system hashes?

Using the EventCode=8 filter alone, check the TargetImage and it will show two events, if we refer to the output from the search used in question 6, we can see that a second process migration takes place between unsecapp.exe and lsass.exe





What is the web shell the exploit deployed to the system?

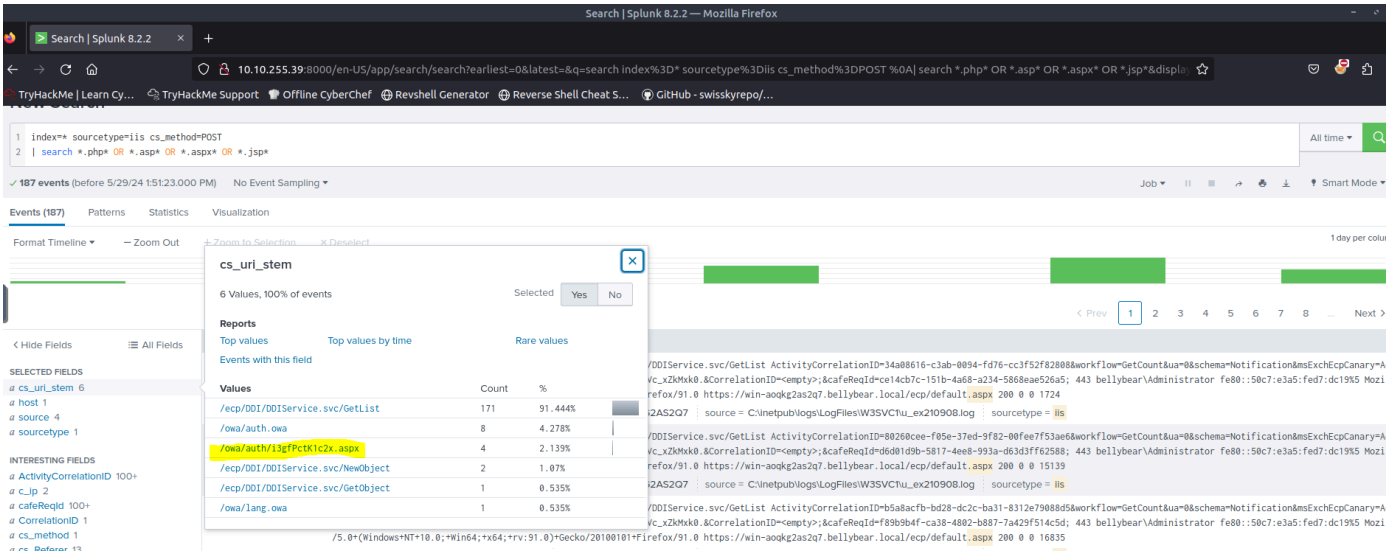
To answer this question, I started by changing my SourceType from Sysmon to IIS events, since it collects events related to web pages. Next, I filtered IIS events for POST requests and common web shell file types (.php, .asp, .aspx, .jsp):

common web shell file types?

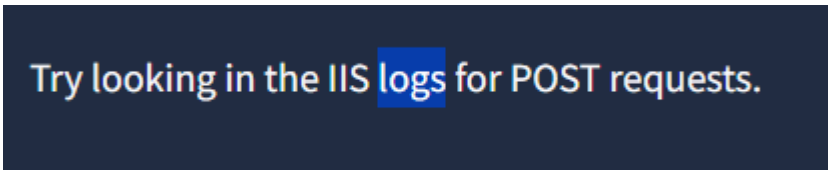
Web shells are malicious scripts that attackers upload to a web server to gain remote control and execute arbitrary commands. They are often disguised as legitimate files to avoid detection. Here are some common file types and extensions used for web shells:

