GRID-SIEM SD GROUP 29 SPRING '24

Trent Bickford
Westin Chamberlain
Ella Cook
Daniel Ocampo

ML Realtime Option 1 – Kafka Stream

- Adapting current ML approach with Kafka Stream
    - Pros
        - Kafka can handle large volumes of data with real-time data processing
        - can be integrated with Zeek & is open source
    - Cons
        - Training and running the ML model in real time will take a lot of computational overhead and require robust hardware resources
        - Might introduce compatibility issues & data privacy issues
        - Creates a single point of failure
    - Changes required for implementation
        - Install and configure a Zeek to Kafka Plugin
        - Build plugin based on GIT page
        - Configure plugin
            - Will require changing settings by creating a new Zeek Script
        - Restart Zeek to begin forwarding logs to Kafka
        - Verify log forwarding
        - The model must be loaded into application's memory to ensure minimal latency in processing and predicting
    - Will continually need to collect new batches of zeek logs over time
    - The ML model will also need to continually be retrained & re-evaluated

ML Realtime Option 2 – Suricata & Watchdog

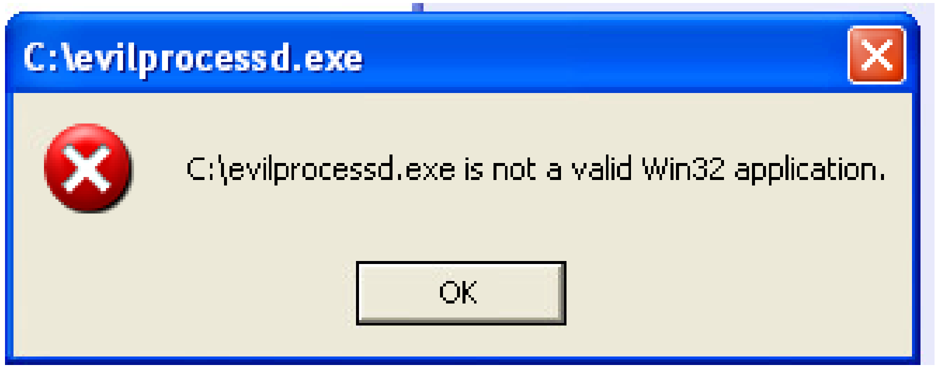
- Pros & Cons
  - Pros
    - Similar to Kafka will be able to ingest the proper number of logs in real time
    - Can be integrated with Suricata & is open source
  - Cons
    - Same as kafka
      - Computational overhead, technical challenges of setup and implementation, data privacy, and creates a single point of failure
- Requirements
  - Configure suricata
  - Output logs will need to be in eve.json format
  - Will need a watchdog-specific script to continually monitor the eve.logs in real time
    - Will also need to specify and understand the types of logs we want to include
  - Then the second script will be the machine learning – which will have to have been loaded onto the python script memory with a library to do that (will need to look into options)
  - All other steps remain the same

ML Considerations

- Implementation of 3rd party app & configuration to make real-time ML a reality
- Feasability of two fully functional ML components in time
- Partial functionality of realtime depending on implementation challenges that arise
- Still need to complete the planning stages of real-time since it was not in scope of project last semester
- Previous two slides are rough outline

Caldera Issues cont.

