Investigation using sysmon

By
Seenuvasan

Task 1: Introduction

It is highly recommended that the Windows Event Log room be completed before attempting this room, as the foundational knowledge on windows events will help us navigate this room. In addition, we will be utilizing the tools we learned in the room.

Moving on, Sysmon is part of the Windows Sysinternals package. It functions similar to Windows Event Logs that it is used to monitor and log events on Windows.

This room uses a modified version of the Blue and Ice boxes, as well as Sysmon logs from the Hololive network lab.

Task 2: Sysmon Overview

Sysmon Overview

From the Microsoft Docs, “System Monitor (Sysmon) is a Windows system service and device driver that, once installed on a system, remains resident across system reboots to monitor and log system activity to the Windows event log. It provides detailed information about process creations, network connections, and changes to file creation time. By collecting the events it generates using Windows Event Collection or SIEM agents and subsequently analyzing them, you can identify malicious or anomalous activity and understand how intruders and malware operate on your network.”

Sysmon is most commonly used in conjunction with SIEM for further analysis to identify malicious or suspicious activities in a network. Sysmon will start in the Windows boot process if it is installed on an endpoint. The

focus of this room will be on Sysmon and how to view events with Windows Event Viewer and the other methods of accessing log files.

Sysmon generated Eevents are stored in *Applications and Services Logs/Microsoft/Windows/Sysmon/Operational*

“Note that *Sysmon* does not provide analysis of the events it generates, nor does it attempt to protect or hide itself from attackers.”

Sysmon Config Overview

A configuration file is required for Sysmon to tell the binary how to analyze the events that it is receiving. In this room, we will be using this quality configuration from SwiftOnSecurity: Sysmon-Config.

Over all Sysmon has 29 Event IDs.

Note: As there are so many Event IDs Sysmon analyzes. we will only be going over a few of the ones that we think are most important to understand.

Event ID 1: Process Creation

This event will look for any processes that have been created. You can use this to look for known suspicious processes or processes with typos that would be considered an anomaly. This event will use the CommandLine and Image XML tags.

```
<RuleGroup name="" groupRelation="or"> <ProcessCreate onmatch="exclude"> <CommandLine condition="is">C:\\Windows\\system32\\svchost.exe -k appmodel -p -s camsvc</CommandLine> </ProcessCreate></RuleGroup>
```

The above code snippet is specifying the Event ID to pull from as well as what condition to look for. In this case, it is excluding the svchost.exe process from the event logs.

Event ID 3: Network Connection

The network connection event will look for events that occur remotely. This will include files and sources of suspicious binaries as well as opened ports. This event will use the Image and DestinationPort XML tags.

```
<RuleGroup name="" groupRelation="or"> <NetworkConnect onmatch="include"> <Image condition="image">nmap.exe</Image> <DestinationPort name="Alert,Metasploit" condition="is">4444</DestinationPort> </NetworkConnect></RuleGroup>
```

The above code snippet includes two ways to identify suspicious network connection activity. The first way will identify files transmitted over open ports. In this case, we are specifically looking for nmap.exe which will then be reflected within the event logs. The second method identifies open ports and specifically port 4444 which is commonly used with Metasploit. If the condition is met an event will be created and ideally trigger an alert for the SOC to further investigate.

Event ID 7: Image Loaded

This event will look for DLLs loaded by processes, which is useful when hunting for DLL Injection and DLL Hijacking attacks. It is recommended to exercise caution when using this Event ID as it causes a high system load. This event will use the Image, Signed, ImageLoaded, and Signature XML tags.

```
<RuleGroup name="" groupRelation="or"> <ImageLoad onmatch="include"> <ImageLoaded condition="contains">\\Temp\\</ImageLoaded> </ImageLoad></RuleGroup>
```

The above code snippet will look for any DLLs that have been loaded within the \Temp\ directory. If a DLL is loaded within this directory it can be considered an anomaly and should be further investigateded.

Event ID 8: CreateRemoteThread

The CreateRemoteThread Event ID will monitor for processes injecting code into other processes. The CreateRemoteThread function is used for legitimate tasks and applications. However, it could be used by malware to hide malicious activity. This event will use the SourceImage, TargetImage, StartAddress, and StartFunction XML tags.

```
<RuleGroup name="" groupRelation="or"> <CreateRemoteThread onmatch="include">
<StartAddress name="Alert,Cobalt Strike" condition="end with">0B80</StartAddress>
<SourceImage condition="contains">\\</SourceImage> </CreateRemoteThread></RuleGroup>
```

The above code snippet shows two ways of monitoring for CreateRemoteThread. The first method will look at the memory address for a specific ending condition which could be an indicator of a Cobalt Strike beacon. The second method will look for injected processes that do not have a parent process. This should be considered an anomaly and require further investigation.

Event ID 10: ProcessAccess

The process accessed event reports when a process opens another process, an operation that’s often followed by information queries or reading and writing the address space of the target process. This enables detection of

hacking tools that read the memory contents of processes like Local Security Authority (Lsass.exe) in order to steal credentials for use in Pass-the-Hash attacks. Enabling it can generate significant amounts of logging if there are diagnostic utilities active that repeatedly open processes to query their state, so it generally should only be done so with filters that remove expected accesses.

Event ID 11: File Created

This event ID is will log events when files are created or overwritten the endpoint. This could be used to identify file names and signatures of files that are written to disk. This event uses TargetFilename XML tags.

```
<RuleGroup name="" groupRelation="or"> <FileCreate onmatch="include">
****<TargetFilename name="Alert,Ransomware"
condition="contains">HELP_TO_SAVE_FILES</TargetFilename> </FileCreate></RuleGroup>
```

The above code snippet is an example of a ransomware event monitor. This is just one example of a variety of different ways you can utilize Event ID 11.

Event ID 12 / 13 / 14: Registry Event

This event looks for changes or modifications to the registry. Malicious activity from the registry can include persistence and credential abuse. This event uses TargetObject XML tags.

```
<RuleGroup name="" groupRelation="or"> <RegistryEvent onmatch="include"> <TargetObject
name="T1484" condition="contains">Windows\\System\\Scripts</TargetObject>
</RegistryEvent></RuleGroup>
```

The above code snippet will look for registry objects that are in the *"Windows\System\Scripts"* directory as this is a common directory for adversaries to place scripts to establish persistence.

Event ID 15: FileCreateStreamHash

This event will look for any files created in an alternate data stream. This is a common technique used by adversaries to hide malware. This event uses TargetFilename XML tags.

```
<RuleGroup name="" groupRelation="or"> <FileCreateStreamHash onmatch="include">
<TargetFilename condition="end with">.hta</TargetFilename>
</FileCreateStreamHash></RuleGroup>
```

The above code snippet will look for files with the .hta extension that have been placed within an alternate data stream.

Event ID 22: Event

This event will log all DNS queries and events for analysis. The most common way to deal with these events is to exclude all trusted domains that you know will be very common "noise" in your environment. Once you get rid of the noise you can then look for DNS anomalies. This event uses QueryName XML tags.

```
<RuleGroup name="" groupRelation="or"> <DnsQuery onmatch="exclude"> <QueryName
condition="end with">.microsoft.com</QueryName> </DnsQuery></RuleGroup>
```

The above code snippet will get exclude any DNS events with the .microsoft.com query. This will get rid of the noise that you see within the environment.

There are a variety of ways and tags that you can use to customize your configuration files. We will be using the ION-Storm and SwiftOnSecurity config files for the rest of this room however feel free to use your own configuration files.

Answer the questions below

Read the above and become familiar with the Sysmon Event IDs.

Task 3: Installing and Preparing Sysmon

Installing Sysmon

The installation for Sysmon is fairly straightforward and only requires downloading the binary from the Microsoft website. You can also download all of the Sysinternals tools with a PowerShell command if you wanted to rather than grabbing a single binary. It is also recommended to use a Sysmon config file along with Sysmon to get more detailed and high-quality event tracing. As an example config file we will be using the sysmon-config file from the SwiftOnSecurity GitHub repo.

You can find the Sysmon binary from the Microsoft Sysinternals website. You can also download the Microsoft Sysinternal Suite or use the below command to run a PowerShell module download and install all of the Sysinternals tools.

PowerShell command: `Download-SysInternalsTools C:\\Sysinternals`

To fully utilize Sysmon you will also need to download a Sysmon config or create your own config. We suggest downloading the SwiftOnSecurity

sysmon-config. A Sysmon config will allow for further granular control over the logs as well as more detailed event tracing. In this room, we will be using both the SwiftOnSecurity configuration file as well as the ION-Storm config file.

Starting Sysmon

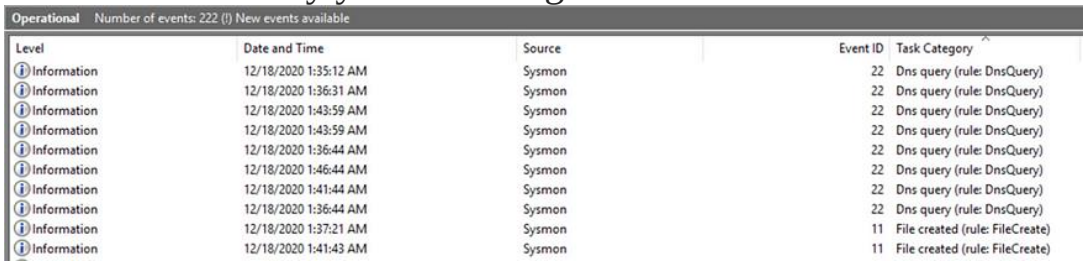
To start Sysmon you will want to open a new PowerShell or Command Prompt as an Administrator. Then, run the below command it will execute the Sysmon binary, accept the end-user license agreement, and use SwiftOnSecurity config file.

```
Sysmon.exe -accepteula -i ..\\Configuration\\swift.xml
```

Now that Sysmon is started with the configuration file we want to use, we can look at the Event Viewer to monitor events. The event log is located under *Applications and Services Logs/Microsoft/Windows/Sysmon/Operational*

Note: At any time you can change the configuration file used by uninstalling or updating the current configuration and replacing it with a new configuration file. For more information look through the Sysmon help menu.

If installed correctly your event log should look similar to the following:



For this room, we have already created an environment with Sysmon and configuration files for you. Deploy and use this machine for the remainder of this room.

Machine IP: `MACHINE_IP`

User: `THM-Analyst`

Pass: `5TgcYzF84tcBSuL1Boa%dzcvf`

The machine will start in a split-screen view. In case the VM is not visible, use the blue Show Split View button at the top-right of the page.

Answer the questions below

Deploy the machine and start Sysmon.

Task 4: Cutting out the Noise

Malicious Activity Overview

Since most of the normal activity or “noise” seen on a network is excluded or filtered out with Sysmon we’re able to focus on meaningful events. This allows us to quickly identify and investigate suspicious activity. When actively monitoring a network you will want to use multiple detections and techniques simultaneously in an effort to identify threats. For this room, we will only be looking at what suspicious logs will look like with both Sysmon configs and how to optimize your hunt using only Sysmon. We will be looking at how to detect ransomware, persistence, Mimikatz, Metasploit, and Command and Control (C2) beacons. Obviously, this is

only showcasing a small handful of events that could be triggered in an environment. The methodology will largely be the same for other threats. It really comes down to using an ample and efficient configuration file as it can do a lot of the heavy lifting for you.

You can either download the event logs used for this task or you can open them from the Practice directory on the provided machine.

Sysmon “Best Practices”

Sysmon offers a fairly open and configurable platform for you to use. Generally speaking, there are a few best practices that you could implement to ensure you’re operating efficiently and not missing any potential threats. A few common best practices are outlined and explained below.

- Exclude > Include

When creating rules for your Sysmon configuration file it is typically best to prioritize excluding events rather than including events. This prevents you from accidentally missing crucial events and only seeing the events that matter the most.

- CLI gives you further control

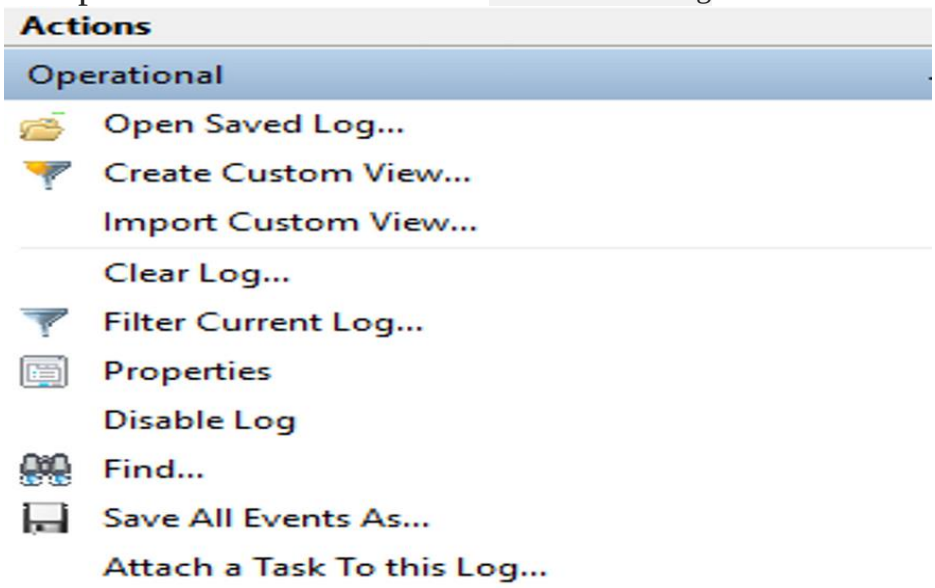
As is common with most applications the CLI gives you the most control and filtering allowing for further granular control. You can use either `Get-WinEvent` or `wevutil.exe` to access and filter logs. As you incorporate Sysmon into your SIEM or other detection solutions these tools will become less used and needed.