Common Web Shell File Types

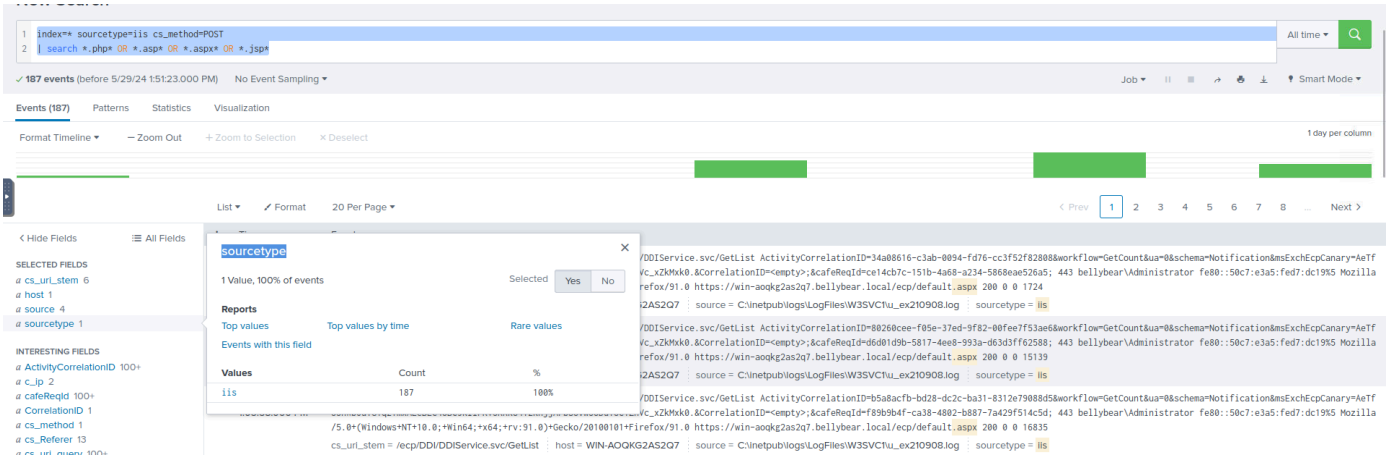
1. PHP Files
   - Extensions: `.php`, `.php5`, `.phtml`
   - Example Web Shells: `c99.php`, `r57.php`, `b374k.php`
2. ASP and ASP.NET Files
   - Extensions: `.asp`, `.aspx`, `.ashx`
   - Example Web Shells: `cmdasp.asp`, `aspxspy.aspx`
3. JSP (Java Server Pages) Files
   - Extensions: `.jsp`, `.jspx`
   - Example Web Shells: `cmd.jsp`, `webshell.jsp`
4. CFM (ColdFusion Markup) Files
   - Extensions: `.cfm`, `.cfml`



index=* sourcetype=iis cs_method=POST
| search *.php* OR *.asp* OR *.aspx* OR *.jsp*

