OFFENSE / DEFENSE

PURPLE TEAM

DAVIS ESTHER
MICHAEL MARTINEZ
THOMAS DANNA
PRECIOUS

OFFENSE:

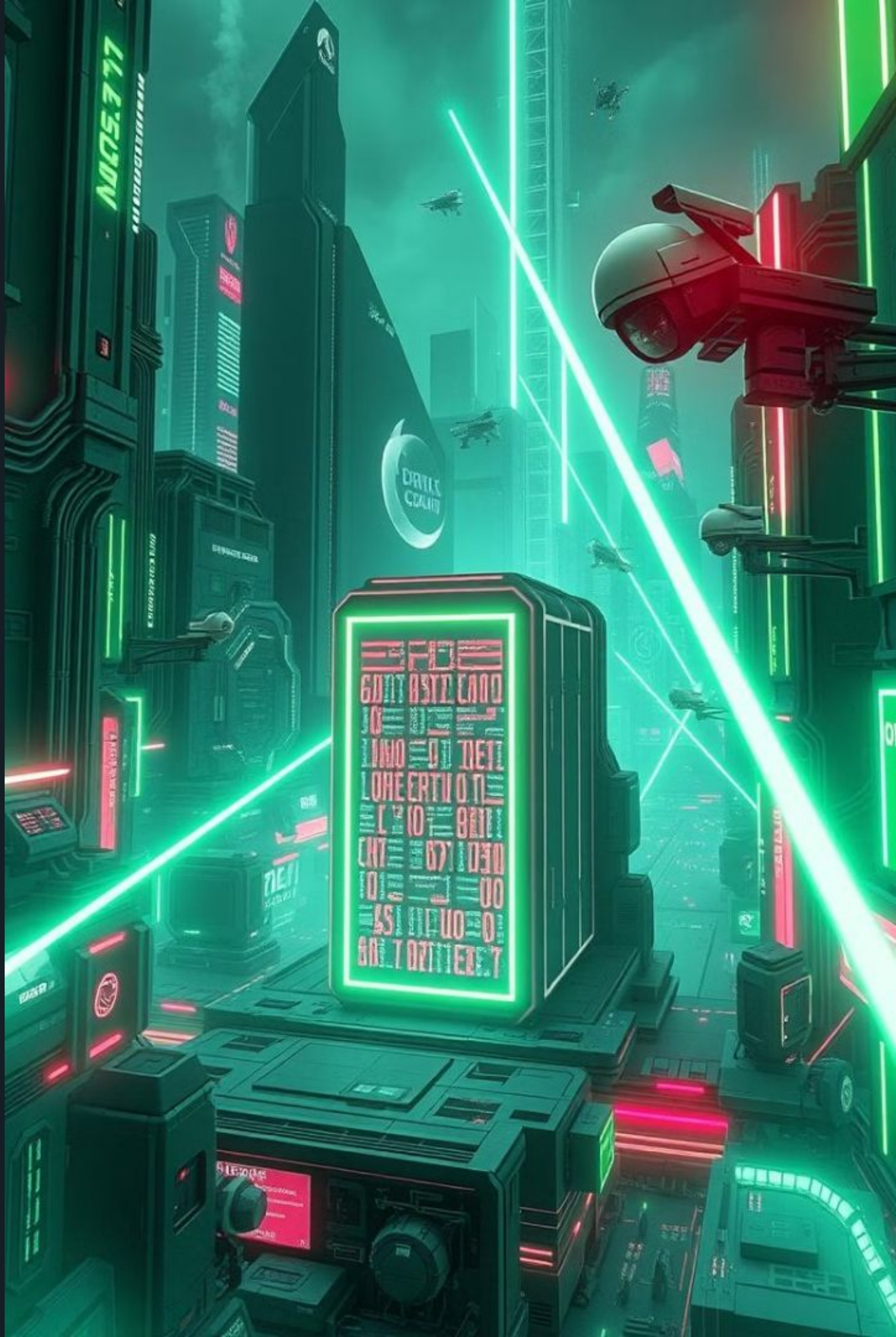
Automated Python Scripts for nmap

DEFENSE:

IDS

IT Cyber Security: Creating Automation through Python

(Esther)



Developed Three Custom Python Scripts

- nmap.py (base script)
  - Automates Nmap scans on user-specified target IP/hostname at user-defined intervals
  - Allows for user-input for all Nmap scan options
  - Allows for continuous, repeated scanning at the specified intervals
  - May be run continuously in the background until manually stopped
  - Outputs results of scans to console and appends to specified file for record-keeping
  - Helps to monitor changes in the target's network status

- scanmap.py (base script run with default options)
  - Runs Nmap with predefined default options (-sS -sV -A -T4 -p-) for quick extensive scan

- spoofscan.py (base script with option for spoofing source IP)
  - Scans with quick option for spoofing source IP and specifying a network interface

Custom scripts offer a more hands-on approach to managing Nmap scans compared to the more rigid and less interactive nature of Cron jobs. Python scripts can also be easily altered to suit current needs.