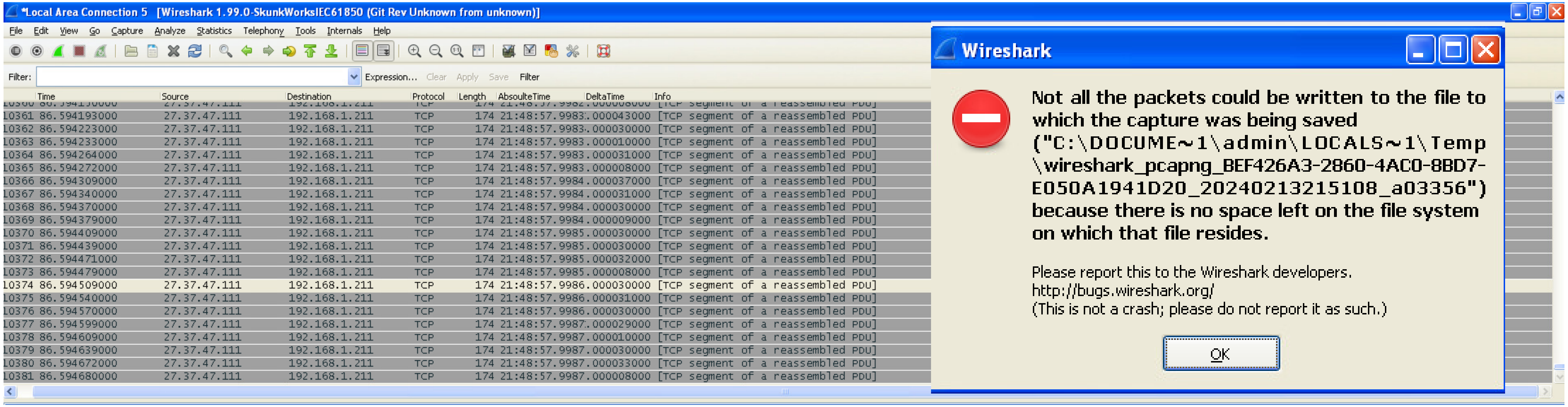
- Exe file not compatible with RTU machines
  - Security preventing exe file from running
  - Compatibility issues (old windows machine)
- to get caldera working it would probably require rewriting the entire agent program



C:\evilprocessd.exe

C:\evilprocessd.exe is not a valid Win32 application.

Ping Flood attack



*Local Area Connection 5 [Wireshark 1.99.0-SkunkWorksIEC61850 (Git Rev Unknown from unknown)]

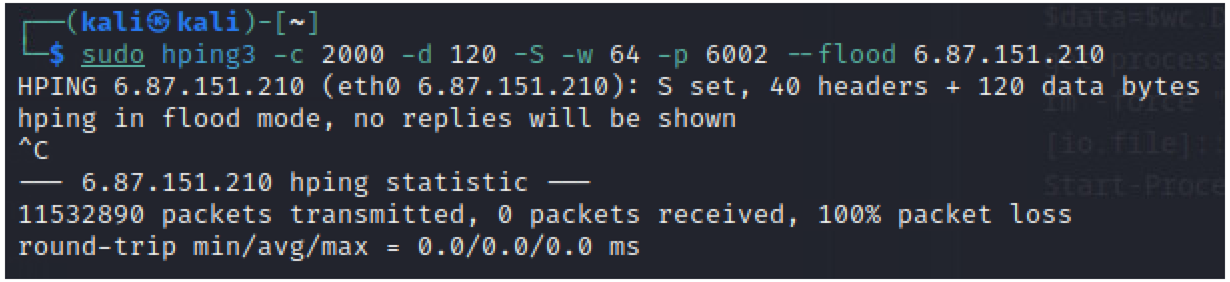
```
       Time          Source         Destination     Protocol Length AbsoulteTime      DeltaTime  Info
10361 86.594193000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9983.000043000 [TCP segment of a reassembled PDU]
10362 86.594223000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9983.000030000 [TCP segment of a reassembled PDU]
10363 86.594233000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9983.000010000 [TCP segment of a reassembled PDU]
10364 86.594264000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9983.000031000 [TCP segment of a reassembled PDU]
10365 86.594272000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9983.000008000 [TCP segment of a reassembled PDU]
10366 86.594309000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9984.000037000 [TCP segment of a reassembled PDU]
10367 86.594340000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9984.000031000 [TCP segment of a reassembled PDU]
10368 86.594370000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9984.000030000 [TCP segment of a reassembled PDU]
10369 86.594379000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9984.000009000 [TCP segment of a reassembled PDU]
10370 86.594409000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9985.000030000 [TCP segment of a reassembled PDU]
10371 86.594439000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9985.000030000 [TCP segment of a reassembled PDU]
10372 86.594471000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9985.000032000 [TCP segment of a reassembled PDU]
10373 86.594479000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9985.000008000 [TCP segment of a reassembled PDU]
10374 86.594509000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9986.000030000 [TCP segment of a reassembled PDU]
10375 86.594540000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9986.000031000 [TCP segment of a reassembled PDU]
10376 86.594570000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9986.000030000 [TCP segment of a reassembled PDU]
10377 86.594599000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9987.000029000 [TCP segment of a reassembled PDU]
10378 86.594609000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9987.000010000 [TCP segment of a reassembled PDU]
10379 86.594639000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9987.000030000 [TCP segment of a reassembled PDU]
10380 86.594672000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9987.000033000 [TCP segment of a reassembled PDU]
10381 86.594680000   27.37.47.111   192.168.1.211   TCP      174 21:48:57.9987.000008000 [TCP segment of a reassembled PDU]
```