- Know your environment before implementation

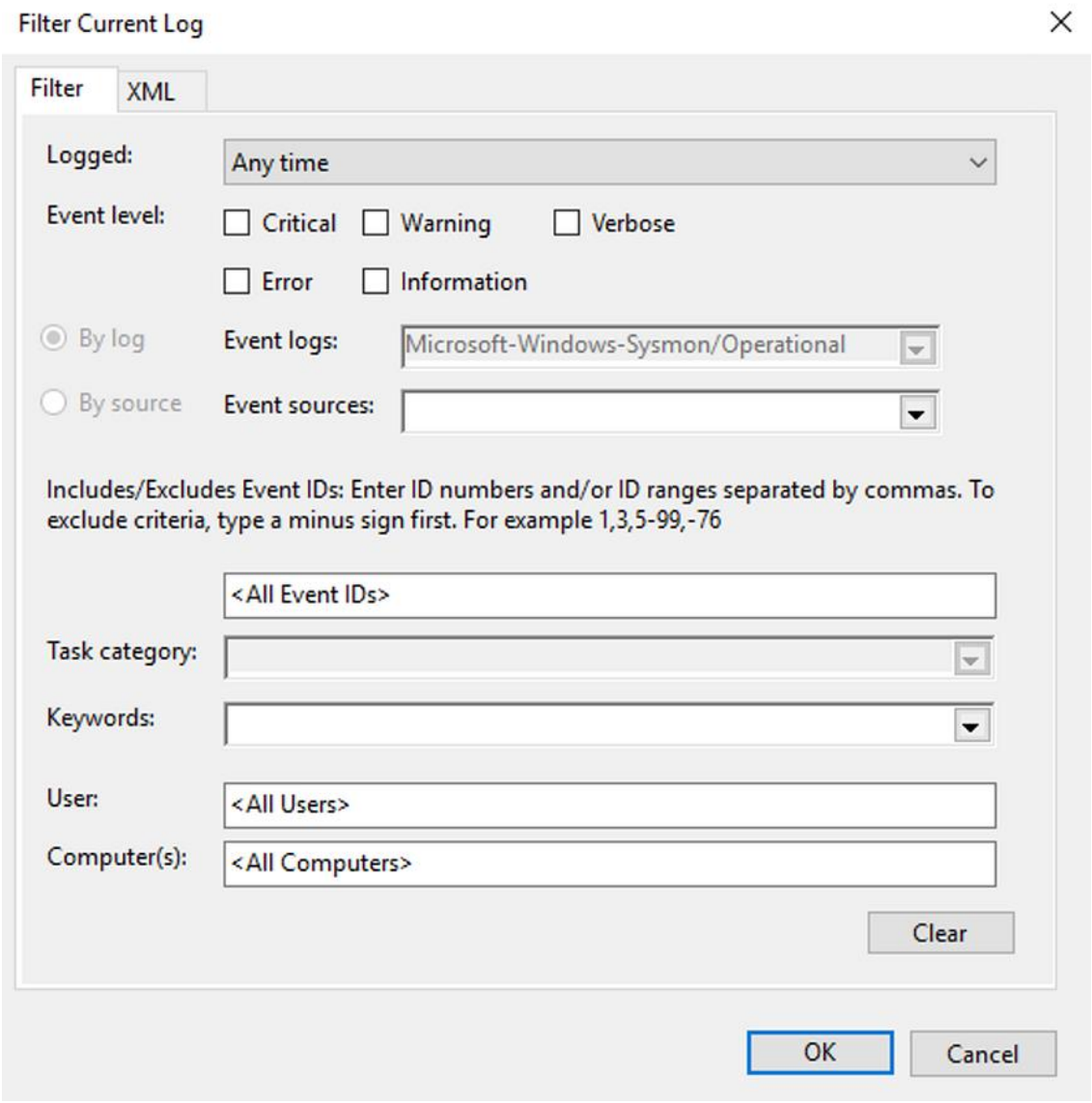
Knowing your environment is important when implementing any platform or tool. You should have a firm understanding of the network or environment you are working within to fully understand what is normal and what is suspicious in order to effectively craft your rules.

Filtering Events with Event Viewer

Event Viewer might not the best for filtering events and out-of-the-box offers limited control over logs. The main filter you will be using with Event Viewer is by filtering the `EventID` and keywords. You can also choose to filter by writing XML but this is a tedious process that doesn't scale well. To open the filter menu select `Filter Current Log` from the Actions menu.



If you have successfully opened the filter menu it should look like the menu below.



From this menu, we can add any filters or categories that we want.

Filtering Events with PowerShell

To view and filter events with PowerShell we will be using `Get-WinEvent` along with `XPath` queries. We can use any XPath queries that can be found in the XML view of events. We will be using `wevutil.exe` to view events once filtered. The command line is typically used over the Event Viewer GUI as it allows for further granular control and filtering whereas the GUI does not. For more information about using `Get-WinEvent` and `wevutil.exe` check out the Windows Event Log room.

For this room, we will only be going over a few basic filters as the Windows Event Log room already extensively covers this topic.

Filter by Event ID: `*/System/EventID=<ID>`

Filter by XML Attribute/Name: `*/EventData/Data[@Name="<XML Attribute/Name>"]`

Filter by Event Data: `*/EventData/Data=<Data>`

We can put these filters together with various attributes and data to get the most control out of our logs. Look below for an example of using `Get-WinEvent` to look for network connections coming from port 4444.

```
Get-WinEvent -Path C:\\Users\\THM-Analyst\\Desktop\\Scenarios\\Practice\\Hunting_Metasploit.evtx -FilterXPath '*/System/EventID=3 and */EventData/Data[@Name="DestinationPort"] and */EventData/Data=4444'
```

Answer the questions below

Read the above and practice filtering events.

How many event ID 3 events are in C:\Users\THM-Analyst\Desktop\Scenarios\Practice\Filtering.evtx?

Answer: 73,951

Using the two commands below would give us the same answer.

```
(Get-WinEvent  -Path.\\Filtering.evtx -FilterXPath '*/System/EventID=3').count
```

```
Get-WinEvent -Path .\\Filtering.evtx -FilterXPath '*/System/EventID=3' | Measure-Object -Line
```

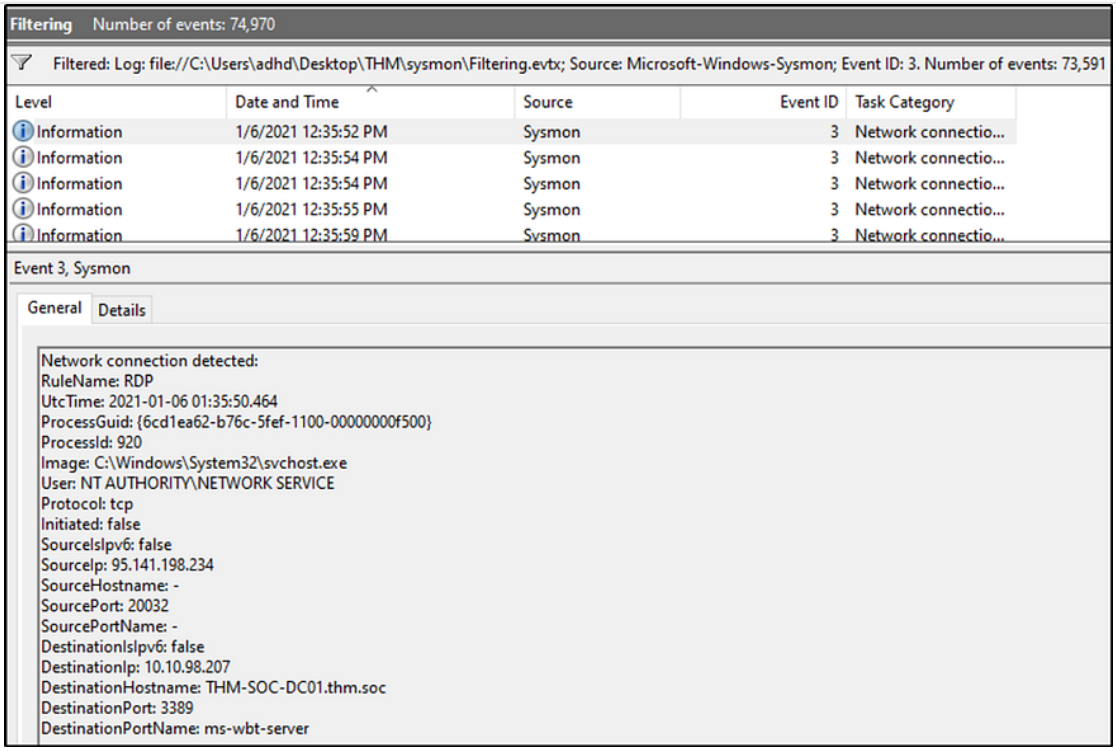
```
PS C:\Users\THM-Analyst\Desktop\Scenarios\Practice> Get-WinEvent -Path .\Filtering.evtx -FilterXPath '*/System/EventID=3' | Measure-Object -Line

Lines Words Characters Property
----- ----- ---------- --------
73591
```

What is the UTC time created of the first network event in C:\Users\THM-Analyst\Desktop\Scenarios\Practice\Filtering.evtx?

Answer: 2021–01–06 01:35:50.464

We can get the answer either using PowerShell or the Event Viewer.

```
Get-WinEvent -Path .\\Filtering.evtx -FilterXPath '*/System/EventID=3' -Oldest -MaxEvents 1 | Select-Object -Property *
```



Task 5: Hunting Metasploit

Hunting Metasploit

Metasploit is a commonly used exploit framework for penetration testing and red team operations. Metasploit can be used to easily run exploits on a machine and connect back to a meterpreter shell. We will be hunting the meterpreter shell itself and the functionality it uses. To begin hunting we will look for network connections that originate from suspicious ports such as `4444` and `5555`. By default, Metasploit uses port 4444. If there is a connection to any IP known or unknown it should be investigated. To start an investigation you can look at packet captures from the date of the log to begin looking for further information about the adversary. We can also look for suspicious processes created. This method of hunting can be applied to other various RATs and C2 beacons.

For more information about this technique and tools used check out MITRE ATT&CK Software.

For more information about how malware and payloads interact with the network check out the Malware Common Ports Spreadsheet. This will be covered in further depth in the Hunting Malware task.

You can download the event logs used in this room from this task or you can open them in the Practice folder on the provided machine.

Hunting Network Connections

We will first be looking at a modified Ion-Security configuration to detect the creation of new network connections. The code snippet below will use event ID 3 along with the destination port to identify active connections specifically connections on port `4444` and `5555`.

<RuleGroup name="" groupRelation="or"> <NetworkConnect onmatch="include">
<DestinationPort condition="is">4444</DestinationPort> <DestinationPort
condition="is">5555</DestinationPort> </NetworkConnect></RuleGroup>

Open C:\\Users\\THM-Analyst\\Desktop\\Scenarios\\Practice\\Hunting_Metasploit.evtx in Event Viewer to view a basic Metasploit payload being dropped onto the machine.

!https://i.imgur.com/1VkrpJ3.png

Once we identify the event it can give us some important information we can use for further investigation like the ProcessID and Image.

Hunting for Open Ports with PowerShell

To hunt for open ports with PowerShell we will be using the PowerShell module Get-WinEvent along with XPath queries. We can use the same XPath queries that we used in the rule to filter out events from NetworkConnect with DestinationPort. The command line is typically used over the Event Viewer GUI because it can allow for further granular control and filtering that the GUI does not offer. For more information about using XPath and the command line for event viewing, check out the Windows Event Log room by Heavenraiza.

```
Get-WinEvent -Path C:\\Users\\THM-Analyst\\Desktop\\Scenarios\\Practice\\Hunting_Metasploit.evtx -FilterXPath '*/System/EventID=3 and */EventData/Data[@Name="DestinationPort"] and */EventData/Data=4444'
```

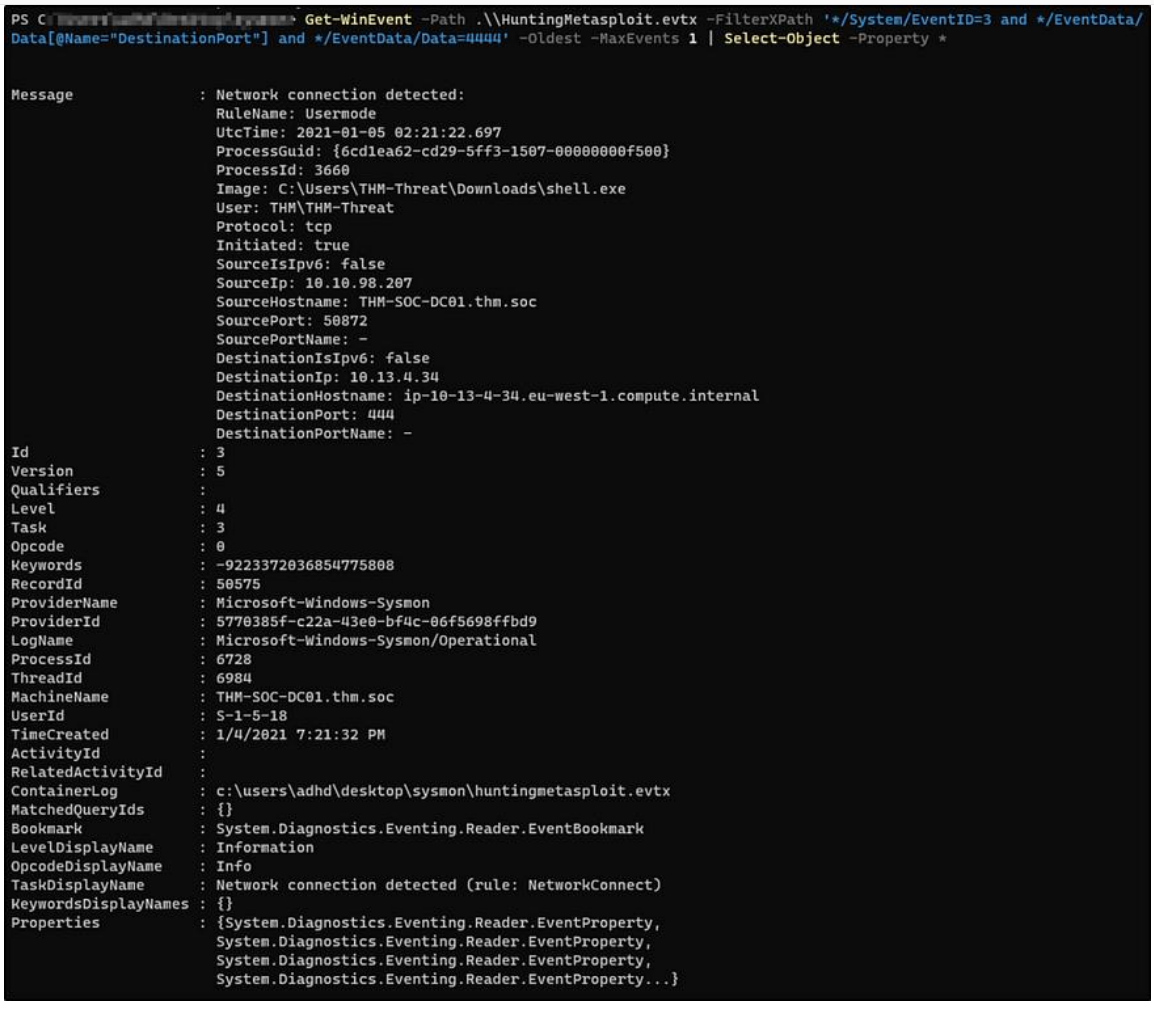
We can break this command down by its filters to see exactly what it is doing. It is first filtering by Event ID 3 which is the network connection ID.

It is then filtering by the data name in this case DestinationPort as well as the specific port that we want to filter. We can adjust this syntax along with our events to get exactly what data we want in return.

Answer the questions below

Read the above and practice hunting Metasploit with the provided event file.

```
Get-WinEvent -Path .\\HuntingMetasploit.evtx -FilterXPath '*/System/EventID=3 and */EventData/Data[@Name="DestinationPort"] and */EventData/Data=4444' -Oldest -MaxEvents 1 | Select-Object -Property *
```



Task 6: Detecting Mimikatz

Detecting Mimikatz Overview

Mimikatz is well known and commonly used to dump credentials from memory along with other Windows post-exploitation activity. Mimikatz is mainly known for dumping LSASS. We can hunt for the file created, execution of the file from an elevated process, creation of a remote thread, and processes that Mimikatz creates. Anti-Virus will typically pick up Mimikatz as the signature is very well known but it is still possible for threat actors to obfuscate or use droppers to get the file onto the device. For this hunt, we will be using a custom configuration file to minimize network noise and focus on the hunt.

For more information about this technique and the software used check out MITRE ATTACK T1055 and S0002.

You can download the event logs used in this room from this task or you can open them in the Practice folder on the provided machine.