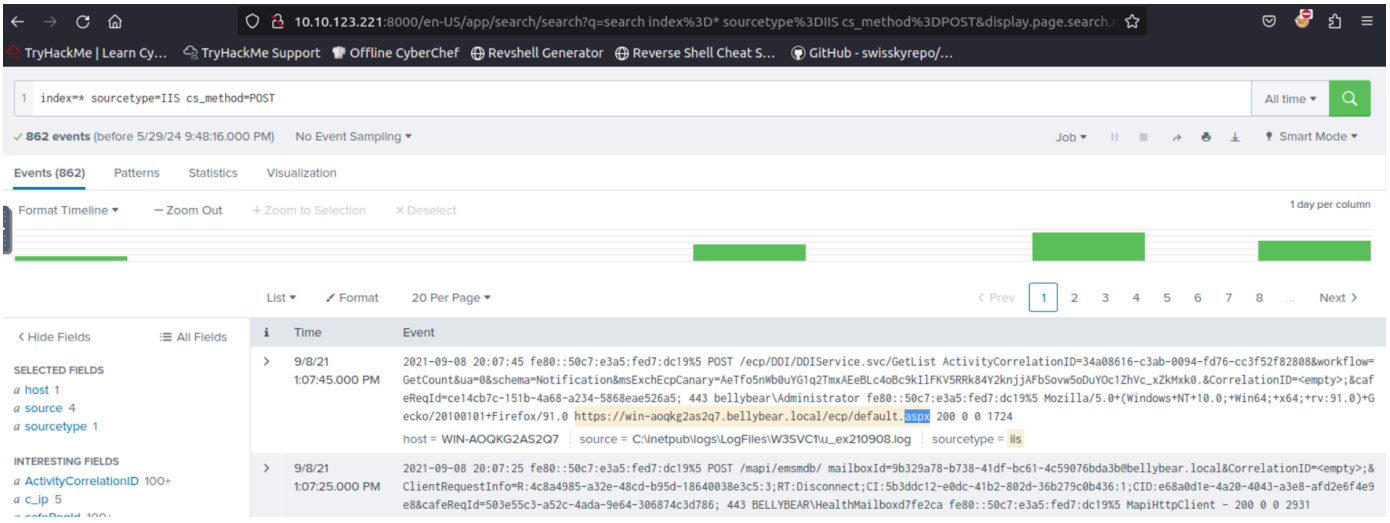


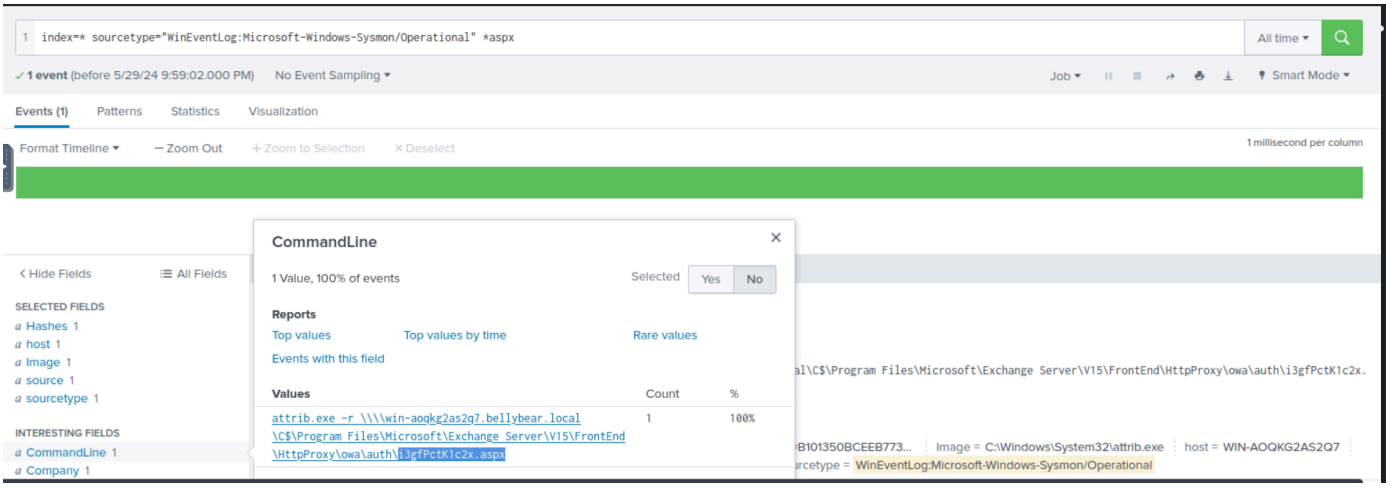
Try looking in the IIS **logs** for POST requests. Logs mean use sourcetype=



index=* sourcetype=iis cs_method=POST | search *.php* OR *.asp* OR *.aspx* OR *.jsp* sourcetype=iis

The term "cs_method" means "Client to Server Method" in English.

