Tactical Detection

Introduction

You're hired as a security engineer, and you want to make a good impression. You noticed that there's a default ruleset available, and it has already been enabled. The SOC team seems to function, albeit not as efficiently as you might expect - then it dawns on you; the default rules just won't cut it.

This scenario is not uncommon - in fact, a common pitfall of modern SOCs today is leaning too much on default rules and settings of the products they deploy, leaving them with security alerts that don't really add value to their security posture.

Learning Objectives:
In this room, we will strive to understand the mindset behind choosing a tactical approach in alerting and detecting threats, IOAs, IOCs, etc. In the process, we will gain practical experience in setting up a basic tactical detection capability leveraging techniques used in real-life environments.

Room Prerequisites and Expectation Setting:
There are no hard prerequisites in order to gain value from this room; however, it would be very helpful to have a basic understanding of navigating cmd and executing basic commands, as well as navigating FullEventLogView as it would be our main tool in reviewing Event Logs.

This room will touch upon some of the most efficient ways to bolster an organization's security posture by leveraging detection mechanisms and walking the user through setting them up tactically. This should serve as a baseline where the user will be able to learn the basics, implement them in their functions, and make them truly their own.

Unique Threat Intel

You stumbled upon documentation of a previous incident containing a couple of unique Indicators of Compromise (IOCs)

Unique IOCs of previous intrusions are good examples of Threat Intel as they're traces of the specific adversary that your environment has already faced. The inclusion of these IOCs in your detection mechanism will help spot re-intrusion of that specific adversary immediately, among others.

File Name	Path	File Size	Description
bad3xe69.exe	C:\Downloads\bad3xe69.exe	13619KB	Downloaded from the internet. Posing as an installer for an application that needs an update.

DNS/IP Address	Description
bad3xe69connection.io	Domain of the source of the malicious executable bad3xe69.exe
kind4bad.com	Suspicious domain related to bad3xe69connection.io
nic3connection.io	Suspicious domain related to bad3xe69connection.io

The screenshots above are excerpts from a spreadsheet that contains IOCs that can be integrated with the organization's current detection mechanism. It's more or less the same format that Incident Responders use as they go through their investigation. Logging in IOCs in a file like this allows for better collaboration among multiple incident responders. It also makes scoping of the incident more effective - more often than not, IOCs lead to more IOCs.

In the spreadsheet excerpt above, based on the description, the direct indicator found by the authors of the documentation is actually just the bad3xe69connection[.]io; however, upon further inspection of the malicious domain, they were able to conclude that two other malicious domains should be recorded as IOCs due to their association with the original malicious domain.

To maximize our efficiency, we will transform these IOCs into detection rules in a vendor-agnostic format using Sigma.

Sigma is an open-source generic signature language developed to describe log events in a structured format. This allows for quick sharing of detection methods by security analysts.

A basic example of how it can be written into a functional Sigma rule is as follows.

baddomains.yml

```
title: Executable Download from Suspicious Domains
status: test
description: Detects download of executable types from hosts found in the IOC spreadsheet
author: Mokmokmok
date: 2022/08/23
modified: 2022/08/23
logsource:
  category: proxy
detection:
  selection:
    c-uri-extension:
      - 'exe'
    r-dns:
      - 'bad3xe69connection.io'
      - 'kind4bad.com'
      - 'nic3connection.io'
  condition: selection
fields:
  - c-uri
falsepositives:
  - Unkown
level: medium
```

The Sigma rule that we came up with from the IOCs presented above is very simple and straightforward, yet the additional layer of detection that it gives the organization is invaluable. In the grander scheme of things, these layers work together to give your analysts the visibility that they need to spot bad actors before it's too late.

Remember that the bad guys need to circumvent all our defenses in order to get to their objectives, but we only need them to fail one layer of detection to have an idea that they're there.

****

Answer the questions below:

What did we use to transform IOCs as detection rules in a vendor-agnostic format?
Answer: Sigma

What is the original indicator found by the authors of the documentation? Write it as written in the spreadsheet.
Answer: bad3xe69connection.io

What is the full file path of the malicious file downloaded from the internet?
Answer: C:\Downloads\bad3xe69.exe

In the Sigma Rule baddomains.yml, what is the logsource category used by the author?
Answer: proxy

Publicly Generated IOCs

You're feeling proud of yourself for being able to implement detection rules that may have an immediate impact on the organization when suddenly, news broke out of a new 0-day vulnerability. Upon taking a closer look at it, you realize that your organization is directly susceptible to this vulnerability.

You don't have to be able to experience everything in order to learn from something - you can learn from other people's experiences or research or learnings. Analogous to that is the array of research being done by the community, and almost always, they release public IOCs. These public IOCs are then transformed into usable mechanisms to detect bad things in the environment.

Going back to our previous task, we've leveraged Sigma to transform unique IOCs into a product-agnostic form that we can use regardless of our SIEM choice. As this technique shows great promise to the community, there are a number of nice repositories that contain user-submitted Sigma rules that anyone can use. You can plug one directly into your SIEM for immediate value, or further edit it to fit your environment and add even more value to your security posture.

The following is a nice exercise: Write a detection rule for these two / transform these publicly generated IOCs into usable alerts for use in the Elastic Stack and Splunk. We will do the first one together, while you can do the rest on your own. For our purposes, we will be using Uncoder to help with the transformation of these sigma rules. Uncoder is a nice tool that helps convert sigma rules to queries that can be immediately used within a SIEM of your choice.