Detecting File Creation

The first method of hunting for Mimikatz is just looking for files created with the name Mimikatz. This is a simple technique but can allow you to find anything that might have bypassed AV. Most of the time when dealing with an advanced threat you will need more advanced hunting techniques like searching for LSASS behavior but this technique can still be useful.

This is a very simple way of detecting Mimikatz activity that has bypassed anti-virus or other detection measures. But most of the time it is preferred

to use other techniques like hunting for LSASS specific behavior. Below is a snippet of a config to aid in the hunt for Mimikatz.

```
<RuleGroup name="" groupRelation="or"> <FileCreate onmatch="include"> <TargetFileName condition="contains">mimikatz</TargetFileName> </FileCreate></RuleGroup>
```

As this method will not be commonly used to hunt for anomalies we will not be looking at any event logs for this specific technique.

Hunting Abnormal LSASS Behavior

We can use the *ProcessAccess* event ID to hunt for abnormal LSASS behavior. This event along with LSASS would show potential LSASS abuse which usually connects back to Mimikatz some other kind of credential dumping tool. Look below for more detail on hunting with these techniques.

If LSASS is accessed by a process other than *svchost.exe* it should be considered suspicious behavior and should be investigated further, to aid in looking for suspicious events you can use a filter to only look for processes besides svchost.exe. Sysmon will provide us further details to help lead the investigation such as the file path the process originated from. To aid in detections we will be using a custom configuration file. Below is a snippet of the config that will aid in the hunt.

```
<RuleGroup name="" groupRelation="or"> <ProcessAccess onmatch="include"> <TargetImage condition="image">lsass.exe</TargetImage> </ProcessAccess></RuleGroup>
```

Open `C:\\Users\\THM-Analyst\\Desktop\\Scenarios\\Practice\\Hunting_LSASS.evtx` in Event Viewer to view an attack using an obfuscated version of Mimikatz to dump credentials from memory.

We see the event that has the Mimikatz process accessed but we also see a lot of svchost.exe events? We can alter our config to exclude events with the `SourceImage` event coming from svhost.exe. Look below for a modified configuration rule to cut down on the noise that is present in the event logs.

```
<RuleGroup name="" groupRelation="or"> <ProcessAccess onmatch="exclude"> <SourceImage condition="image">svchost.exe</SourceImage> </ProcessAccess> <ProcessAccess onmatch="include"> <TargetImage condition="image">lsass.exe</TargetImage> </ProcessAccess></RuleGroup>
```

By modifying the configuration file to include this exception we have cut down our events significantly and can focus on only the anomalies. This technique can be used throughout Sysmon and events to cut down on “noise” in logs.

Detecting LSASS Behavior with PowerShell

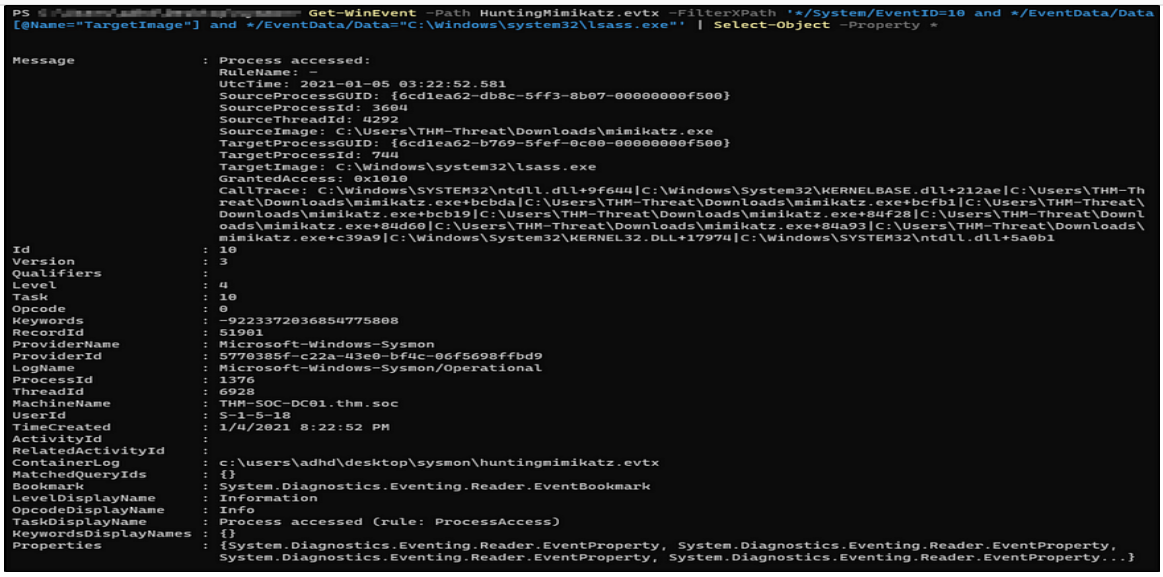
To detect abnormal LSASS behavior with PowerShell we will again be using the PowerShell module `Get-WinEvent` along with `XPath` queries. We can use the same XPath queries used in the rule to filter out the other processes from `TargetImage`. If we use this alongside a well-built configuration file with a precise rule it will do a lot of the heavy lifting for us and we only need to filter a small amount.

```
Get-WinEvent -Path C:\\Users\\THM-Analyst\\Desktop\\Scenarios\\Practice\\Hunting_Mimikatz.evtx -FilterXPath '*/System/EventID=10 and */EventData/Data[@Name="TargetImage"] and */EventData/Data="C:\\Windows\\system32\\lsass.exe"'
```

Answer the questions below

Read the above and practice detecting Mimikatz with the provided evtx.

```
Get-WinEvent -Path HuntingMimikatz.evtx -FilterXPath '*/System/EventID=10 and */EventData/Data[@Name="TargetImage"] and */EventData/Data="C:\\Windows\\system32\\lsass.exe"' | Select-Object -Property *
```



Task 7: Hunting Malware

Hunting Malware Overview

Malware has many forms and variations with different end goals. The two types of malware that we will be focusing on are RATs and backdoors. RATs or Remote Access Trojans are used similar to any other payload to gain remote access to a machine. RATs typically come with other Anti-Virus and detection evasion techniques that make them different than other payloads like MSFVenom. A RAT typically also uses a Client-Server model and comes with an interface for easy user administration. Examples of RATs are `Xeexe` and `Quasar`. To help detect and hunt malware we will need to first identify the malware that we want to hunt or detect and identify ways that we can modify configuration files, this is known as hypothesis-based hunting. There are of course a plethora of other ways to detect and log malware however we will only be covering the basic way of detecting open back connect ports.

For more information about this technique and examples of malware check out MITRE ATT&CK Software.

You can download the event logs used in this room from this task or you can open them in the Practice folder on the provided machine.

Hunting Rats and Servers

The first technique we will use to hunt for malware is a similar process to hunting Metasploit. We can look through and create a configuration file to hunt and detect suspicious ports open on the endpoint. By using known

suspicious ports to include in our logs we can add to our hunting methodology in which we can use logs to identify adversaries on our network then use packet captures or other detection strategies to continue the investigation. The code snippet below is from the Ion-Storm configuration file which will alert when specific ports like `1034` and `1604` as well as exclude common network connections like OneDrive, by excluding events we still see everything that we want without missing anything and cutting down on noise.

When using configuration files in a production environment you must be careful and understand exactly what is happening within the configuration file an example of this is the Ion-Storm configuration file excludes port 53 as an event. Attackers and adversaries have begun to use port 53 as part of their malware/payloads which would go undetected if you blindly used this configuration file as-is.

For more information about the ports that this configuration file alerts on check out this spreadsheet.

```
<RuleGroup name="" groupRelation="or"> <NetworkConnect onmatch="include">
<DestinationPort condition="is">1034</DestinationPort> <DestinationPort
condition="is">1604</DestinationPort> </NetworkConnect> <NetworkConnect
onmatch="exclude"> <Image condition="image">OneDrive.exe</Image>
</NetworkConnect></RuleGroup>
```

Open `C:\\Users\\THM-Analyst\\Desktop\\Scenarios\\Practice\\Hunting_Rats.evtx` in Event Viewer to view a live rat being dropped onto the server.

In the above example, we are detecting a custom RAT that operates on port 8080. This is a perfect example of why you want to be careful when excluding events in order to not miss potential malicious activity.

Hunting for Common Back Connect Ports with PowerShell

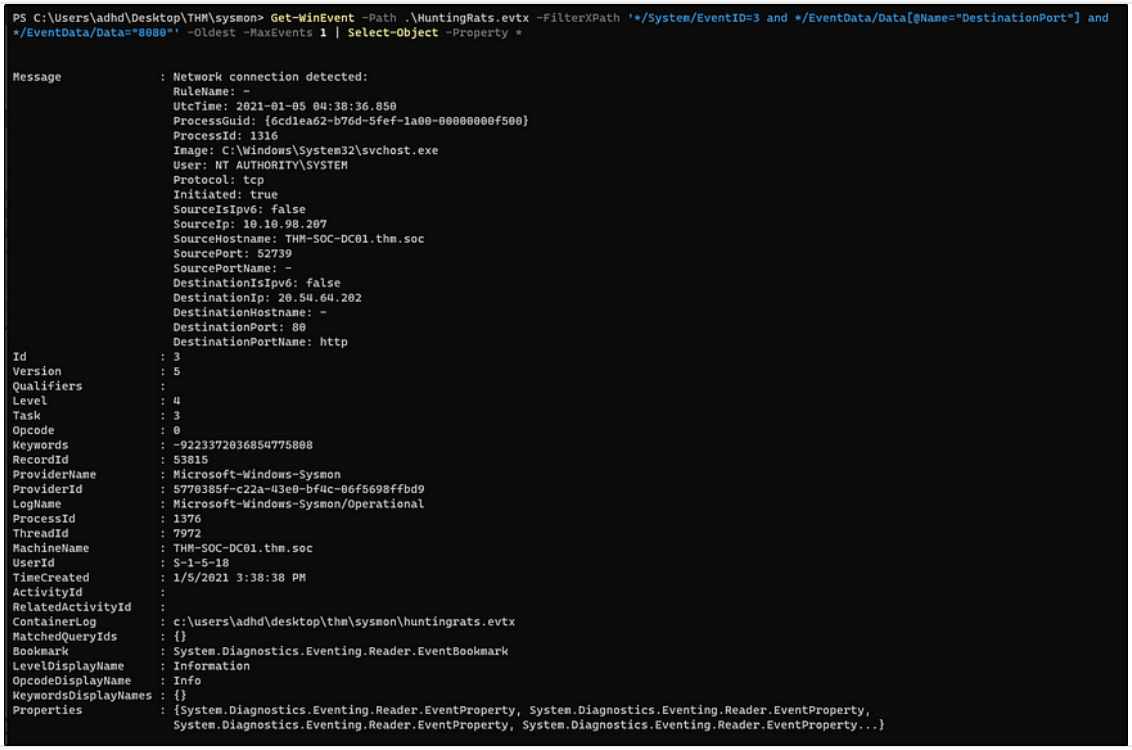
Just like previous sections when using PowerShell we will again be using the PowerShell module `Get-WinEvent` along with `XPath` queries to filter our events and gain granular control over our logs. We will need to filter on the `NetworkConnect` event ID and the `DestinationPort` data attribute. If you're using a good configuration file with a reliable set of rules it will do a majority of the heavy lifting and filtering to what you want should be easy.

```
Get-WinEvent -Path C:\\Users\\THM-Analyst\\Desktop\\Scenarios\\Practice\\Hunting_Rats.evtx -FilterXPath '*/System/EventID=3 and */EventData/Data[@Name="DestinationPort"] and */EventData/Data=8080'
```

Answer the questions below

Read the Above and practice hunting rats and C2 servers with back connect ports.

```
Get-WinEvent -Path .\\HuntingRats.evtx -FilterXPath '*/System/EventID=3 and */EventData/Data[@Name="DestinationPort"] and */EventData/Data="8080"' -Oldest -MaxEvents 1 | Select-Object -Property *
```




```
Get-WinEvent -Path .\\HuntingRats.evtx -FilterXPath '*/System/EventID=3 and */EventData/Data[@Name="DestinationPort"] and */EventData/Data="1034"' -Oldest -MaxEvents 1 | Select-Object -Property *
```

```
PS C:\Users\adhd\Desktop\THM\sysmon> Get-WinEvent -Path .\HuntingRats.evtx -FilterXPath '*/System/EventID=3 and */EventData/Data[@Name="DestinationPort"] and
*/EventData/Data="1034"' -Oldest -MaxEvents 1 | Select-Object -Property *


Message              : Network connection detected:
                       RuleName: -
                       UtcTime: 2021-01-05 04:38:36.850
                       ProcessGuid: {6cd1ea62-b76d-5fef-1a00-00000000f500}
                       ProcessId: 1316
                       Image: C:\Windows\System32\svchost.exe
                       User: NT AUTHORITY\SYSTEM
                       Protocol: tcp
                       Initiated: true
                       SourceIsIpv6: false
                       SourceIp: 10.10.98.207
                       SourceHostname: THM-SOC-DC01.thm.soc
                       SourcePort: 52739
                       SourcePortName: -
                       DestinationIsIpv6: false
                       DestinationIp: 20.54.64.202
                       DestinationHostname: -
                       DestinationPort: 80
                       DestinationPortName: http
Id                   : 3
Version              : 5
Qualifiers           :
Level                : 4
Task                 : 3
Opcode               : 0
Keywords             : -9223372036854775808
RecordId             : 53815
ProviderName         : Microsoft-Windows-Sysmon
ProviderId           : 5770385f-c22a-43e0-bf4c-06f5698ffbd9
LogName              : Microsoft-Windows-Sysmon/Operational
ProcessId            : 1376
ThreadId             : 7972
MachineName          : THM-SOC-DC01.thm.soc
UserId               : S-1-5-18
TimeCreated          : 1/5/2021 3:38:38 PM
ActivityId           :
RelatedActivityId    :
ContainerLog         : c:\users\adhd\desktop\thm\sysmon\huntingrats.evtx
MatchedQueryIds      : {}
Bookmark             : System.Diagnostics.Eventing.Reader.EventBookmark
LevelDisplayName     : Information
OpcodeDisplayName    : Info
TaskDisplayName      : Network connection detected (rule: NetworkConnect)
KeywordsDisplayNames : {}
Properties           : {System.Diagnostics.Eventing.Reader.EventProperty, System.Diagnostics.Eventing.Reader.EventProperty,
                       System.Diagnostics.Eventing.Reader.EventProperty, System.Diagnostics.Eventing.Reader.EventProperty...}
```

Get-WinEvent -Path .\\HuntingRats.evtx -FilterXPath '*/System/EventID=3 and */EventData/Data[@Name="DestinationPort"] and */EventData/Data="1604"' -Oldest -MaxEvents 1 | Select-Object -Property *


```
PS C:\Users\adhd\Desktop\THM\sysmon> Get-WinEvent -Path .\HuntingRats.evtx -FilterXPath '*/System/EventID=3 and */EventData/Data[@Name="DestinationPort"] and
*/EventData/Data="1604"' -Oldest -MaxEvents 1 | Select-Object -Property *


Message              : Network connection detected:
                       RuleName: -
                       UtcTime: 2021-01-05 04:38:36.850
                       ProcessGuid: {6cd1ea62-b76d-5fef-1a00-00000000f500}
                       ProcessId: 1316
                       Image: C:\Windows\System32\svchost.exe
                       User: NT AUTHORITY\SYSTEM
                       Protocol: tcp
                       Initiated: true
                       SourceIsIpv6: false
                       SourceIp: 10.10.98.207
                       SourceHostname: THM-SOC-DC01.thm.soc
                       SourcePort: 52739
                       SourcePortName: -
                       DestinationIsIpv6: false
                       DestinationIp: 20.54.64.202
                       DestinationHostname: -
                       DestinationPort: 80
                       DestinationPortName: http
Id                   : 3
Version              : 5
Qualifiers           :
Level                : 4
Task                 : 3
Opcode               : 0
Keywords             : -9223372036854775808
RecordId             : 53815
ProviderName         : Microsoft-Windows-Sysmon
ProviderId           : 5770385f-c22a-43e0-bf4c-06f5698ffbd9
LogName              : Microsoft-Windows-Sysmon/Operational
ProcessId            : 1376
ThreadId             : 7972
MachineName          : THM-SOC-DC01.thm.soc
UserId               : S-1-5-18
TimeCreated          : 1/5/2021 3:38:38 PM
ActivityId           :
RelatedActivityId    :
ContainerLog         : c:\users\adhd\desktop\thm\sysmon\huntingrats.evtx
MatchedQueryIds      : {}
Bookmark             : System.Diagnostics.Eventing.Reader.EventBookmark
LevelDisplayName     : Information
OpcodeDisplayName    : Info
TaskDisplayName      : Network connection detected (rule: NetworkConnect)
KeywordsDisplayNames : {}
Properties           : {System.Diagnostics.Eventing.Reader.EventProperty, System.Diagnostics.Eventing.Reader.EventProperty,
                       System.Diagnostics.Eventing.Reader.EventProperty, System.Diagnostics.Eventing.Reader.EventProperty...}
```

Task 8: Hunting Persistence

Persistence Overview

Persistence is used by attackers to maintain access to a machine once it is compromised. There is a multitude of ways for an attacker to gain persistence on a machine. We will be focusing on registry modification as well as startup scripts. We can hunt persistence with Sysmon by looking for File Creation events as well as Registry Modification events. The SwiftOnSecurity configuration file does a good job of specifically targeting persistence and techniques used. You can also filter by the Rule Names in order to get past the network noise and focus on anomalies within the event logs.