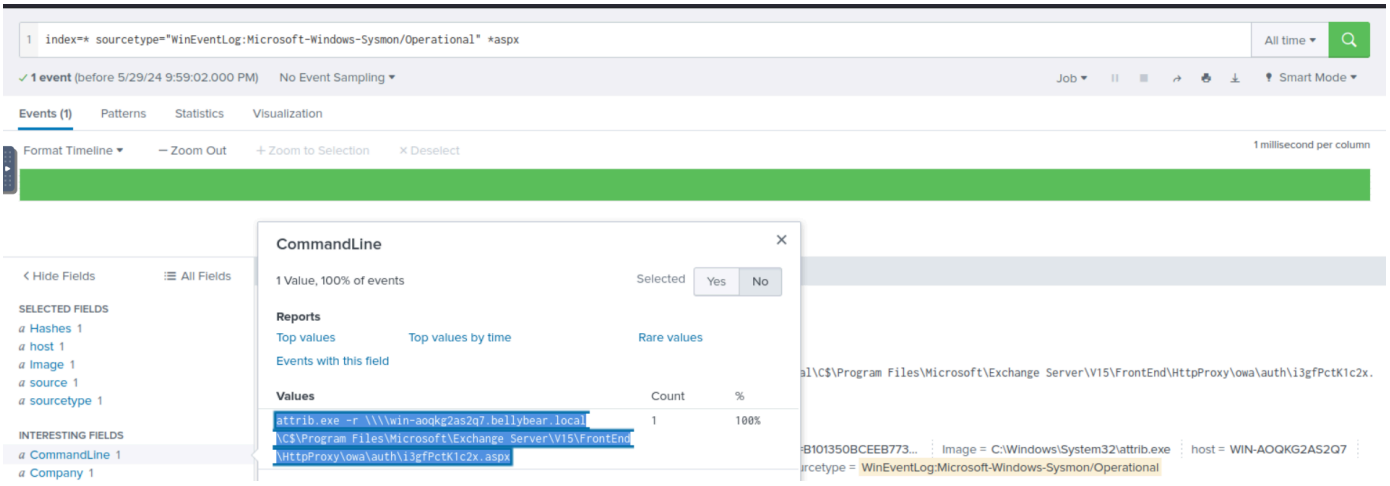




What is the command line that executed this web shell?

index=* sourcetype="WinEventLog:Microsoft-Windows-Sysmon/Operational" *aspx

Now we need to find out which command is used to run the web shell. We can find this by searching for the name of the .aspx file in Splunk. By checking the results we can find the attrib.exe command used to run the web shell.



What three CVEs did this exploit leverage? Provide the answer in ascending order.

https://www.securin.io/articles/is-conti-ransomware-on-a-roll/

CVE-2020-0796,CVE-2018-13374,CVE-2018-13379

Conti - A Cheat Sheet

We analyzed three CVEs being exploited by the Conti group - CVE-2020-0796, CVE-2018-13374, CVE-2018-13379, and here is our analysis about them -

- CVE-2020-0796 is a critical RCE/PE vulnerability, with a severity score of 10. This vulnerability also goes by the names CoronaBlue and SMBGhost, and was one of the top exploited vulnerabilities of 2020.
- CVE-2018-13379 is a critical RCE vulnerability that allows for unauthenticated attacks, and has a severity score of 9.8.
- CVE-2018-13374 is a high-rated vulnerability, with a severity score of 8.8 and can be exploited to compromise web applications.
- CWE Weakness Categories
    - CVE-2020-0796 is categorised under a weakness leading to improper input validation, CWE-20
    - CVE-2018-13379 falls under a weakness leading to improper limitation of a pathname to a restricted directory, CWE-22
    - CVE-2018-13374 belongs to CWE-732 that leads to assignment of incorrect permissions for critical resources
- CVE-2020-0796 is present in two Microsoft products - Windows 10 and Windows Server 2016, while the other two exist in Fortinet's FortiOS.
- A patch for CVE-2020-0796 has been available since March 2020, while vendors recommend upgrading to the latest version of FortiOS for the other two vulnerabilities.