A fairly recent 0-day vulnerability, Follina-MSDT, has a publicly available sigma rule developed by huntress's Matthew Brennan:

Follina-MSDT Sigma Rule

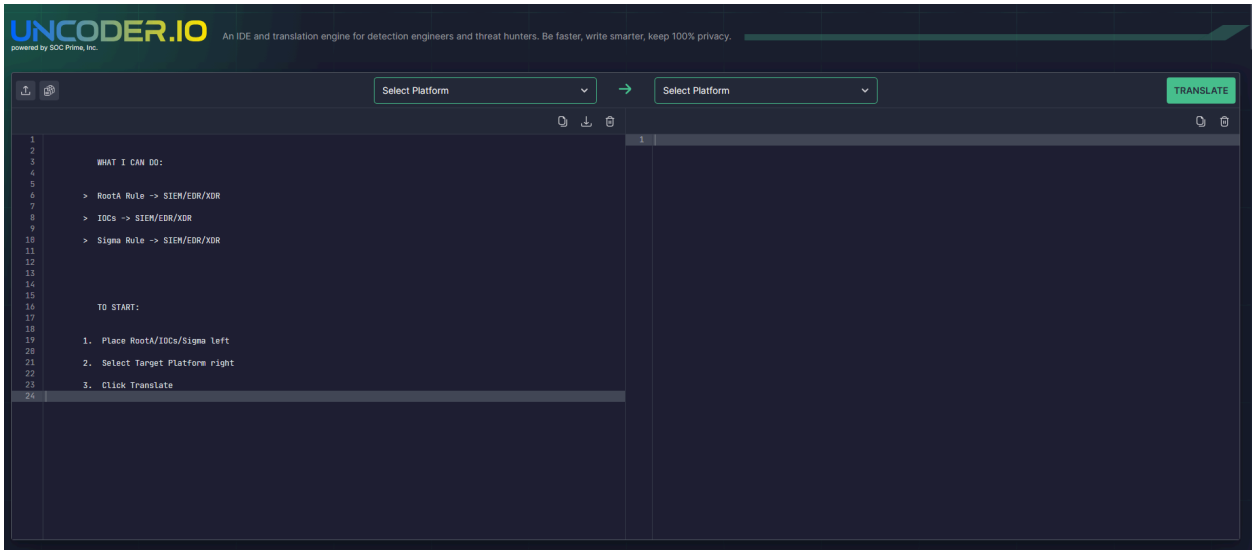
```
title: Suspicious msdt.exe execution - Office Exploit
id: 97a80ed7-1f3f-4d05-9ef4-65760e634f6b
status: experimental
description: This rule will monitor suspicious arguments passed to the msdt.exe process. These arguments are an indicator of recent Offic
references:
    - https://doublepulsar.com/follina-a-microsoft-office-code-execution-vulnerability-1a47fce5629e
    - https://twitter.com/MalwareJake/status/1531019243411623939
author: 'Matthew Brennan'
tags:
    - attack.execution
logsource:
    category: process_creation
    product: windows
detection:

    selection1:
      Image|endswith:
        - 'msdt.exe'
    selection2:
      CommandLine|contains:
        - 'PCWDiagnostic'
    selection3:
      CommandLine|contains:
        - 'ms-msdt:-id'
        - 'ms-msdt:/id'

    selection4:
      CommandLine|contains:
        - 'invoke'
    condition: selection1 and (selection4 or (selection2 and selection3))
falsepositives:
  - Unknown
level: high
```

Another 0-day vulnerability that made waves this past year, log4j, has multiple publicly available sigma rules. One such rule can detect suspicious shells spawned from a Java host process, written by Andreas Hunkeler and Florian Roth:

Log4j Suspicious Shells Sigma Rule

```
title: Suspicious Shells Spawned by Java
id: 0d34ed8b-1c12-4ff2-828c-16fc860b766d
description: Detects suspicious shell spawned from Java host process (e.g. log4j exploitation)
status: experimental
author: Andreas Hunkeler (@Karneades), Florian Roth
date: 2021/12/17
modified: 2022/08/02
tags:
    - attack.initial_access
    - attack.persistence
    - attack.privilege_escalation
logsource:
    category: process_creation
    product: windows
detection:
    selection:
        ParentImage|endswith: '\java.exe'
        Image|endswith:
            - '\sh.exe'
            - '\bash.exe'
            - '\powershell.exe'
            - '\pwsh.exe'
            - '\schtasks.exe'
            - '\certutil.exe'
            - '\whoami.exe'
            - '\bitsadmin.exe'
            - '\wscript.exe'
            - '\cscript.exe'
            - '\scrcons.exe'
            - '\regsvr32.exe'
            - '\hh.exe'
            - '\wmic.exe'        # https://app.any.run/tasks/c903e9c8-0350-440c-8688-3881b556b8e0/
            - '\mshta.exe'
            - '\rundll32.exe'
            - '\forfiles.exe'
            - '\scriptrunner.exe'
            - '\mftrace.exe'
            - '\AppVLP.exe'
            - '\curl.exe'
    condition: selection
falsepositives:
    - Legitimate calls to system binaries
    - Company specific internal usage
level: high
```

Upon navigating to Uncoder, you will immediately see two text boxes, as shown below:



Make sure that on the left side, the Sigma tab is selected as shown above. Copy the Follina-MSDT Sigma Rule contents and then paste it in the left text box. Click on the downward arrow and select ElastAlert. Click on the Translate button when you're ready.