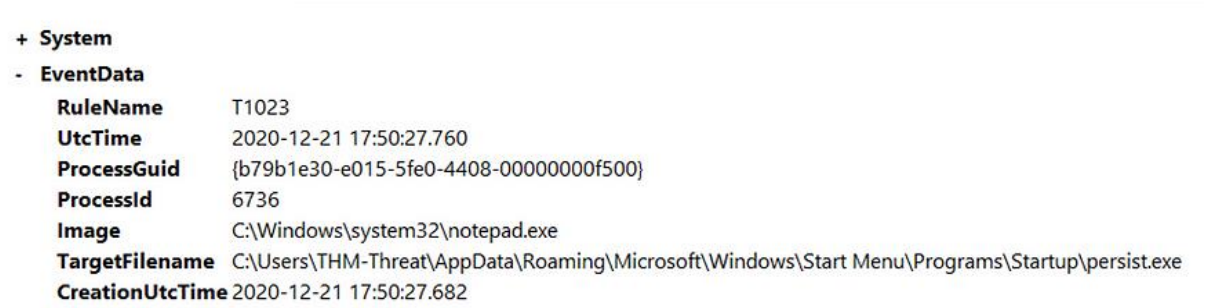
You can download the event logs used in this room from this task or you can open them in the Practice folder on the provided machine.

Hunting Startup Persistence

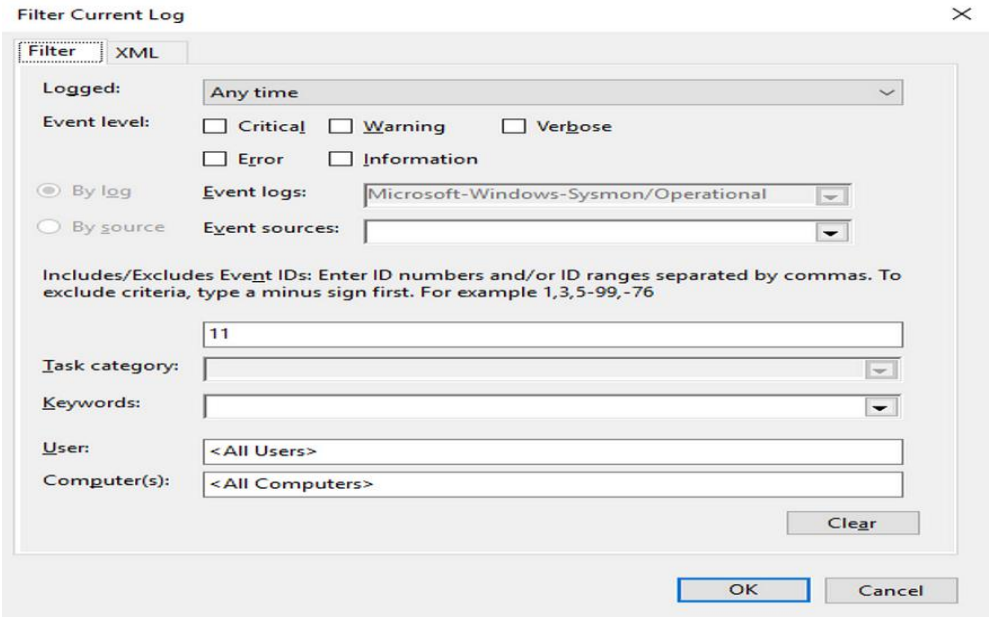
We will first be looking at the SwiftOnSecurity detections for a file being placed in the `\\Startup\\` or `\\Start Menu` directories. Below is a snippet of the config that will aid in event tracing for this technique. For more information about this technique check out MITRE ATT&CK T1547.

```
<RuleGroup name="" groupRelation="or"> <FileCreate onmatch="include"> <TargetFilename name="T1023" condition="contains">\\Start Menu</TargetFilename> <TargetFilename name="T1165" condition="contains">\\Startup\\</TargetFilename> </FileCreate></RuleGroup>
```

Open `C:\\Users\\THM-Analyst\\Desktop\\Scenarios\\Practice\\T1023.evtx` in Event Viewer to view a live attack on the machine that involves persistence by adding a malicious EXE into the Startup folder.



When looking at the Event Viewer we see that persist.exe was placed in the Startup folder. Threat Actors will almost never make it this obvious but any changes to the Start Menu should be investigated. You can adjust the configuration file to be more granular and create alerts past just the *File Created* tag. We can also filter by the `Rule Name T1023`



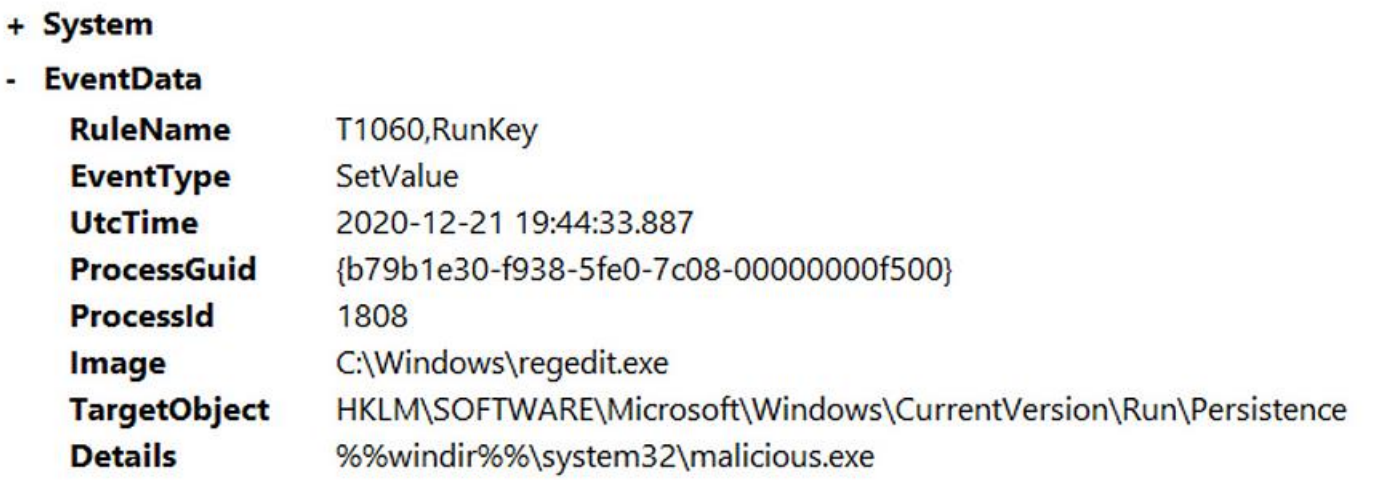
Once you have identified that a suspicious binary or application has been placed in a startup location you can begin an investigation on the directory.

Hunting Registry Key Persistence

We will again be looking at another SwiftOnSecurity detection this time for a registry modification that adjusts that places a script inside `CurrentVersion\\Windows\\Run` and other registry locations. For more information about this technique check out MITRE ATT&CK T1112.

```
<RuleGroup name="" groupRelation="or"> <RegistryEvent onmatch="include"> <TargetObject name="T1060,RunKey" condition="contains">CurrentVersion\\Run</TargetObject> <TargetObject name="T1484" condition="contains">Group Policy\\Scripts</TargetObject> <TargetObject name="T1060" condition="contains">CurrentVersion\\Windows\\Run</TargetObject> </RegistryEvent></RuleGroup>
```

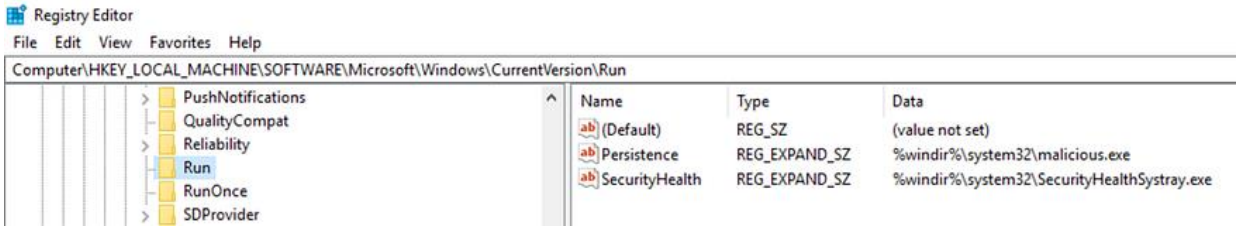
Open `C:\\Users\\THM-Analyst\\Desktop\\Scenarios\\Practice\\T1060.evtx` in Event Viewer to view an attack where the registry was modified to gain persistence.



When looking at the event logs we see that the registry was modified and malicious.exe was added to `HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\Persistence` We also see that the exe can be found at `%windir%\\System32\\malicious.exe`

Just like the startup technique, we can filter by the `RuleName T1060` to make finding the anomaly easier.

If we wanted to investigate this anomaly we would need to look at the registry as well as the file location itself. Below is the registry area where the malicious registry key was placed.



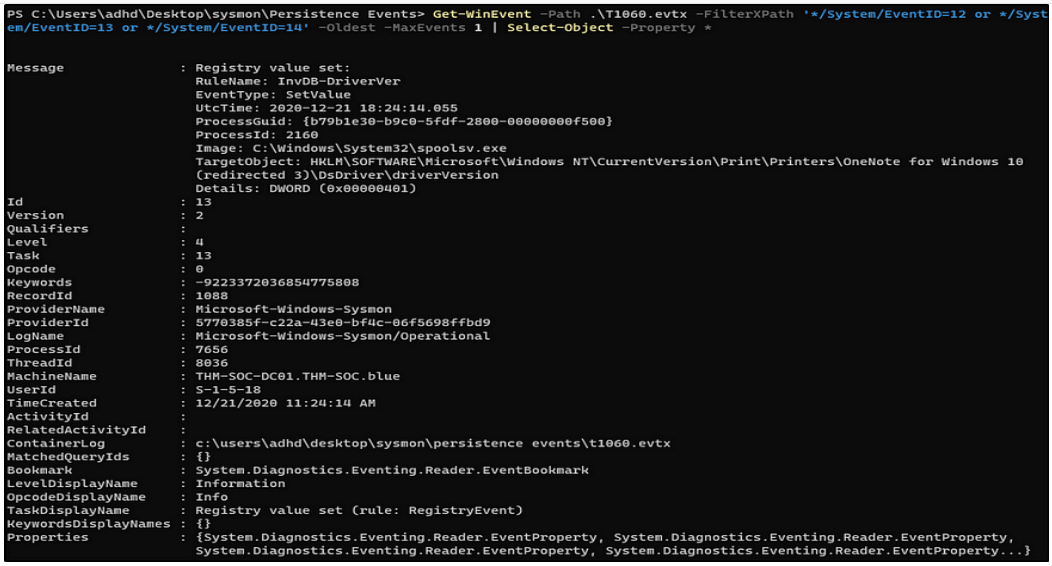
Answer the questions below

Read the above and practice hunting persistence techniques.

```
Get-WinEvent -Path .\T1023.evtx -FilterXPath '*/System/EventID=12 or */System/EventID=13 or */System/EventID=14' -Oldest -MaxEvents 1 | Select-Object -Property *
```

```
PS C:\Users\adhd\Desktop\sysmon\Persistence Events> Get-WinEvent -Path .\T1023.evtx -FilterXPath '*/System/EventID=12 or */System/EventID=13 or */System/EventID=14' -Oldest -MaxEvents 1 | Select-Object -Property *


Message             : Registry value set:
                      RuleName: InvDB-DriverVer
                      EventType: SetValue
                      UtcTime: 2020-12-21 17:45:34.861
                      ProcessGuid: {b79b1e30-b9c0-5fdf-2800-00000000f500}
                      ProcessId: 2160
                      Image: C:\Windows\System32\spoolsv.exe
                      TargetObject: HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Print\Printers\Fax (redirected
                      3)\DsDriver\driverVersion
                      Details: DWORD (0x00000401)
Id                  : 13
Version             : 2
Qualifiers          :
Level               : 4
Task                : 13
Opcode              : 0
Keywords            : -9223372036854775808
RecordId            : 132
ProviderName        : Microsoft-Windows-Sysmon
ProviderId          : 5770385f-c22a-43e0-bf4c-06f5698ffbd9
LogName             : Microsoft-Windows-Sysmon/Operational
ProcessId           : 7656
ThreadId            : 8036
MachineName         : THM-SOC-DC01.THM-SOC.blue
UserId              : S-1-5-18
TimeCreated         : 12/21/2020 10:45:34 AM
ActivityId          :
RelatedActivityId   :
ContainerLog        : c:\users\adhd\desktop\sysmon\persistence events\t1023.evtx
MatchedQueryIds     : {}
Bookmark            : System.Diagnostics.Eventing.Reader.EventBookmark
LevelDisplayName    : Information
OpcodeDisplayName   : Info
TaskDisplayName     : Registry value set (rule: RegistryEvent)
KeywordsDisplayNames : {}
Properties          : {System.Diagnostics.Eventing.Reader.EventProperty, System.Diagnostics.Eventing.Reader.EventProperty,
                      System.Diagnostics.Eventing.Reader.EventProperty, System.Diagnostics.Eventing.Reader.EventProperty...}
```

```
Get-WinEvent -Path .\T1060.evtx -FilterXPath '*/System/EventID=12 or */System/EventID=13 or */System/EventID=14' -Oldest -MaxEvents 1 | Select-Object -Property *
```



Task 9: Detecting Evasion Techniques

Evasion Techniques Overview

There are a number of evasion techniques used by malware authors to evade both anti-virus and detections. Some examples of evasion techniques are Alternate Data Streams, Injections, Masquerading, Packing/Compression, Recompiling, Obfuscation, Anti-Reversing Techniques. In this task, we will be focusing on Alternate Data Streams and Injections. Alternate Data Streams are used by malware to hide its files from normal inspection by saving the file in a different stream apart from `$DATA`. Sysmon comes with an event ID to detect newly created and accessed streams allowing us to quickly detect and hunt malware that uses ADS. Injection techniques come in many different types: Thread Hijacking, PE Injection, DLL Injection, and more. In this room, we will be focusing on DLL Injection and backdooring DLLs. This is done by taking

an already used DLL that is used by an application and overwriting or including your malicious code within the DLL.