Wireshark

Not all the packets could be written to the file to which the capture was being saved ("C:\DOCUME~1\admin\LOCALS~1\Temp\wireshark_pcapng_BEF426A3-2860-4AC0-8BD7-E050A1941D20_20240213215108_a03356") because there is no space left on the file system on which that file resides.

Please report this to the Wireshark developers.
http://bugs.wireshark.org/
(This is not a crash; please do not report it as such.)

Says it dropped all packets?



```
┌──(kali㉿kali)-[~]
└─$ sudo hping3 -c 2000 -d 120 -S -w 64 -p 6002 --flood 6.87.151.210
HPING 6.87.151.210 (eth0 6.87.151.210): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
^C
--- 6.87.151.210 hping statistic ---
11532890 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

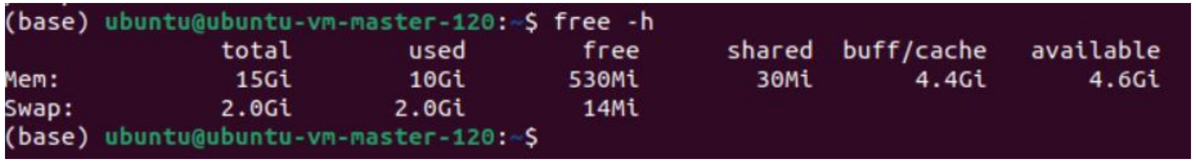
Storage filling up was an issue I was facing with agent deployment I didn’t know about

Other Potential Attacks

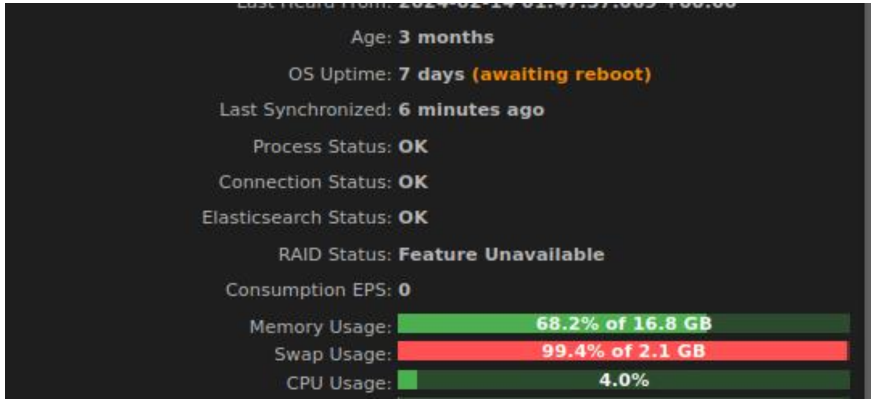
- Use the scripts written in old adversary box
  - Could be a good way to remind myself of basic red-teaming
  - They are tested so we know they work
- With PowerShell I can automate DoS, Malware, or scripts to run periodically
  - "Start-Process –FilePath "C:/…/…" -NoNewWindow"
- Could use the machine to attack the other machines
  - Think botnet

Security Onion Work

- Created another swap file to try to fix the alerts and log issues