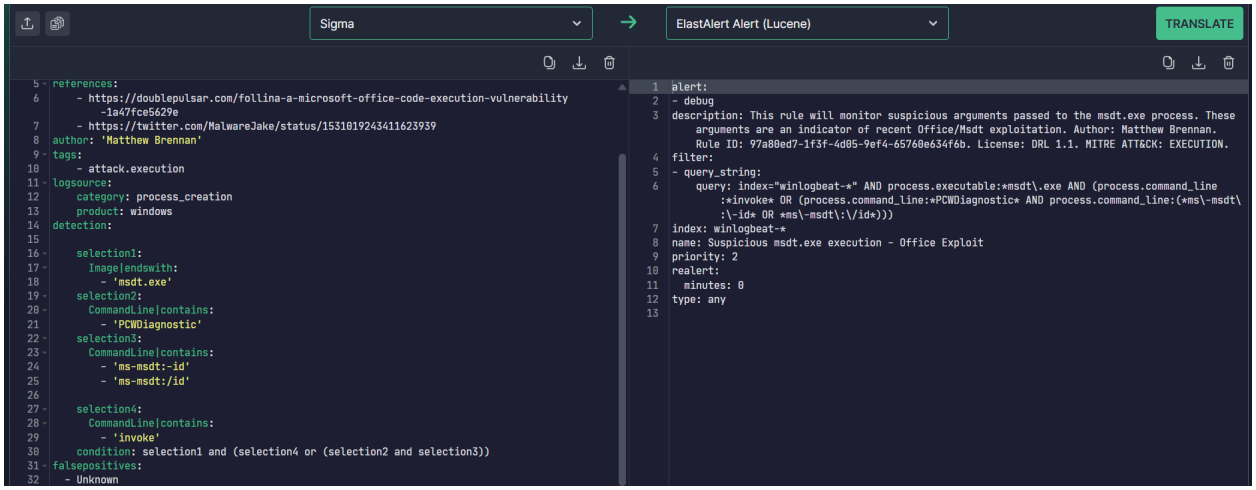
Upon clicking Translate, it shouldn't take long before the results come out of the right text box.

It is important to note that there's no guarantee that the transformed Sigma rules will work perfectly straight out of Uncoder. In order to be production ready, you need to do a lot of testing and fine-tuning. What Uncoder essentially offers is a generic blueprint - it is up to the user to further improve upon it.

****

Answer the questions below:

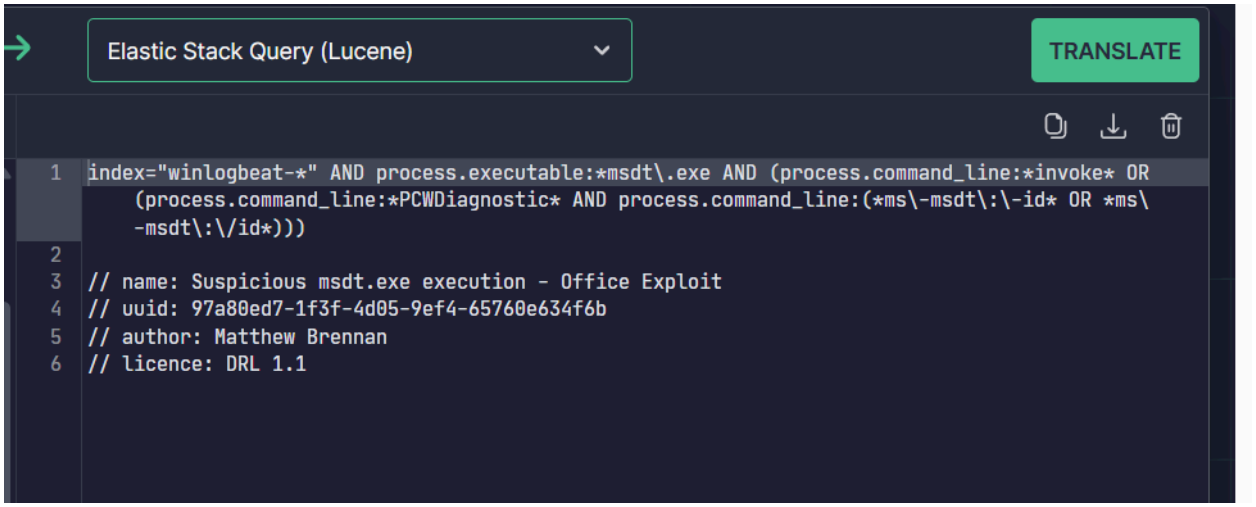
Upon translating the Follina Sigma Rule, what is the index name that the rule will be using, as shown in the output?