For more information about this technique check out MITRE ATT&CK T1564 and T1055.

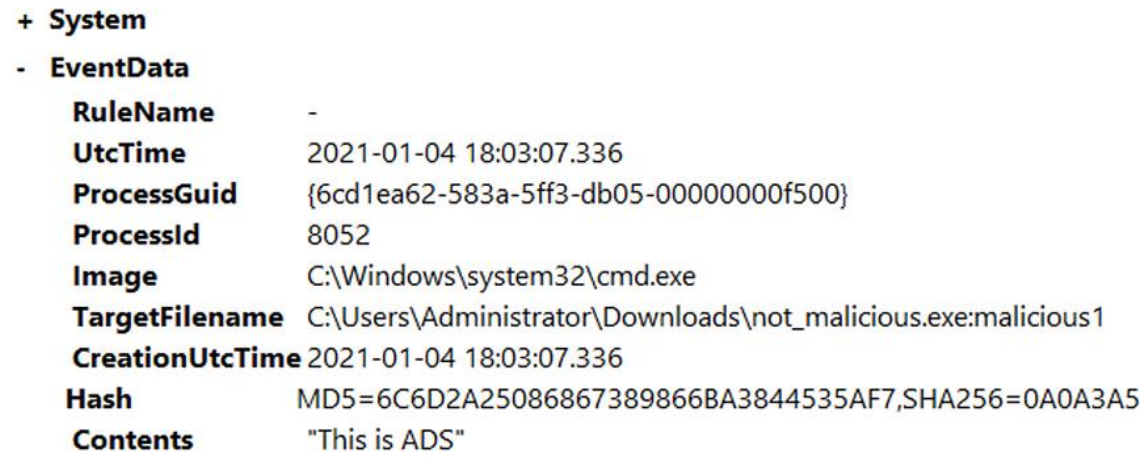
You can download the event logs used in this room from this task or you can open them in the Practice folder on the provided machine.

Hunting Alternate Data Streams

The first technique we will be looking at is hiding files using alternate data streams using Event ID 15. Event ID 15 will hash and log any NTFS Streams that are included within the Sysmon configuration file. This will allow us to hunt for malware that evades detections using ADS. To aid in hunting ADS we will be using the SwiftOnSecurity Sysmon configuration file. The code snippet below will hunt for files in the `Temp` and `Startup` folder as well as `.hta` and `.bat` extension.

```
<RuleGroup name="" groupRelation="or"> <FileCreateStreamHash onmatch="include"> <TargetFilename condition="contains">Downloads</TargetFilename> <TargetFilename condition="contains">Temp\\7z</TargetFilename> <TargetFilename condition="ends with">.hta</TargetFilename> <TargetFilename condition="ends with">.bat</TargetFilename> </FileCreateStreamHash></RuleGroup>
```

Open `C:\\Users\\THM-Analyst\\Desktop\\Scenarios\\Practice\\Hunting_ADS.evtx` in Event Viewer to view hidden files using an alternate data stream.



Listing Data Streams

```
dir /r
```

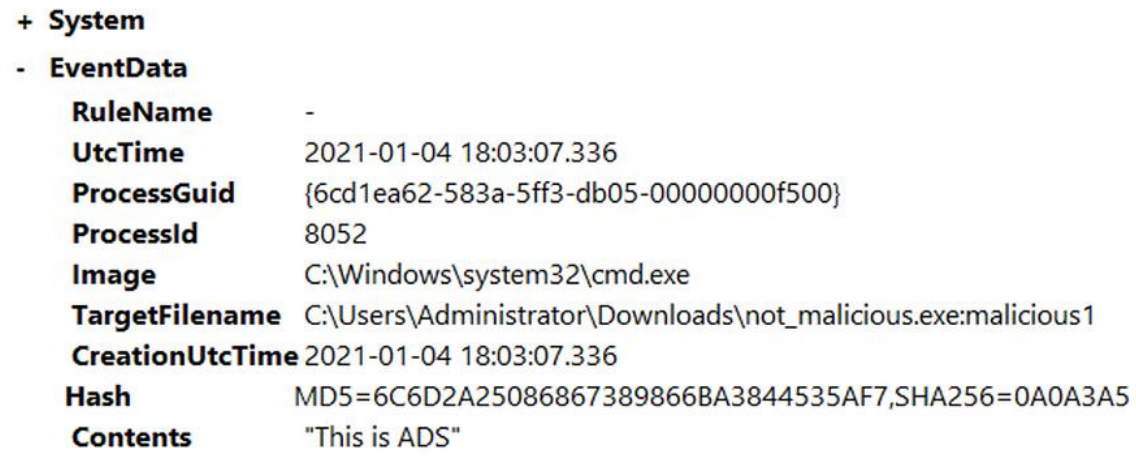
The event will show us the location of the file name as well as the contents of the file this will be useful if an investigation is necessary.

Detecting Remote Threads

Adversaries also commonly use remote threads to evade detections in combination with other techniques. Remote threads are created using the Windows API `CreateRemoteThread` and can be accessed using `OpenThread` and `ResumeThread`. This is used in multiple evasion techniques including DLL Injection, Thread Hijacking, and Process Hollowing. We will be using the Sysmon event ID 8 from the SwiftOnSecurity configuration file. The code snippet below from the rule will exclude common remote threads without including any specific attributes this allows for a more open and precise event rule.

```
<RuleGroup name="" groupRelation="or"> <CreateRemoteThread onmatch="exclude">
<SourceImage condition="is">C:\\Windows\\system32\\svchost.exe</SourceImage>
<TargetImage condition="is">C:\\Program Files
(x86)\\Google\\Chrome\\Application\\chrome.exe</TargetImage>
</CreateRemoteThread></RuleGroup>
```

Open `C:\\Users\\THM-Analyst\\Desktop\\Scenarios\\Practice\\Detecting_RemoteThreads.evtx` in Event Viewer to observe a Process Hollowing attack that abuses the notepad.exe process.



As you can see in the above image powershell.exe is creating a remote thread and accessing notepad.exe. This is obviously a PoC and could in theory execute any other kind of executable or DLL. The specific technique used in this example is called Reflective PE Injection.

Detecting Evasion Techniques with PowerShell

We have already gone through a majority of the syntax required to use PowerShell with events. Like previous tasks, we will be using `Get-WinEvent` along with the `XPath` to filter and search for files that use an alternate data stream or create a remote thread. In both of the events, we

will only need to filter by the `EventID` because the rule used within the configuration file is already doing a majority of the heavy lifting.

Detecting Remote Thread Creation

Syntax: `Get-WinEvent -Path <Path to Log> -FilterXPath '*/System/EventID=8'`

Detecting Remote Threads

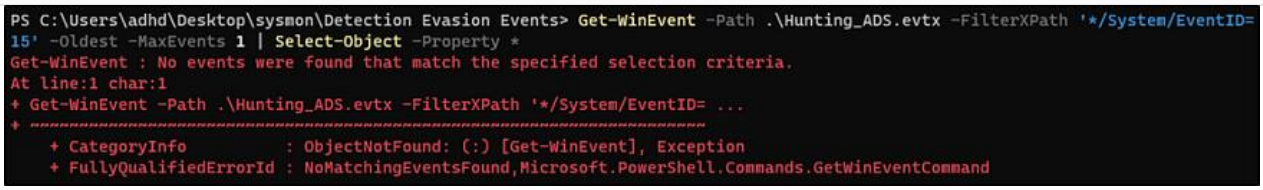
```
Get-WinEvent -Path C:\\Users\\THM-Analyst\\Desktop\\Scenarios\\Practice\\Detecting_RemoteThreads.evtx -FilterXPath '*/System/EventID=8'
```

Answer the questions below

Read the above and practice detecting evasion techniques

Detecting ADS

```
Get-WinEvent -Path .\\Hunting_ADS.evtx -FilterXPath '*/System/EventID="15"' -Oldest -MaxEvents 1 | Select-Object -Property *
```



Detecting Remote Threads

```
Get-WinEvent -Path .\\Detecting_RemoteThreads.evtx -FilterXPath '*/System/EventID="8"' -Oldest -MaxEvents 1 | Select-Object -Property *
```

```
PS C:\Users\adhd\Desktop\sysmon\Detection Evasion Events> Get-WinEvent -Path .\Detecting_RemoteThreads.evtx -FilterXPath '*/Sys
tem/EventID=8' -Oldest -MaxEvents 1 | Select-Object -Property *


Message              : CreateRemoteThread detected:
                       RuleName:
                       UtcTime: 2019-07-03 20:39:30.129
                       SourceProcessGuid: {365abb72-0c16-5d1d-0000-00108b721100}
                       SourceProcessId: 3092
                       SourceImage: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
                       TargetProcessGuid: {365abb72-1256-5d1d-0000-0010fb1a1b00}
                       TargetProcessId: 1632
                       TargetImage: C:\Windows\System32\notepad.exe
                       NewThreadId: 3788
                       StartAddress: 0x00560000
                       StartModule:
                       StartFunction:
Id                   : 8
Version              : 2
Qualifiers           :
Level                : 4
Task                 : 8
Opcode               : 0
Keywords             : -9223372036854775808
RecordId             : 8343
ProviderName         : Microsoft-Windows-Sysmon
ProviderId           : 5770385f-c22a-43e0-bf4c-06f5698ffbd9
LogName              : Microsoft-Windows-Sysmon/Operational
ProcessId            : 112
ThreadId             : 2084
MachineName          : IEWIN7
UserId               : S-1-5-18
TimeCreated          : 7/3/2019 2:39:30 PM
ActivityId           :
RelatedActivityId    :
ContainerLog         : c:\users\adhd\desktop\sysmon\detection evasion events\detecting_remotethreads.evtx
MatchedQueryIds      : {}
Bookmark             : System.Diagnostics.Eventing.Reader.EventBookmark
LevelDisplayName     : Information
OpcodeDisplayName    : Info
TaskDisplayName      : CreateRemoteThread detected (rule: CreateRemoteThread)
KeywordsDisplayNames : {}
Properties           : {System.Diagnostics.Eventing.Reader.EventProperty, System.Diagnostics.Eventing.Reader.EventProperty,
                       System.Diagnostics.Eventing.Reader.EventProperty, System.Diagnostics.Eventing.Reader.EventProperty...}
```

Task 10: Practical Investigations

Event files used within this task have been sourced from the [EVTX-ATTACK-SAMPLES] and [SysmonResources] Github repositories.

You can download the event logs used in this room from this task or you can
open them in the Investigations folder on the provided machine.

Investigation 1 — ugh, BILL THAT'S THE WRONG USB!

In this investigation, your team has received reports that a malicious file was dropped onto a host by a malicious USB. They have pulled the logs suspected and have tasked you with running the investigation for it.

Logs are located in `C:\\Users\\THM-Analyst\\Desktop\\Scenarios\\Investigations\\Investigation-1.evtx`.

Investigation 2 — This isn't an HTML file?

Another suspicious file has appeared in your logs and has managed to execute code masking itself as an HTML file, evading your anti-virus detections. Open the logs and investigate the suspicious file.

Logs are located in `C:\\Users\\THM-Analyst\\Desktop\\Scenarios\\Investigations\\Investigation-2.evtx`.

Investigation 3.1–3.2 — Where's the bouncer when you need him

Your team has informed you that the adversary has managed to set up persistence on your endpoints as they continue to move throughout your network. Find how the adversary managed to gain persistence using logs provided.

Logs are located in `C:\\Users\\THM-Analyst\\Desktop\\Scenarios\\Investigations\\Investigation-3.1.evtx`

and `C:\\Users\\THM-Analyst\\Desktop\\Scenarios\\Investigations\\Investigation-3.2.evtx`.

Investigation 4 — Mom look! I built a botnet!

As the adversary has gained a solid foothold onto your network it has been brought to your attention that they may have been able to set up C2 communications on some of the endpoints. Collect the logs and continue your investigation.

Logs are located in `C:\\Users\\THM-Analyst\\Desktop\\Scenarios\\Investigations\\Investigation-4.evtx`

Answer the questions below

What is the full registry key of the USB device calling svchost.exe in Investigation 1?

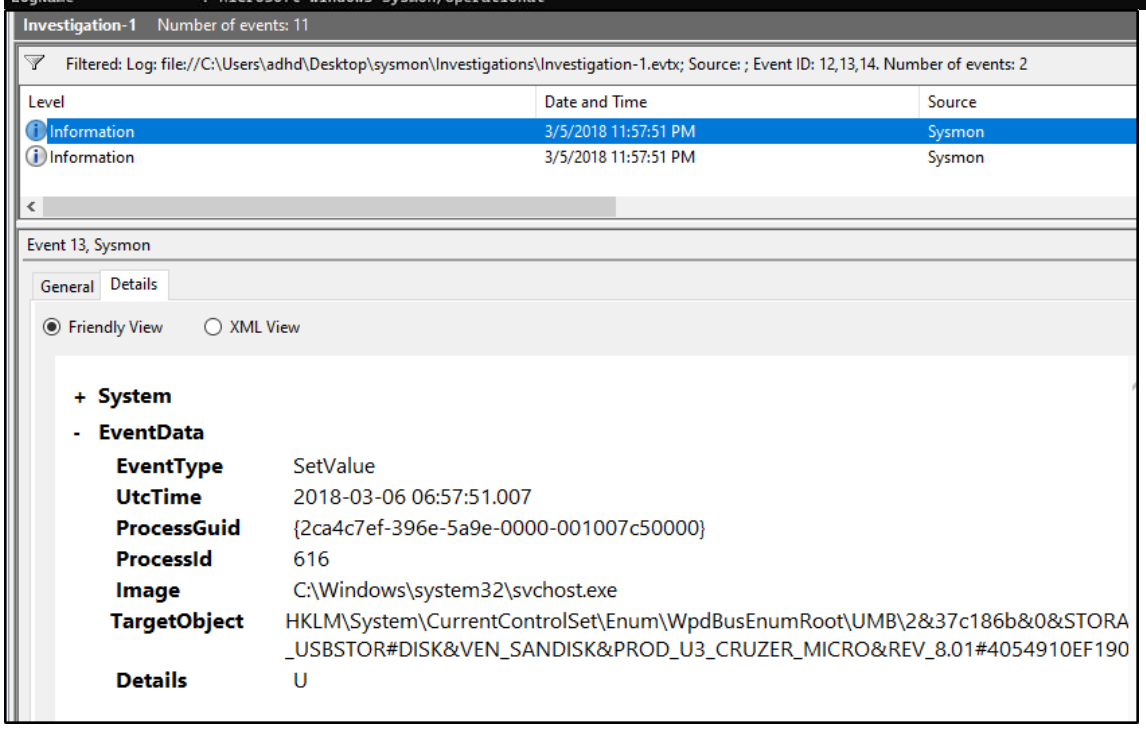
Answer: HKLM\System\CurrentControlSet\Enum\WpdBusEnumRoot\UMB\2&37c186b&0&STORAGE#VOLUME#_??_USBSTOR#DISK&VEN_SANDISK&PROD_U3_CRUZER_MICRO&REV_8.01#4054910EF19005B3&0#\FriendlyName

```
Get-WinEvent -Path .\\Investigation-1.evtx -FilterXPath '*/System/EventID=12 or */System/EventID=13 or */System/EventID=14' -Oldest  | Select-Object -Property *
```

There are two events, but the result is unreliable as it does not show which image is calling. So go to event viewer and filter events there.