Advantages of Using Custom Scripts (nmap.py)

1. **Interactive Configuration**

   Allows users to specify target addresses, Nmap options, scan frequency, and output file names interactively at runtime, making it flexible and user-friendly for ad-hoc scanning rather than requiring pre-defined settings in a Cron job.

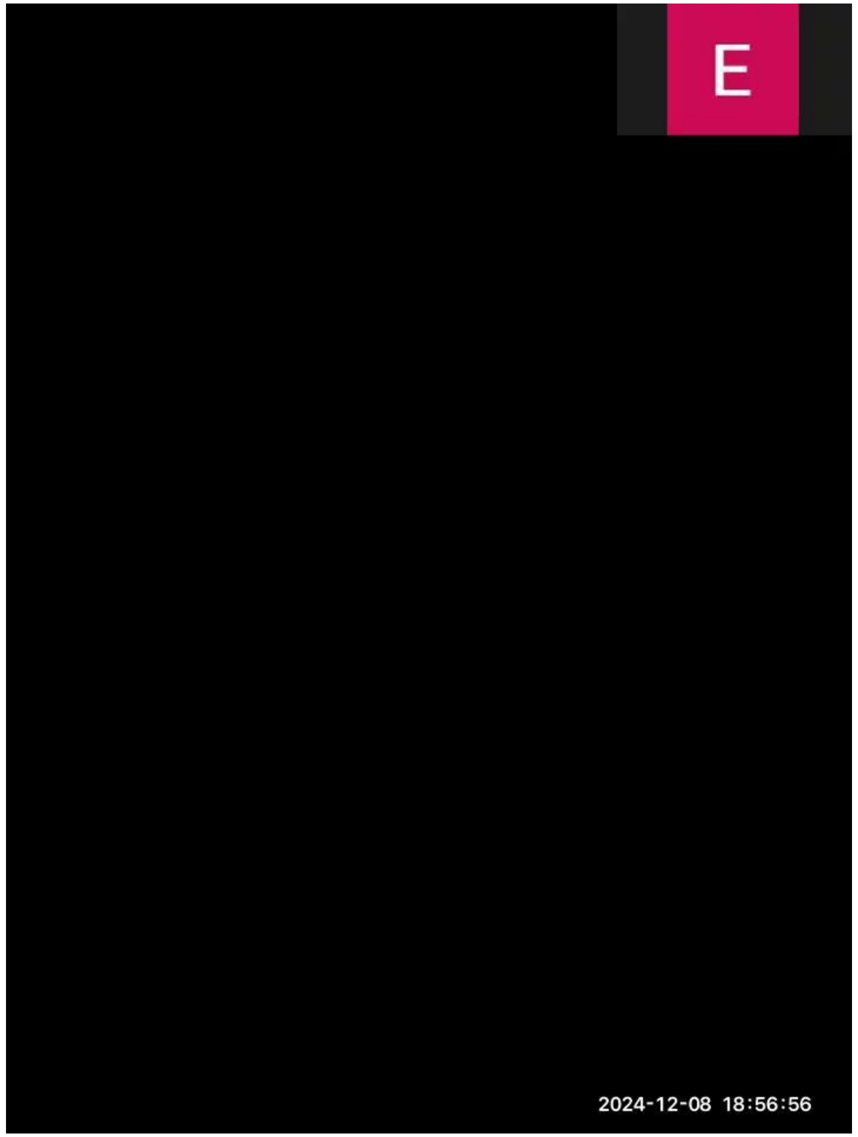
1. **Dynamic Scanning**

   Users can easily modify the scan frequency or options between runs without altering the Cron job configuration, enabling quick adjustments based on current needs or conditions.

1. **Immediate Feedback and Error Handling**

   Provides real-time feedback on the success or failure of each scan attempt, along with detailed error messages if something goes wrong, which is not possible with Cron jobs that typically log outputs to files without interactive reporting.

Live Demonstration



Automating Threat Detection Visualized

(Davis)