Answer: winlogbeat-*

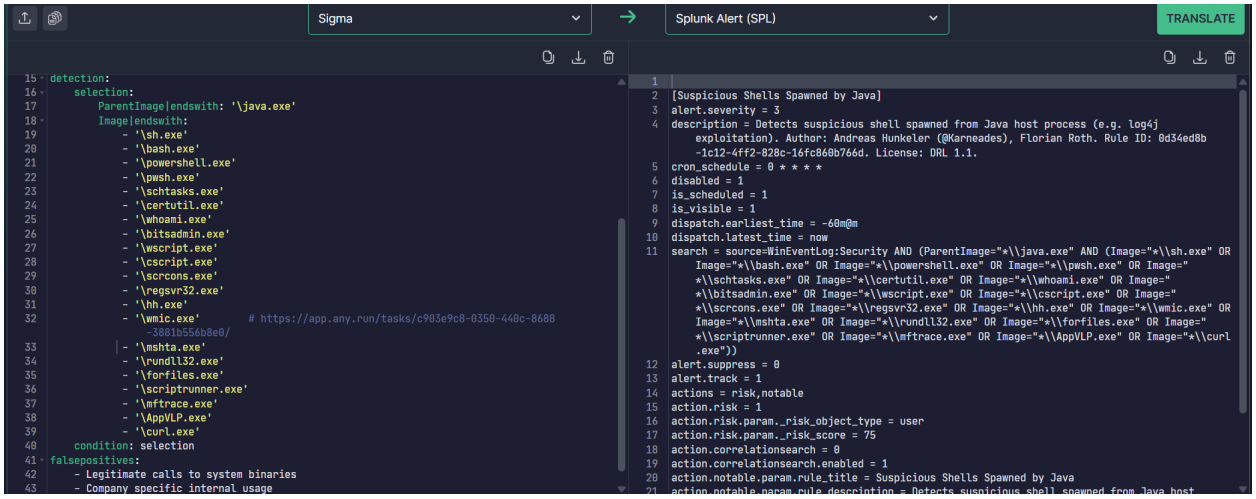
What is the Alerter subclass, as shown in the output?
Answer: debug

Change the Uncoder output to Elastic Stack Query (Lucene).
Which part of the ElastAlert output looks exactly like the Elastic Query?



This query matches the one found under filter on the ElastiAlert.
Answer: filter

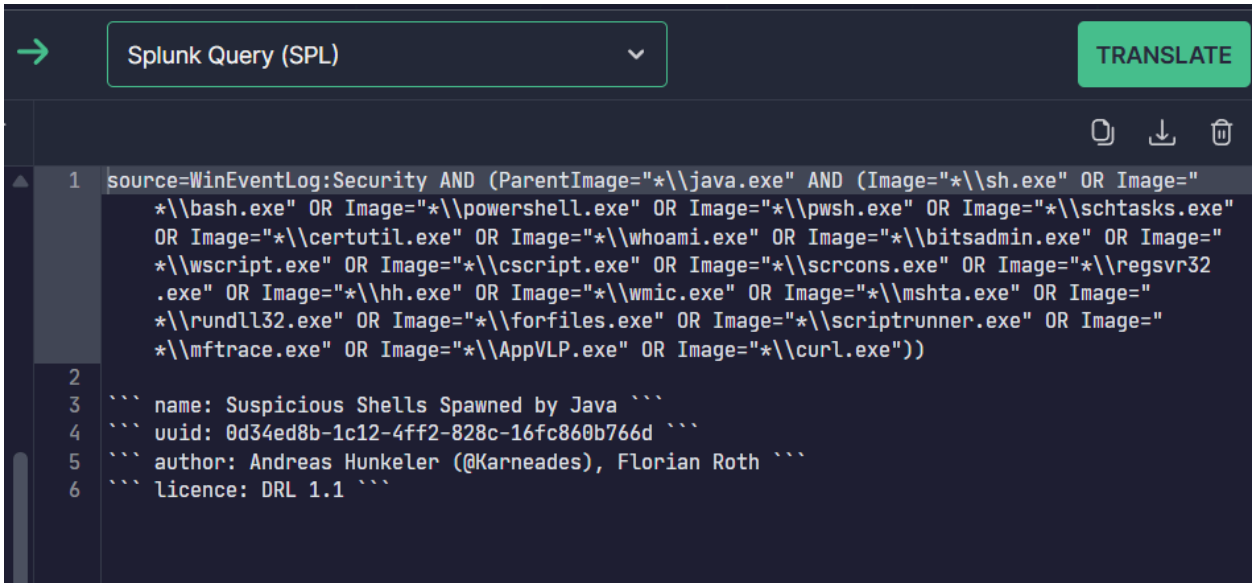
Translate the Log4j Sigma Rule into a Splunk Alert (SPL).
What is the alert severity, as shown in the output?



Answer: 3

What is the dispatch.earliest_time value, as shown in the output?
Answer: -60m@m

Change the Uncoder output to Splunk Query (SPL).
What is the source, as shown in the output?



Answer: WinEventLog:Security

Leveraging “Know Your Environment”: Tripwires

One way to create immediate impact is to leverage very specific data to your advantage. Depending on the environment/organization, more often than not, there

exist pieces of data that not everyone is entitled to have access to. Usually, controls are set by the IT team to limit these accesses.

For example, suppose there are ultra-sensitive files your organization intends to keep secret, such as a hidden treasure map. In that case, you can set alerts for instances that the said map has been accessed, edited, and deleted among other things, and then filter out the ones allowed access to make detections more actionable.