This is what we want

```
Message              : Registry value set:
                       RuleName: SetValue
                       EventType: 2018-03-06 06:57:51.007
                       UtcTime: {2ca4c7ef-3bec-5a9e-0000-0010d9070e00}
                       ProcessGuid: 2532
                       ProcessId: 0
                       Image: HKLM\SOFTWARE\Microsoft\Windows Portable Devices\Devices\WPDBUSENUMROOT#UMB#2&37C186B&0&STORAGE#
                       VOLUME#_??_USBSTOR#DISK&VEN_SANDISK&PROD_U3_CRUZER_MICRO&REV_8.01#4054910EF19005B3&0#\FriendlyName
                       TargetObject: U
                       Details: %8
Id                   : 13
Version              : 2
Qualifiers           :
Level                : 4
Task                 : 13
Opcode               : 0
Keywords             : -9223372036854775808
RecordId             : 50
ProviderName         : Microsoft-Windows-Sysmon
ProviderId           : 5770385f-c22a-43e0-bf4c-06f5698ffbd9
LogName              : Microsoft-Windows-Sysmon/Operational
ProcessId            : 1820
ThreadId             : 3144
MachineName          : WIN-7JKBJEGBO38
UserId               : S-1-5-18
TimeCreated          : 3/5/2018 11:57:51 PM
ActivityId           :
RelatedActivityId    :
ContainerLog         : c:\users\adhd\desktop\sysmon\investigations\investigation-1.evtx
MatchedQueryIds      : {}
Bookmark             : System.Diagnostics.Eventing.Reader.EventBookmark
LevelDisplayName     : Information
OpcodeDisplayName    : Info
TaskDisplayName      : Registry value set (rule: RegistryEvent)
KeywordsDisplayNames : {}
Properties           : {System.Diagnostics.Eventing.Reader.EventProperty, System.Diagnostics.Eventing.Reader.EventProperty,
                       System.Diagnostics.Eventing.Reader.EventProperty, System.Diagnostics.Eventing.Reader.EventProperty...}

Message              : Registry value set:
                       RuleName: SetValue
                       EventType: 2018-03-06 06:57:51.007
                       UtcTime: {2ca4c7ef-396e-5a9e-0000-001007c50000}
                       ProcessGuid: 616
                       ProcessId: 0
                       Image: HKLM\System\CurrentControlSet\Enum\WpdBusEnumRoot\UMB\2&37c186b&0&STORAGE#VOLUME#_??_USBSTOR#DIS
                       K&VEN_SANDISK&PROD_U3_CRUZER_MICRO&REV_8.01#4054910EF19005B3&0#\FriendlyName
                       TargetObject: U
                       Details: %8
Id                   : 13
Version              : 2
Qualifiers           :
Level                : 4
Task                 : 13
Opcode               : 0
Keywords             : -9223372036854775808
RecordId             : 51
ProviderName         : Microsoft-Windows-Sysmon
ProviderId           : 5770385f-c22a-43e0-bf4c-06f5698ffbd9
LogName              : Microsoft-Windows-Sysmon/Operational
```



What is the device name when being called by RawAccessRead in Investigation 1?

Answer: \Device\HarddiskVolume3

The EventID for RawAccessRead is 9

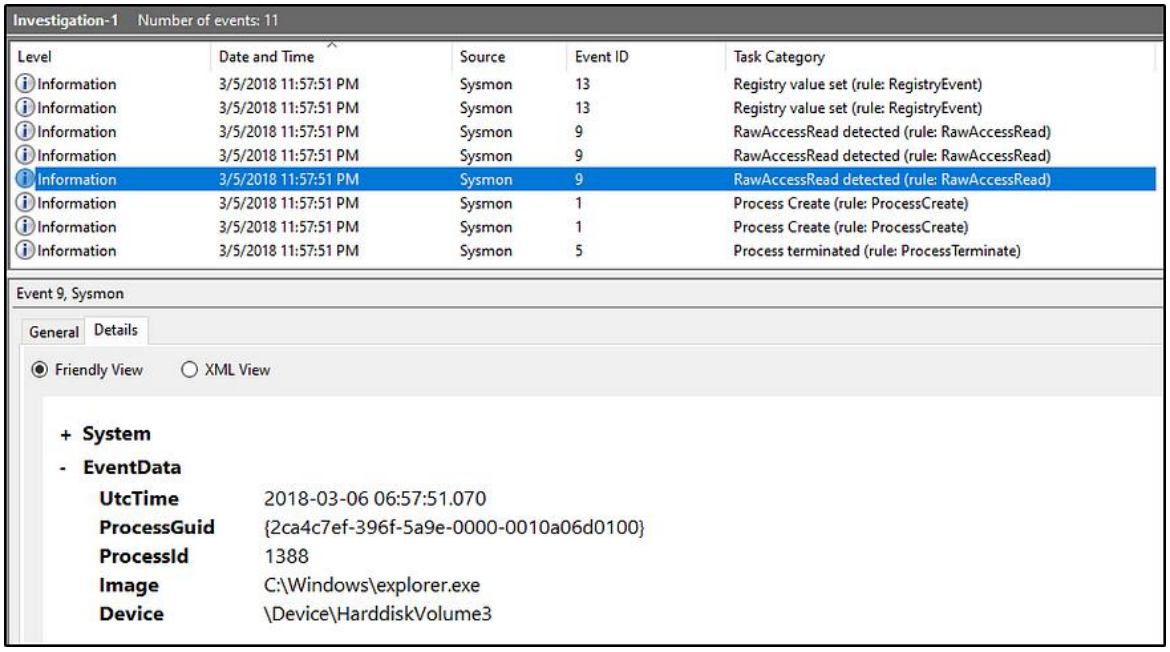
```
Get-WinEvent -Path .\Investigation-1.evtx -FilterXPath '*/System/EventID=9' -Oldest -MaxEvents 1 | Select-Object -Property *
```

What is the first exe the process executes in Investigation 1?

Answer: rundll32.exe

This is the last process for eventID 9, notice that it calls for explorer.exe

It’s a bit hard to analyze using powershell because the information is not detailed. Missing info like parentimage or Parent command line. So we use the eventviewer

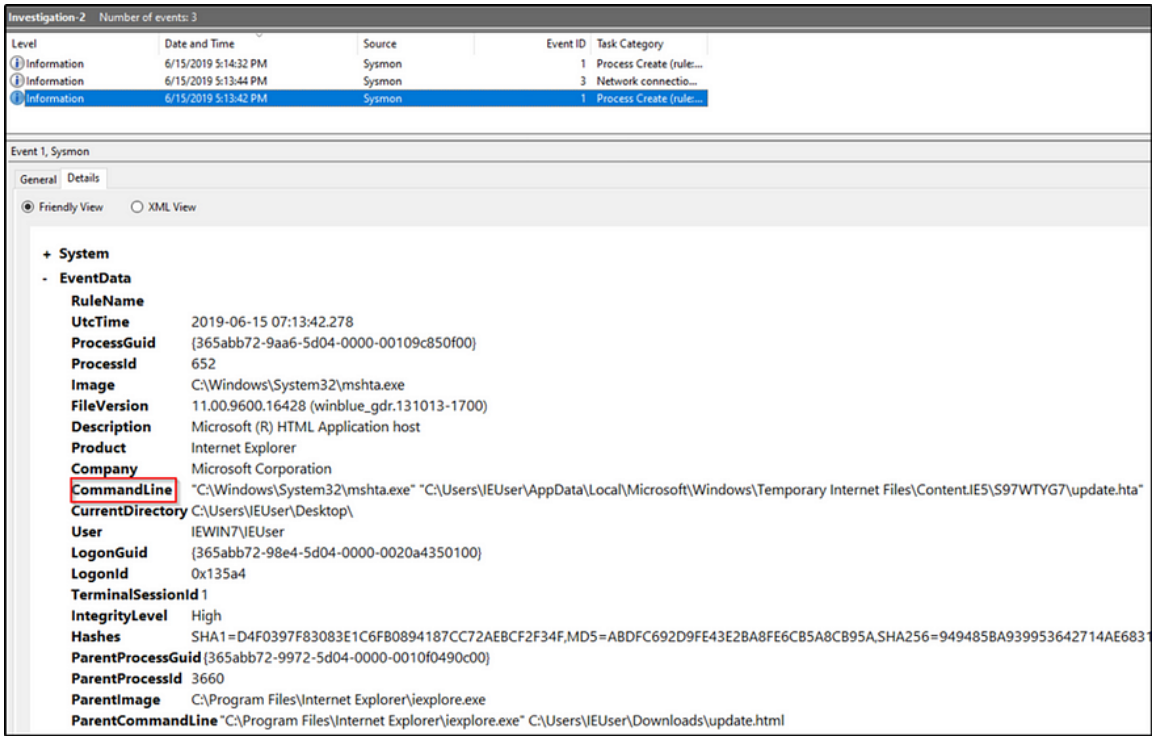


The next process right after it is a process creation. We see that `explorer.exe` is the `ParentCommandLine` and it calls for `rundll32.exe` to be executed. So the first process executed after the usb was inserted is `rundll32.exe`.

What is the full path of the payload in Investigation 2?

Answer: C:\Users\IEUser\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\S97WTYG7\update.hta

The earliest event created a process. The `CommandLine` tag would tell us what was the payload that was executed to trigger the event.



What is the full path of the file the payload masked itself as in Investigation 2?

Answer: C:\Users\IEUser\Downloads\update.html

`C:\Users\IEUser\Downloads\update.html`, is masked as if not malicious. But its executions triggers the payload to be executed.

What signed binary executed the payload in Investigation 2?

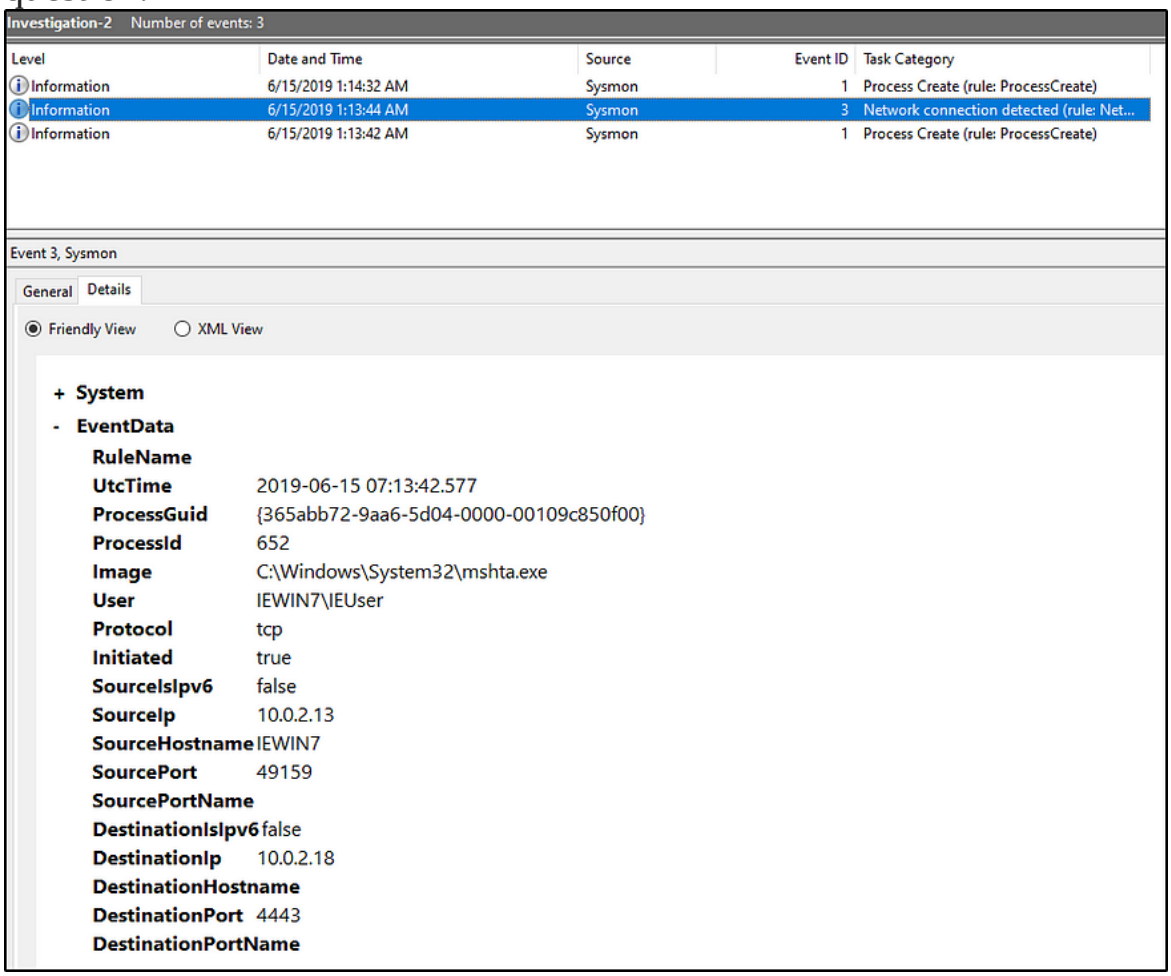
Answer: C:\Windows\System32\mshta.exe

Once `update.html` is executed, `mshta.exe` will now execute the payload `update.hta`.

What is the IP of the adversary in Investigation 2?

Answer: 10.0.2.18

We focus on events with `Network Connection Detection` category. We would see then the IP of the adversary. The `Details` tab would answer the succeeding question.