```
(base) ubuntu@ubuntu-vm-master-120:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:            15Gi        10Gi       530Mi        30Mi       4.4Gi       4.6Gi
Swap:          2.0Gi       2.0Gi        14Mi
(base) ubuntu@ubuntu-vm-master-120:~$
```



Age: **3 months**

OS Uptime: **7 days (awaiting reboot)**

Last Synchronized: **6 minutes ago**

Process Status: **OK**

Connection Status: **OK**

Elasticsearch Status: **OK**

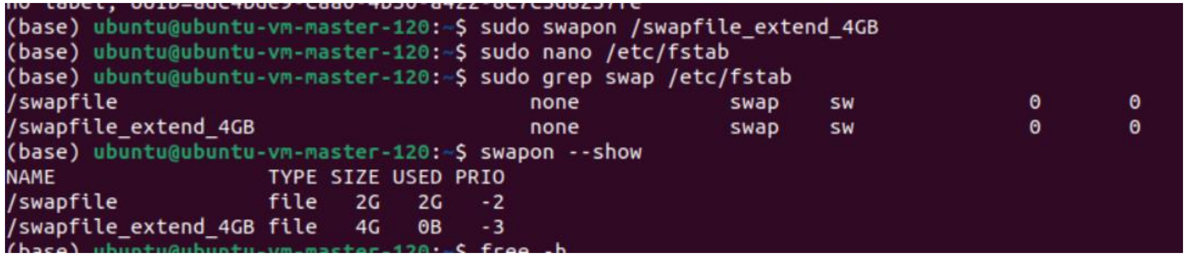
RAID Status: **Feature Unavailable**

Consumption EPS: **0**

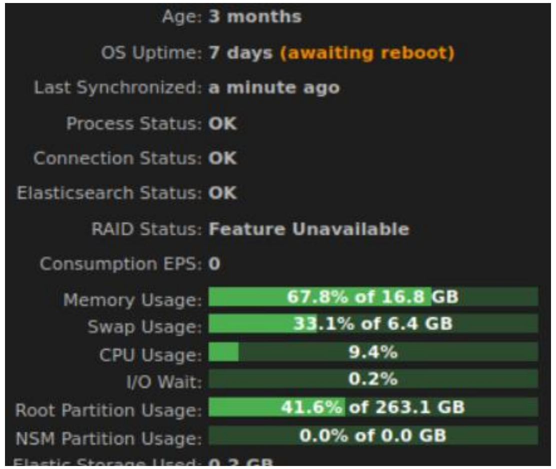
Memory Usage: **68.2% of 16.8 GB**

Swap Usage: **99.4% of 2.1 GB**

CPU Usage: **4.0%**