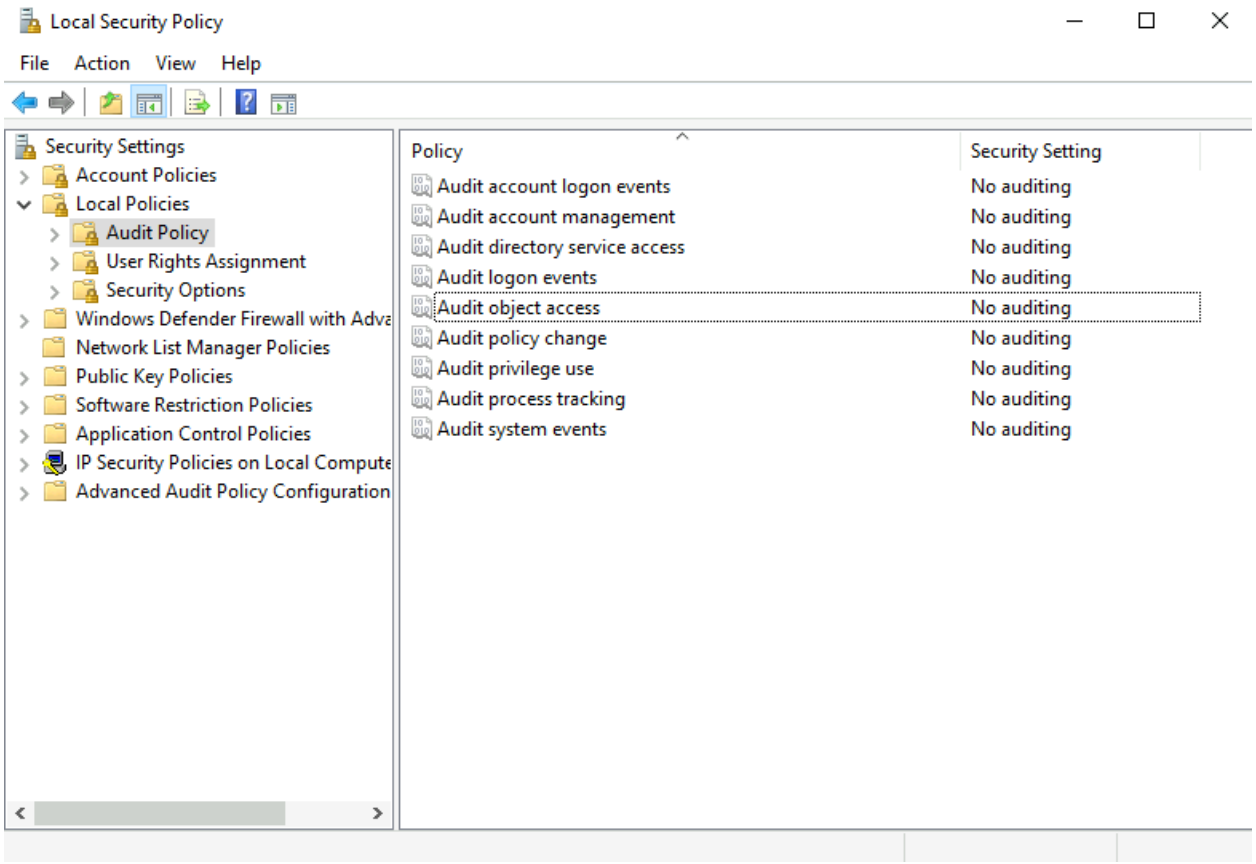
Tripwires are a great way to supplement the detection mechanisms that you already have in place. It's a way to cover "unknown unknowns" and even study an adversary. The most common tripwires are Honeypots and Hidden Files and Folders.

The way honeypots work is that they serve no legitimate business purpose, so any activity concerning them should raise immediate red flags. Hidden files and folders, on the other hand, are not meant to even be seen by normal end users, and so it works best when dealing with crawlers like worms making them particularly effective to detect intrusions.

Setting up a Basic Tripwire:
Click on the Start Machine button at the top right corner of this task. The machine will be available on your web browser, in split-screen view. In case the VM is not visible, use the blue Show Split View button at the top-right of the page. You may use the following credentials if you prefer accessing the machine via RDP:

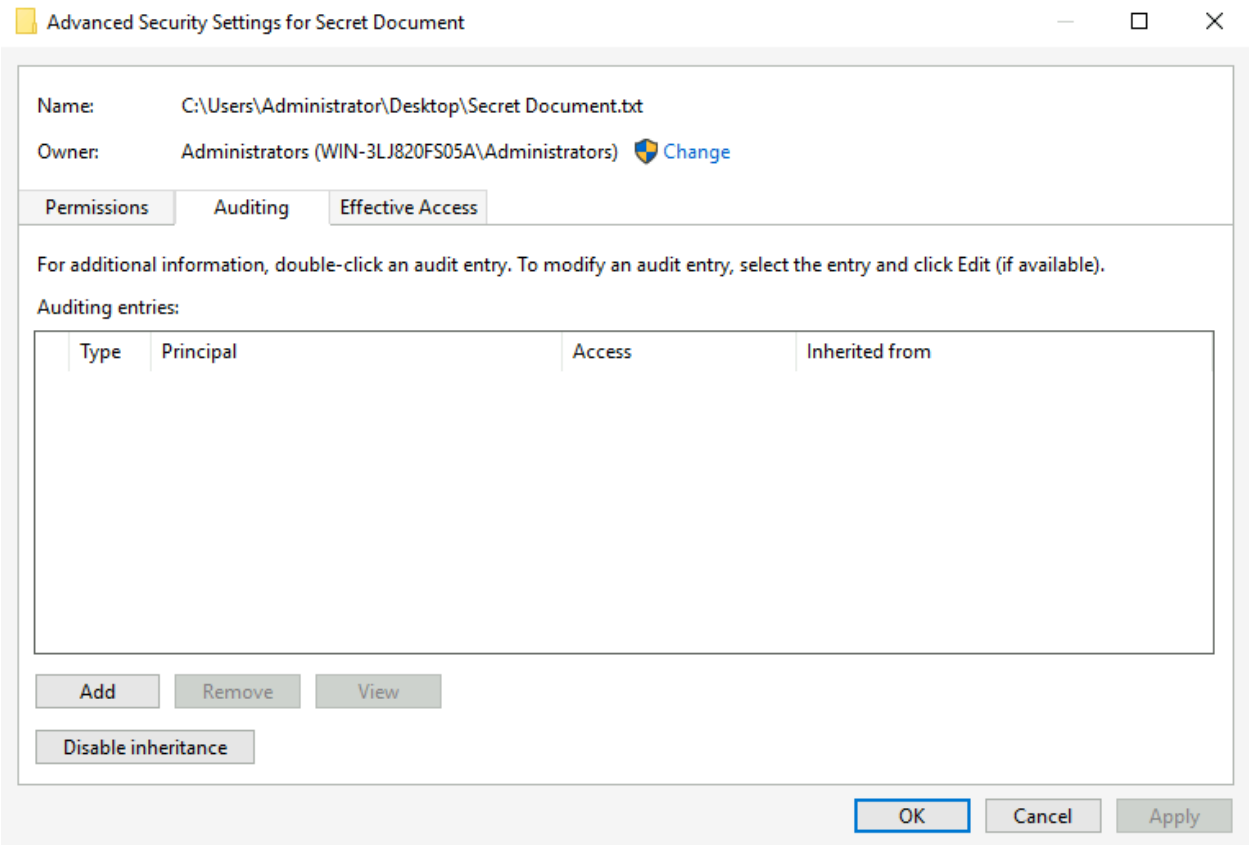
Machine IP: 10.10.92.46
User: Administrator
Password: Follina_0438