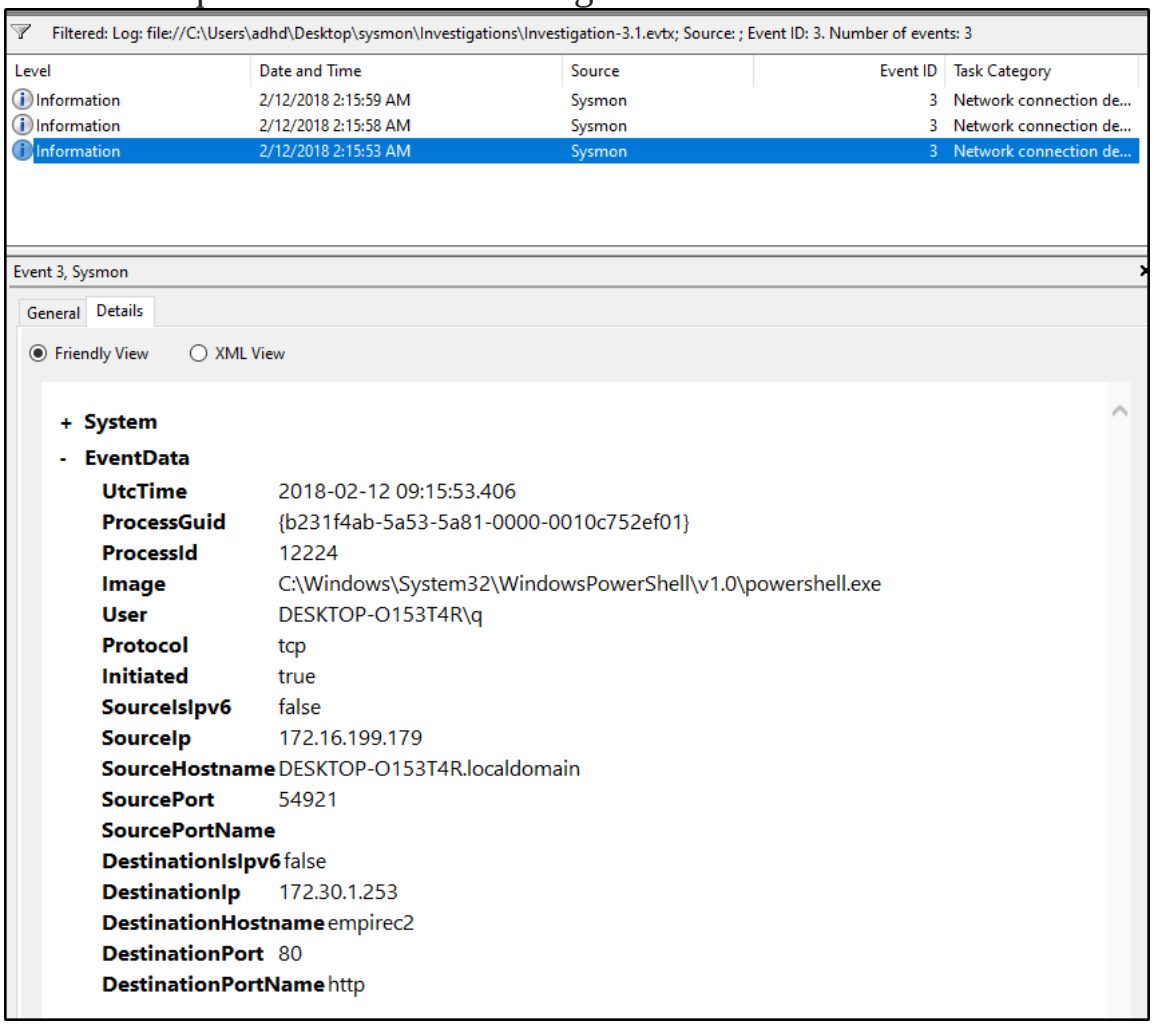
What back connect port is used in Investigation 2?

Answer: 4443

What is the IP of the suspected adversary in Investigation 3.1?

Answer: 172.30.1.253

Let’s filter events with EventID of 3. The `Details` of the first event would answer the questions for this investigation.



What is the hostname of the affected endpoint in Investigation 3.1?

Answer: DESKTOP-O153T4R

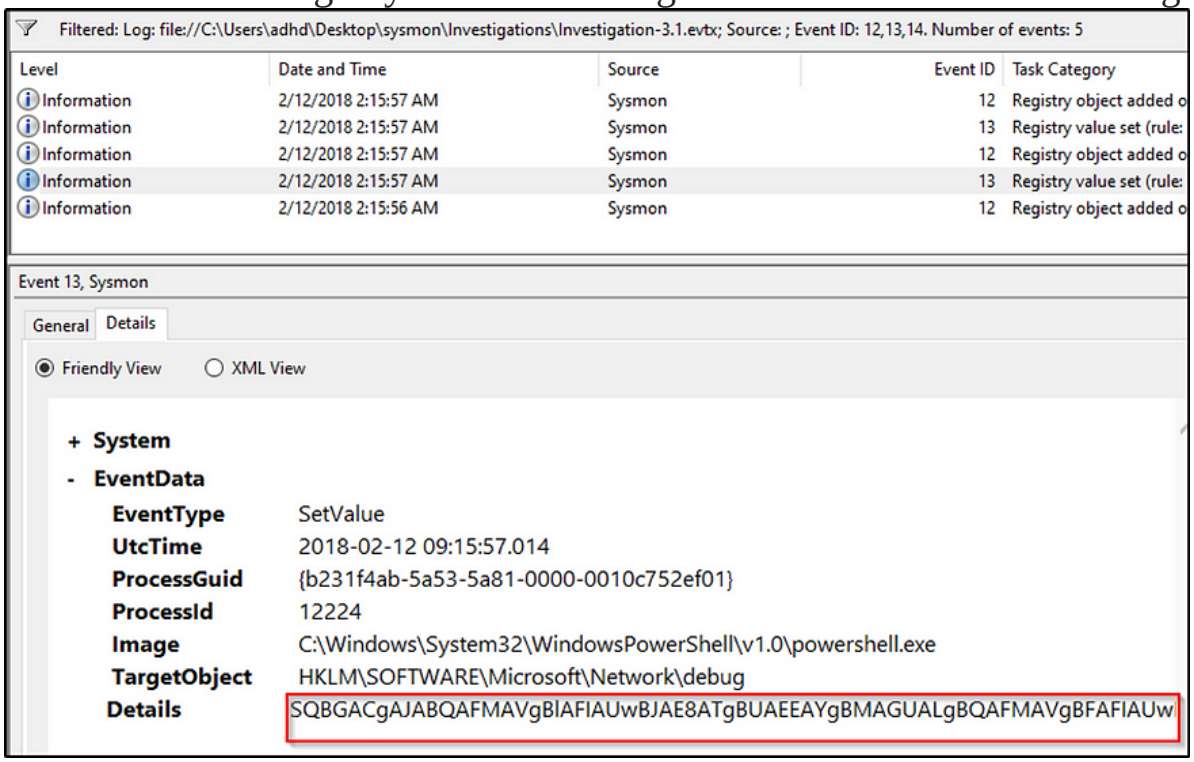
What is the hostname of the C2 server connecting to the endpoint in Investigation 3.1?

Answer: empirec2

Where in the registry was the payload stored in Investigation 3.1?

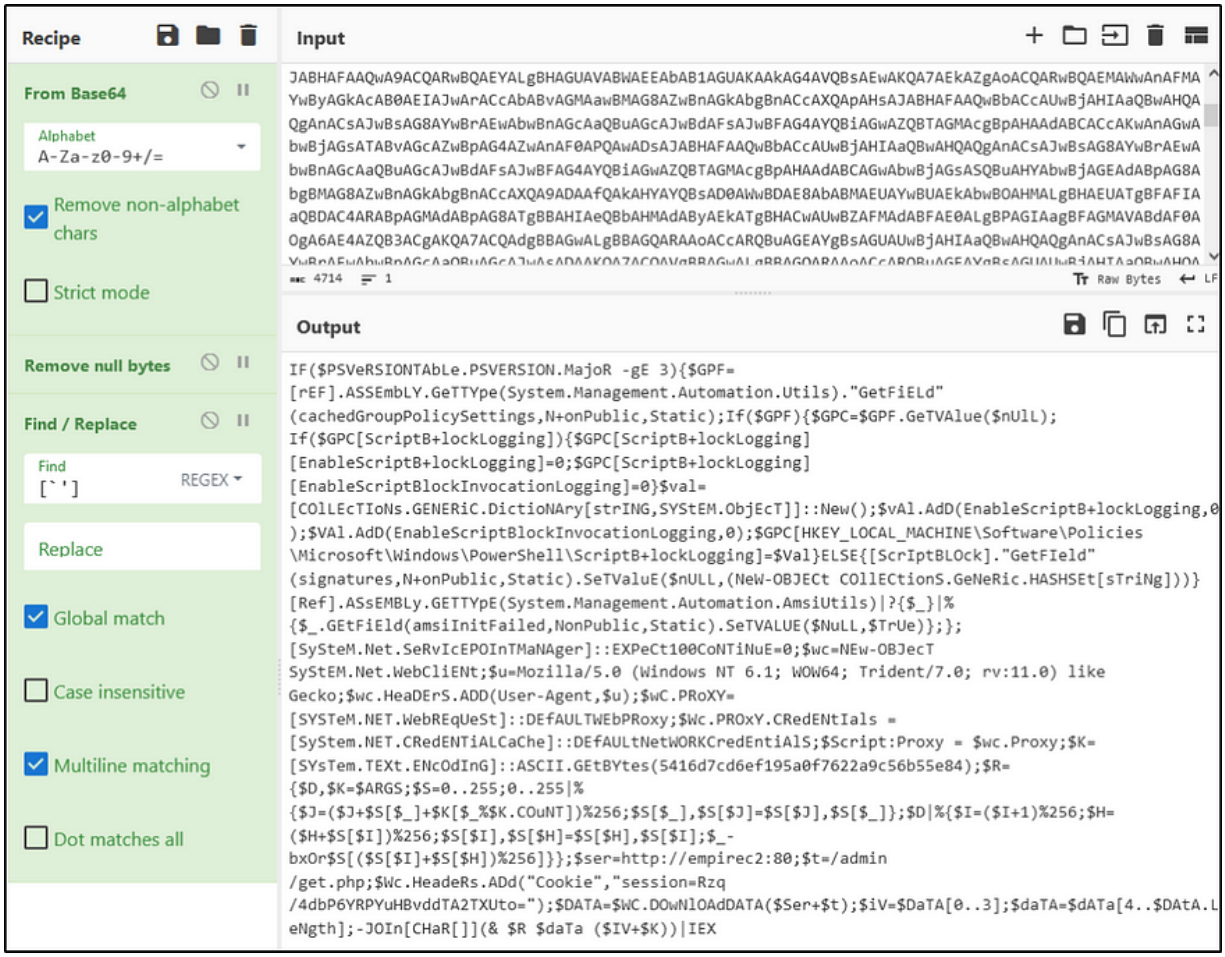
Answer: HKLM\SOFTWARE\Microsoft\Network\debug

Filter events with EventIDs 12,13, and 14. With the second event filtered, we can see that a registry value was changed to an obfuscated set of strings.



If we wanted to know what it is, paste the values in cyberchef and decode. It would look something like this.

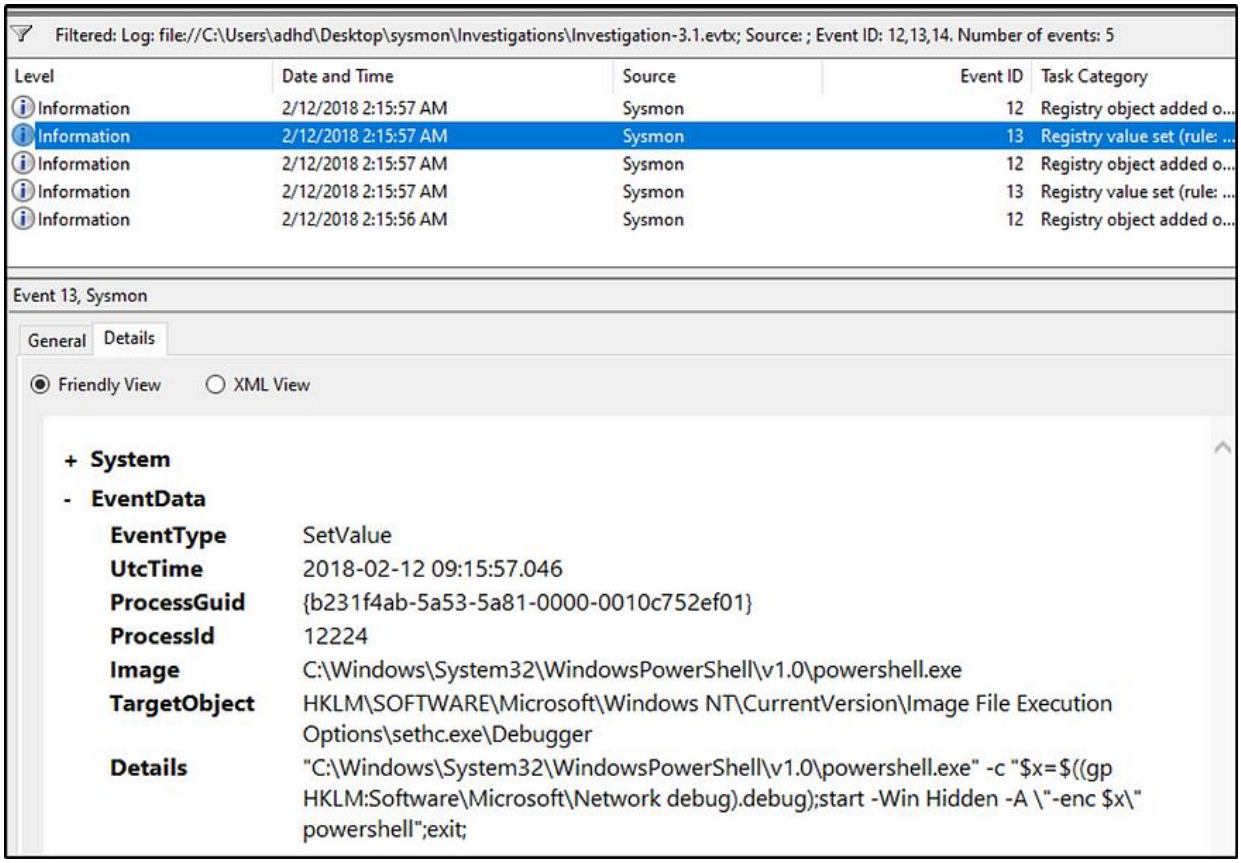
The attacker is basicaly trying to disable ScriptBlockLogging.



What PowerShell launch code was used to launch the payload in Investigation 3.1?