```
(base) ubuntu@ubuntu-vm-master-120:~$ sudo swapon /swapfile_extend_4GB
(base) ubuntu@ubuntu-vm-master-120:~$ sudo nano /etc/fstab
(base) ubuntu@ubuntu-vm-master-120:~$ sudo grep swap /etc/fstab
/swapfile                                 none            swap    sw              0       0
/swapfile_extend_4GB                      none            swap    sw              0       0
(base) ubuntu@ubuntu-vm-master-120:~$ swapon --show
NAME                 TYPE SIZE USED PRIO
/swapfile            file   2G   2G   -2
/swapfile_extend_4GB file   4G   0B   -3
(base) ubuntu@ubuntu-vm-master-120:~$ free -h
```



Age: **3 months**

OS Uptime: **7 days (awaiting reboot)**

Last Synchronized: **a minute ago**

Process Status: **OK**

Connection Status: **OK**

Elasticsearch Status: **OK**

RAID Status: **Feature Unavailable**

Consumption EPS: **0**

Memory Usage: **67.8% of 16.8 GB**

Swap Usage: **33.1% of 6.4 GB**

CPU Usage: **9.4%**

I/O Wait: **0.2%**

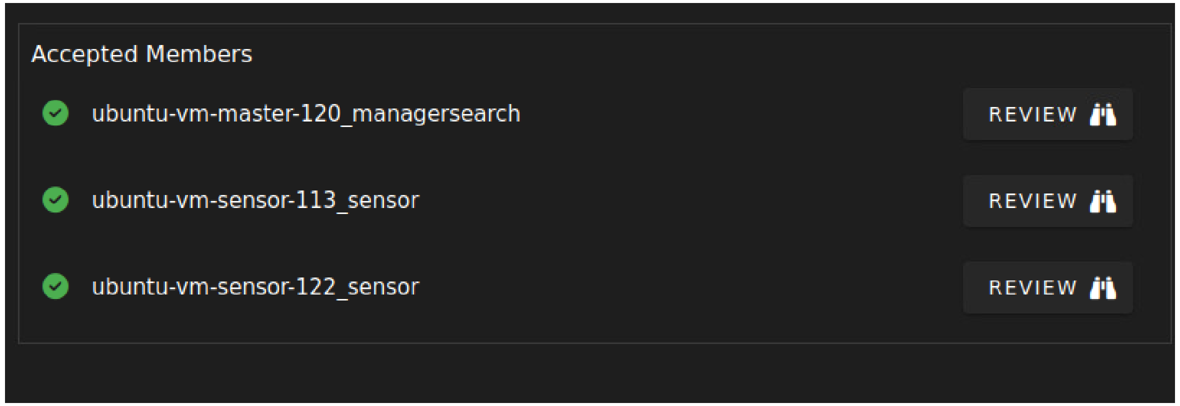
Root Partition Usage: **41.6% of 263.1 GB**

NSM Partition Usage: **0.0% of 0.0 GB**

Elastic Storage Used: **0.2 GB**

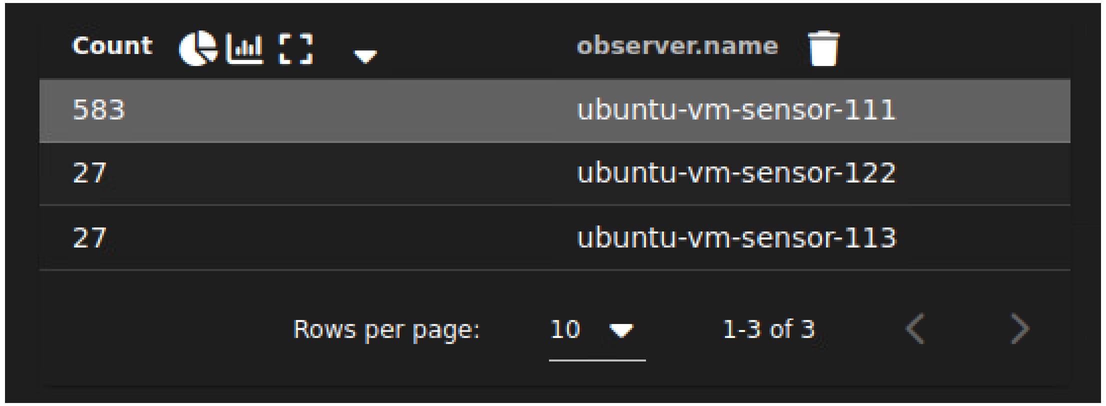
Security Onion Work

- Got the logs coming back into Security Onion Console by reverting to a previous snapshot, so had to redo some of the work in the past weeks