Once the machine has initiated, click on the start icon and type "Local". The Local Security Policy application should appear. Open the application, then navigate to Security Settings → Local Policies → Audit Policy.

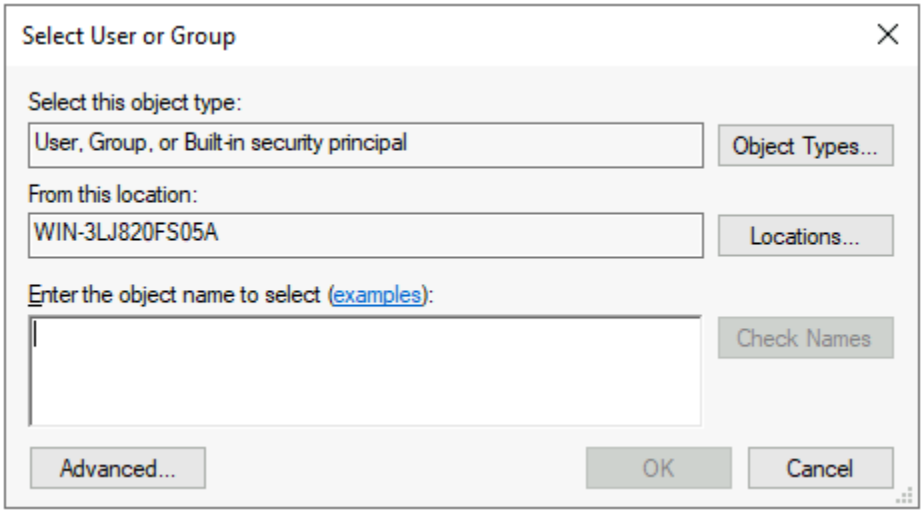


Open the Audit object access policy, tick the boxes beside Success and Failure, click Apply, and finally click OK. This entails that all access attempts will be logged, regardless of whether it succeeded. Once you're done, you may proceed to close the Local Security Policy application.