Answer:"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -c "$x=$((gp HKLM:Software\Microsoft\Network debug).debug);start -Win Hidden -A \"-enc $x\" powershell";exit;

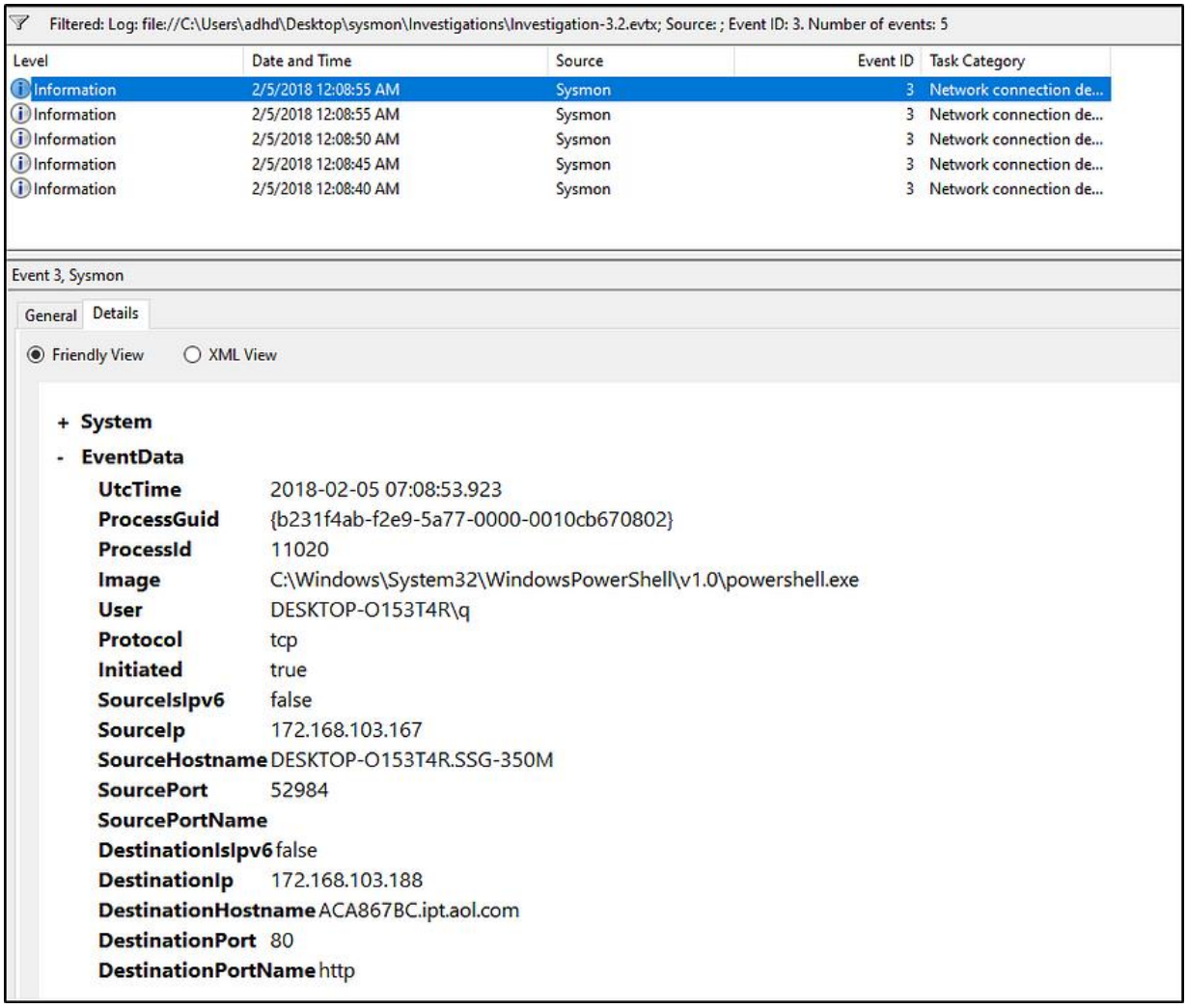
The second event ID of 13 contains the code that would launch for the payload in the registry, whose values were modified, to be executed.



What is the IP of the adversary in Investigation 3.2?

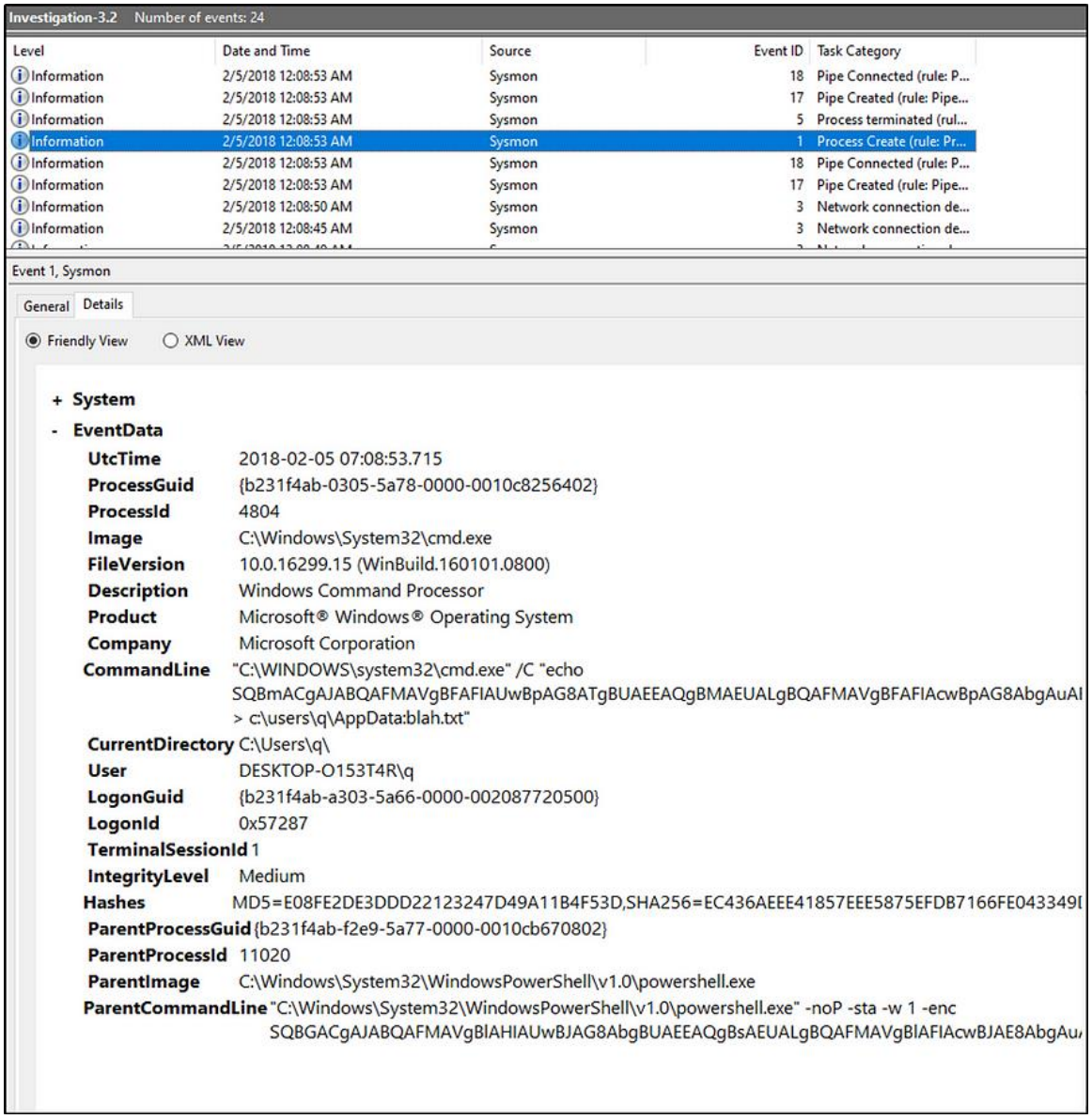
Answer:172.168.103.188

Filter events with EventID of 3 and we will locate the IP of the attacker.



What is the full path of the payload location in Investigation 3.2?

Answer:c:\users\q\AppData:blah.txt



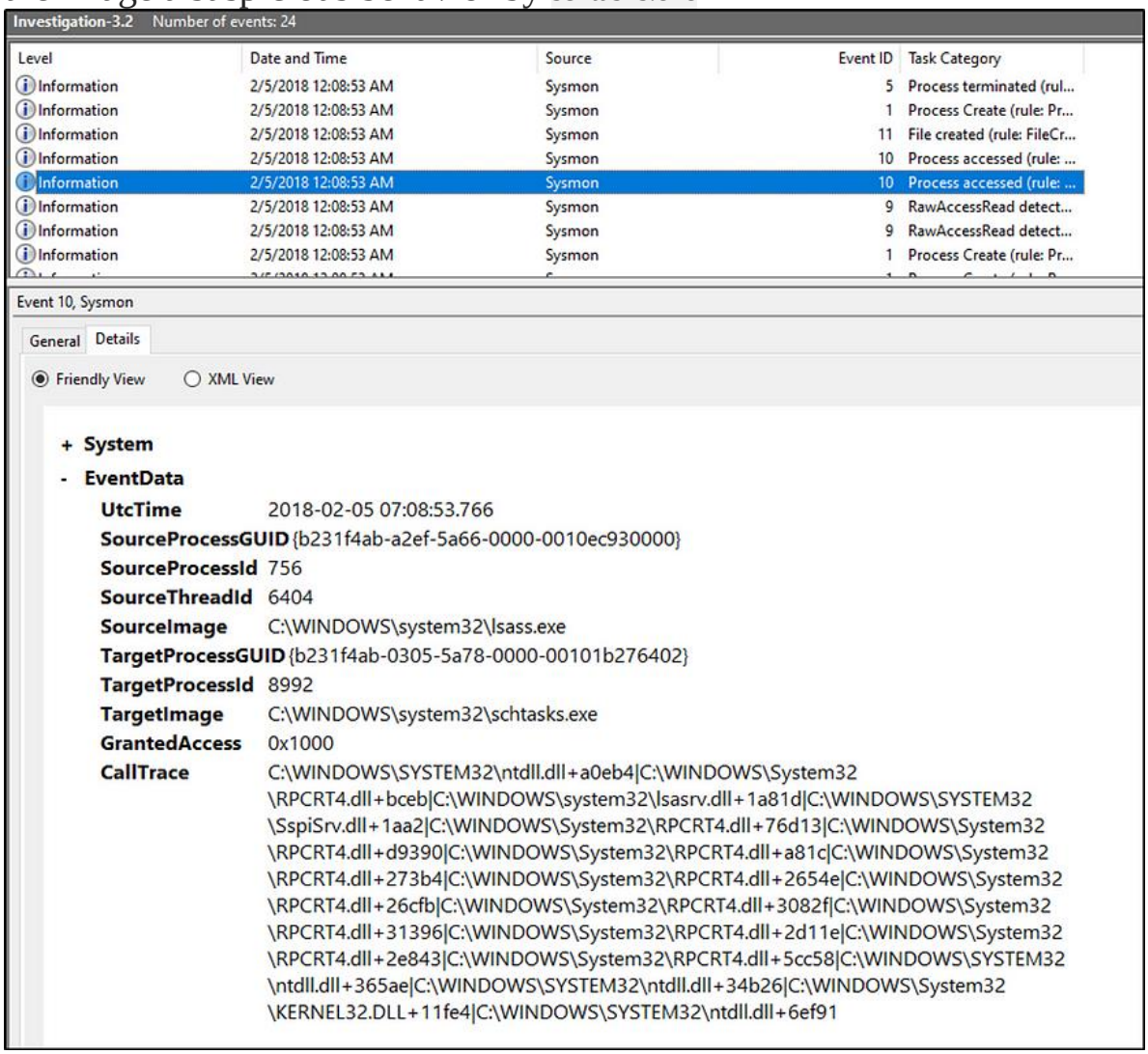
What was the full command used to create the scheduled task in Investigation 3.2?

Answer: “C:\WINDOWS\system32\schtasks.exe” /Create /F /SC DAILY /ST 09:00 /TN Updater /TR “C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -NonI -W hidden -c \”IEX ([Text.Encoding]::UNICODE.GetString([Convert]::FromBase64String($(cmd /c ‘’more < c:\users\q\AppData:blah.txt’’’)))))\””

What process was accessed by schtasks.exe that would be considered suspicious behavior in Investigation 3.2?

Answer: lsass.exe

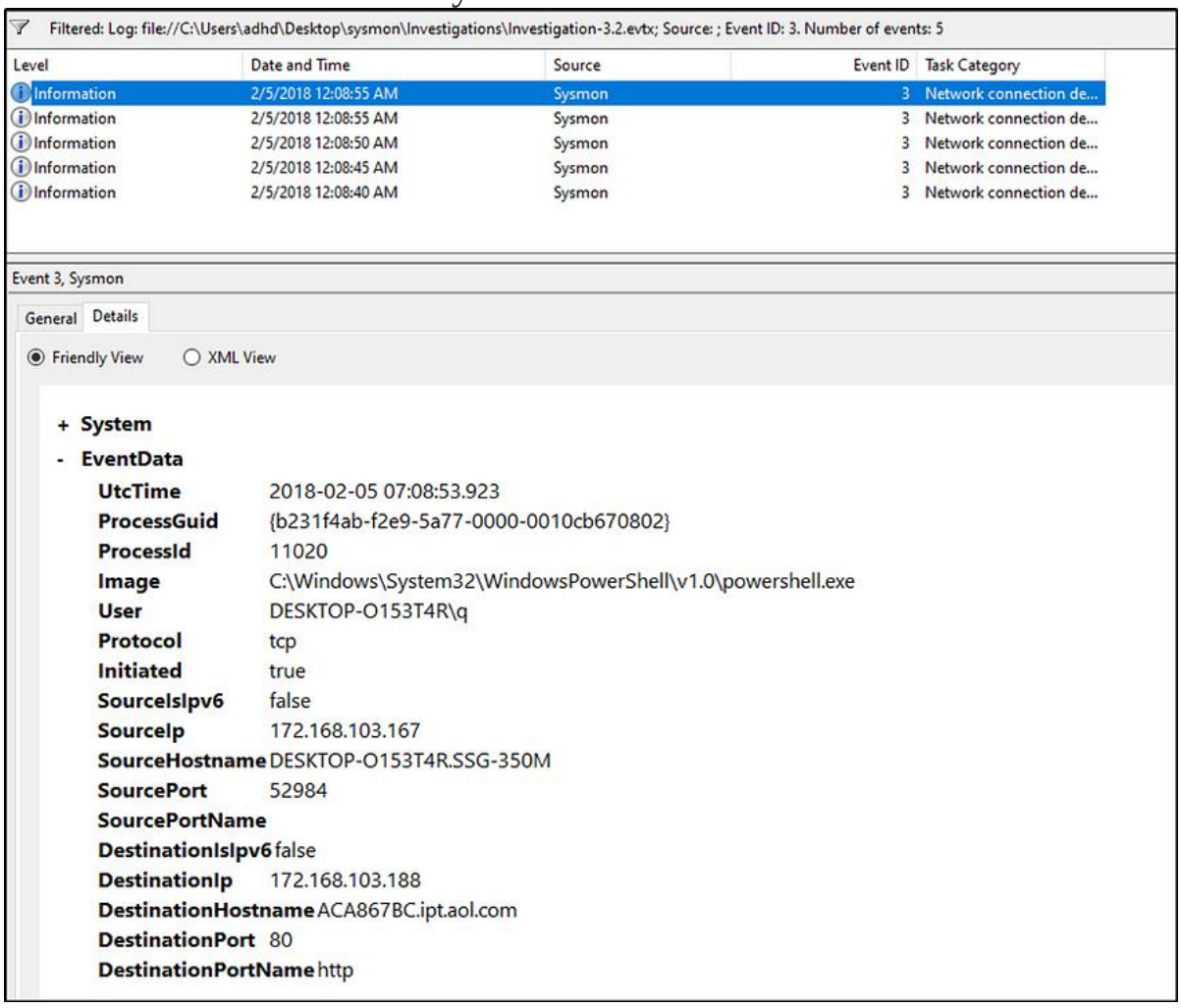
If we follow the timeline after the payload was executed, we can observe in the image a suspicious behavior by `schtasks.exe`



What is the IP of the adversary in Investigation 4?

Answer:172.30.1.253

Investigate the events. We would see the destination IP, port number, and the hostname of the adversary.



What port is the adversary operating on in Investigation 4?

Answer: 80

What C2 is the adversary utilizing in Investigation 4?

Answer: empire