- Caused some issues with the grid display, but still intaking the sensors' logs



Accepted Members

- ubuntu-vm-master-120_managersearch
- ubuntu-vm-sensor-113_sensor
- ubuntu-vm-sensor-122_sensor



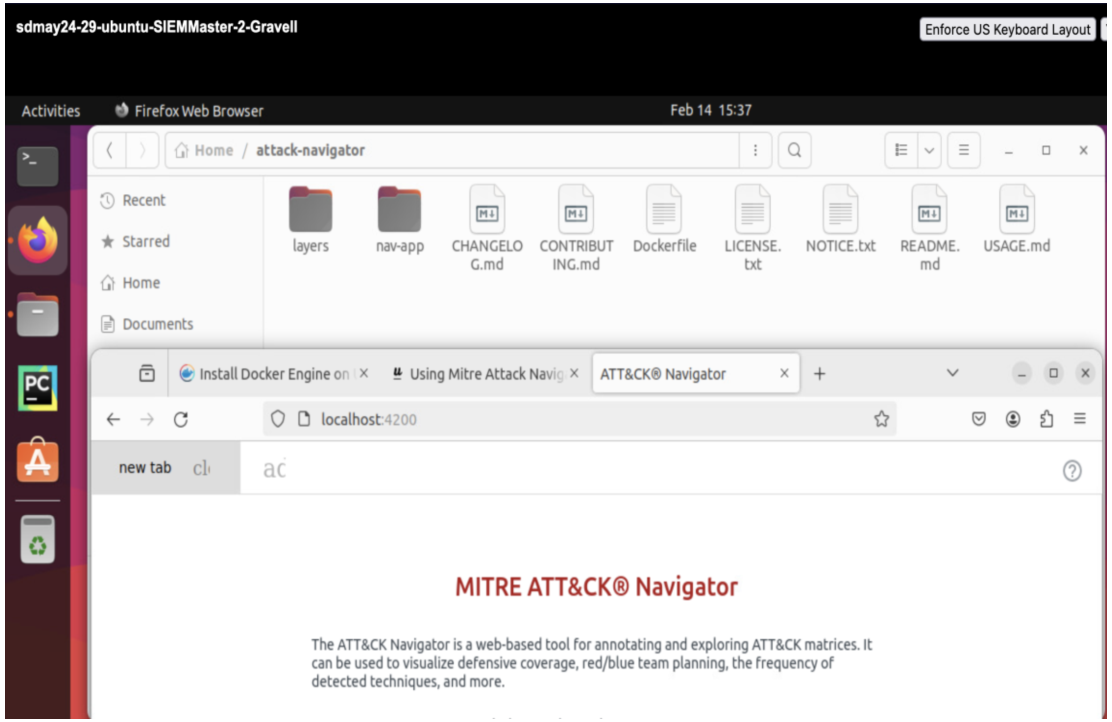
Count	observer.name
583	ubuntu-vm-sensor-111
27	ubuntu-vm-sensor-122
27	ubuntu-vm-sensor-113

Security Onion Future Work

- Adjust the rules in Suricata for OT
- Explore more of the applications like Kibana since the alerts and logs are displayed
- Focus on better connection and transparency with each of the components, i.e., machine learning, Caldera, and Navigator

ATT&CK Navigator

- Docker container running on Gravwell VM to host Navigator locally.
- This way we can modify and dynamically update the matrix using ATT&CK APIs.
- Plan: Caldera/kali attacks > SO collects logs > logs build matrix with Navigator > SO Playbook uses this info to build prevention/detection rules > Playbook constructs its own matrix of TTP defense perimeter.



MITRE ATT&CK® Navigator

The ATT&CK Navigator is a web-based tool for annotating and exploring ATT&CK matrices. It can be used to visualize defensive coverage, red/blue team planning, the frequency of detected techniques, and more.