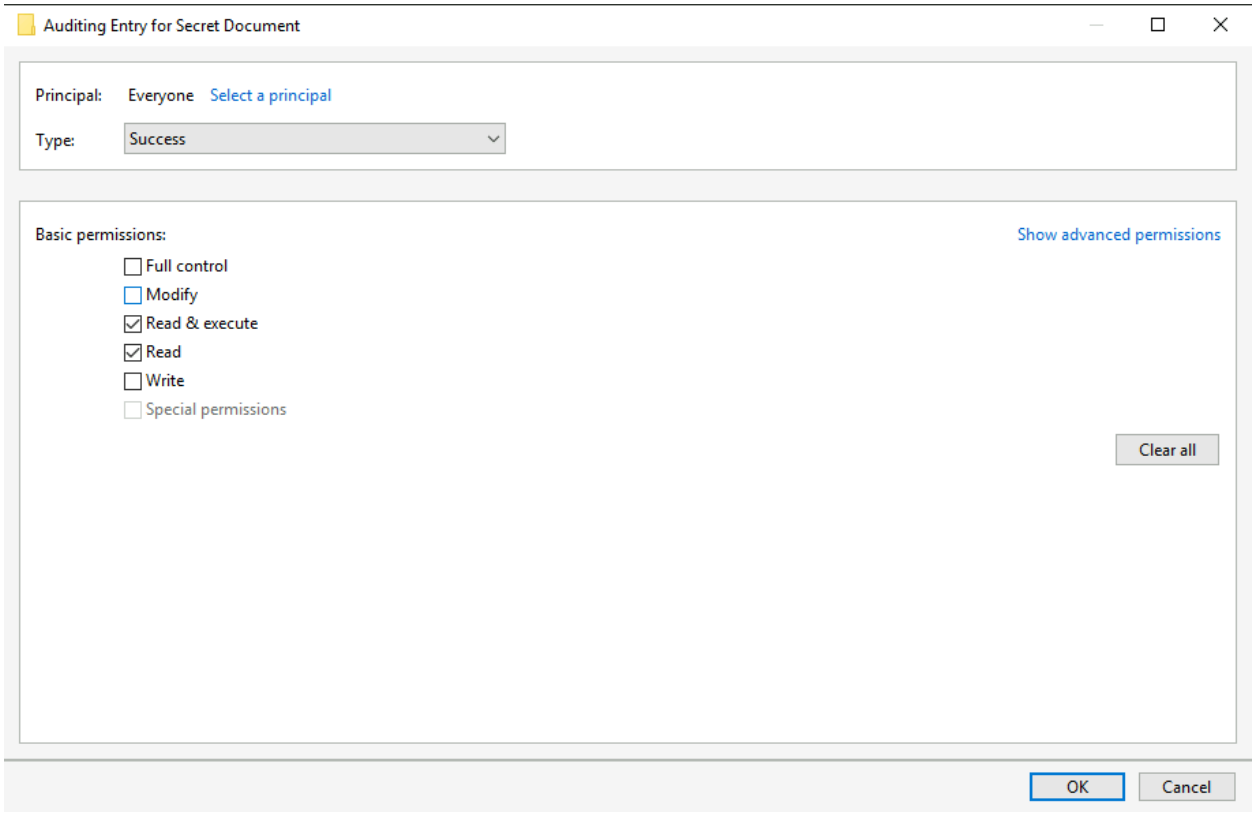
After configuring the Local Security Policy, nothing new will immediately happen by default. We have to specify in the actual file or folder that we intend to monitor it and as such, for our purposes, we will create a fresh file wherein our tripwire setup can be implemented and consequently observed. Right-click anywhere on the desktop → New → Text Document. A new unnamed text file will appear on the Desktop, and we can name it "Secret Document". Right-click the document → Properties → Security → Advanced → Auditing.



This is where we will create and specify an audit entry for our Secret Document. Click on Add → Select a principal.



This particular pop-up is where we can specify which users or groups we want to be alerted on whenever they access our Secret Document. You have a choice to be very granular, but for our purposes, we intend to be general and so, on the text box, write "Everyone", then press Enter.



The Principal should now reflect "Everyone" and the permissions should now allow granular selection. Here you can specify all the activities on our Secret Document you want to audit. You can toggle to the advanced permissions via the Show advanced permissions on the upper right-hand corner of the area for an even broader selection of activities. Click OK when you're done exploring, and then click Apply and OK on the Audit entry page. Finally, we click on OK on our Secret Document properties.

An important note here is if you intend to track multiple files, it is recommended to have them categorized into folders so as to make it easier to audit their access.

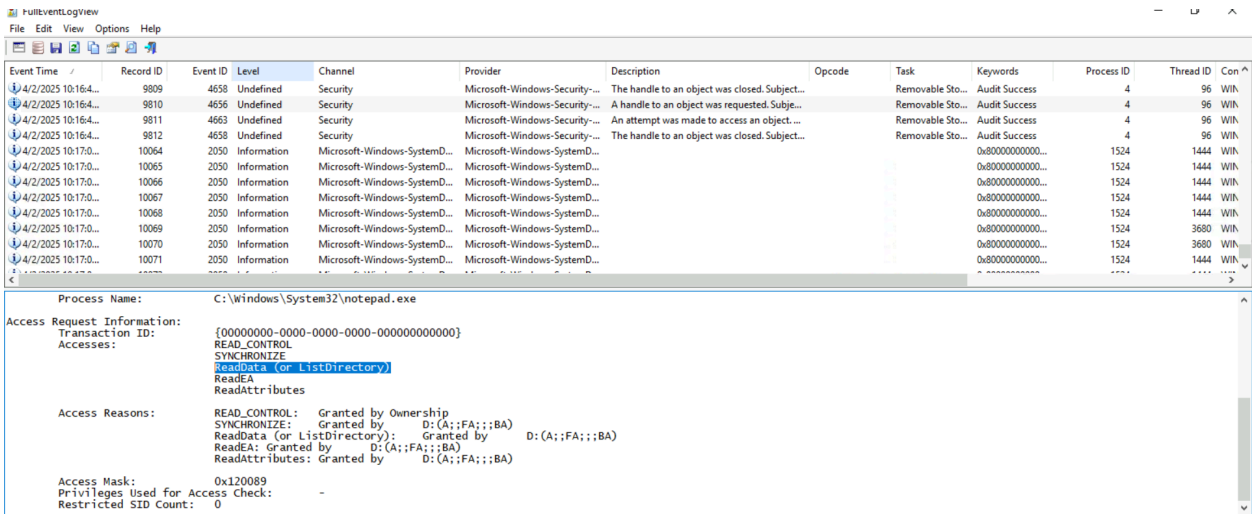
At this point, the setup is complete and auditing is active for our Secret Document. Anyone who accesses it will be logged and its details will be recorded in the Security event log with an Event ID 4663. This Event ID, along with the other Object Access Event IDs, can then be filtered and furnished into alerts that would immediately tell your analysts of tripwires being activated, immediately giving value to the organization's security.

****

Answer the questions below:

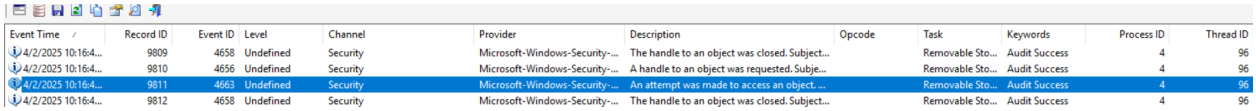
The FullEventLogView application comes installed on your Windows machine. Use it for the following questions.

What is the "Accesses" value in the log details when you try reading our Secret Document's contents via cmd?



Answer: ReadData (or ListDirectory)

Event ID 4663 is always preceded by?



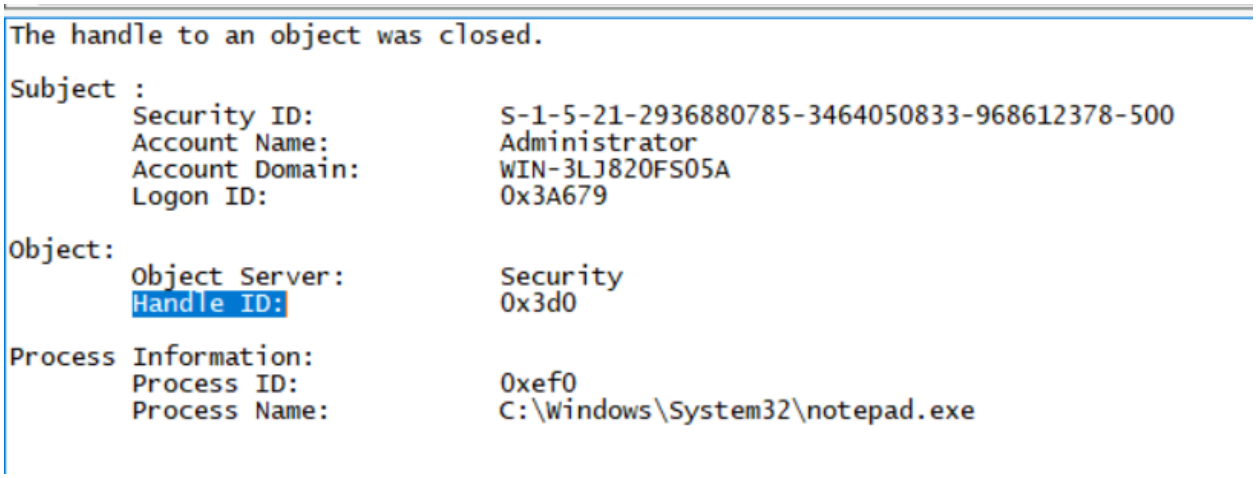
Answer: 4656

What Event ID signifies the closure of an "object"?

Answer: 4658

Event ID 4658 helps determine how long a specific object was open. What description field will you check in between events to be able to do so?

The hint for this question says to read through the external resources and after a little digging around I learned that Windows uses “Handle” to “display information about open handles for any process in the system. You can use it to see the programs that have a file open, or to see the object types and names of all the handles of a program.” according to the official Microsoft documentation.



Answer: Handle ID

Purple Teaming

To cap this room off is a quick lesson on one of the best ways to strengthen an organization's overall security posture - by leveraging purple team tactics. A couple of rooms that showcase purple teaming are the Tempest and Follina (CVE-2022-30190) rooms.

The idea is simple: if you want to see how your defenses fare against an attack, understand how a certain vulnerability works and what it looks like on the logs, or you simply want to know the extent of your visibility on your environment - simulate an attack and then check the results.

Consequently, reflect on the following questions:

- What are the attacker techniques that you did?
- Which ones did you detect?
- Which ones flew under the radar?

The ones that you detected will affirm that you're doing a good job in those areas, whereas the ones you failed to detect constitute improvements that ought to be made in your visibility and/or detection mechanisms.

Quick Discussion for Tempest:

For the Tempest room, the room creator designed a full attack chain and emulated an adversary from start to finish, collecting valuable logs and showcasing detection and analysis tools as the room progresses. This is a classic example of the application of purple team tactics. The goal is to understand how logs will look like when specific attack techniques are being implemented against your environment.

Quick Discussion for Follina MSDT:
From the Follina MSDT room, the room creator focused on the effects of the exploitation of the vulnerability in the environment and introduced how reviewing logs and process artifacts compliments publicly available IOCs.

From these findings, you can furnish alerts from the artifacts you've collected as well as from findings that you're able to observe via your logging mechanisms. These are just a couple of ways to leverage purple team tactics, and both show emphasis on how effective it is when leveraged well.

****

Answer the questions below:

Fill in the Blanks: The Tempest and Follina rooms are examples of leveraging _____ ____ tactics.
Answer: Purple Team

What CVE is the Follina MSDT room about?
Answer Format: CVE-XXXX-XXXXX
Answer: CVE-2022-30190

Purple teaming is awesome! Acknowledge me if you can!
Answer: No answer needed

Room Recap

This room explored some of the simplest ways that an organization can leverage immediately available information, "knowing your environment", and purple team tactics to develop a more robust detection mechanism - one that is more suited for each unique environment. In a world of purchase-deploy-and-forget, these tactics aim to provide an additional layer of security that would immediately be of value to wherever they are applied.

This room has also emphasized the value of layering defenses. It is never a good idea to put all your eggs in one basket in the same way that an organization shouldn't put all their time, money, and reputation on the shoulders of one method of detection and/or prevention. No method is perfect, so having depth in these defensive methods would allow for more opportunities to catch the bad guys.

Finally, a couple of rooms that leverage purple team tactics have been the chosen method of closing this topic.