Enhancing Nmap Scan Readability and Organization

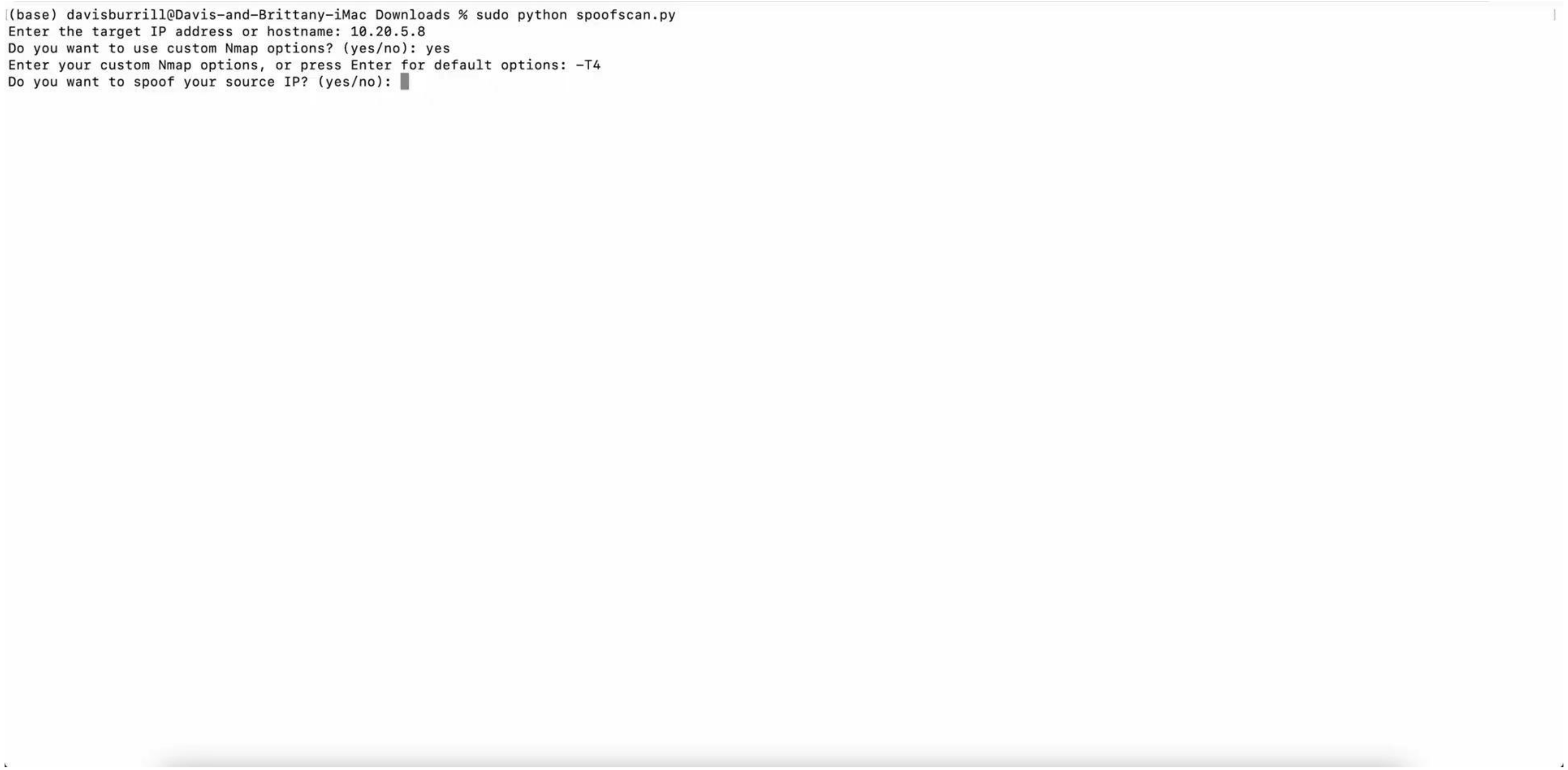
- Objective:
  - To transform raw Nmap scan results into a clear and professional format for easy interpretation.
- Key Contributions:
  - XML-to-HTML Transformation:
    - Used XSLT to convert Nmap XML Output into a styled, readable HTML format.
  - Timestamping:
    - Ensured file names include timestamps (e.g., 2024-12-09_19-56) to prevent overwriting and enable traceability
  - Organized Storage:
    - Created a dedicated folder, Nmap_Scans, to keep all scan results systematically stored.

How the Enhancements Work

Steps Taken:

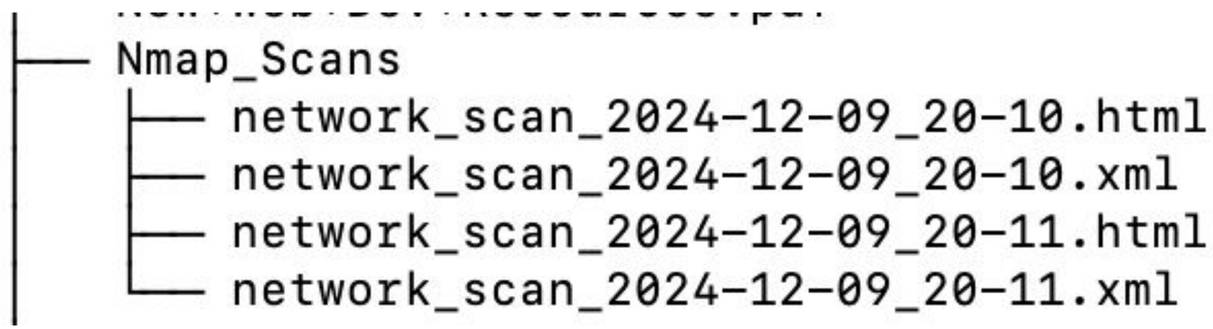
1. User runs the script and inputs:
   a. Target for the scan
   b. Desired filename
   c. Scan frequency
2. The script:
   a. Runs the Nmap scan and generates XML output.
   b. Uses XSLT to convert XML to HTML for readability.
   c. Saves both the XML and HTML files in the Nmap_Scans folder.
3. Outputs are time-stamped for organization and consistency.

Live Demonstration

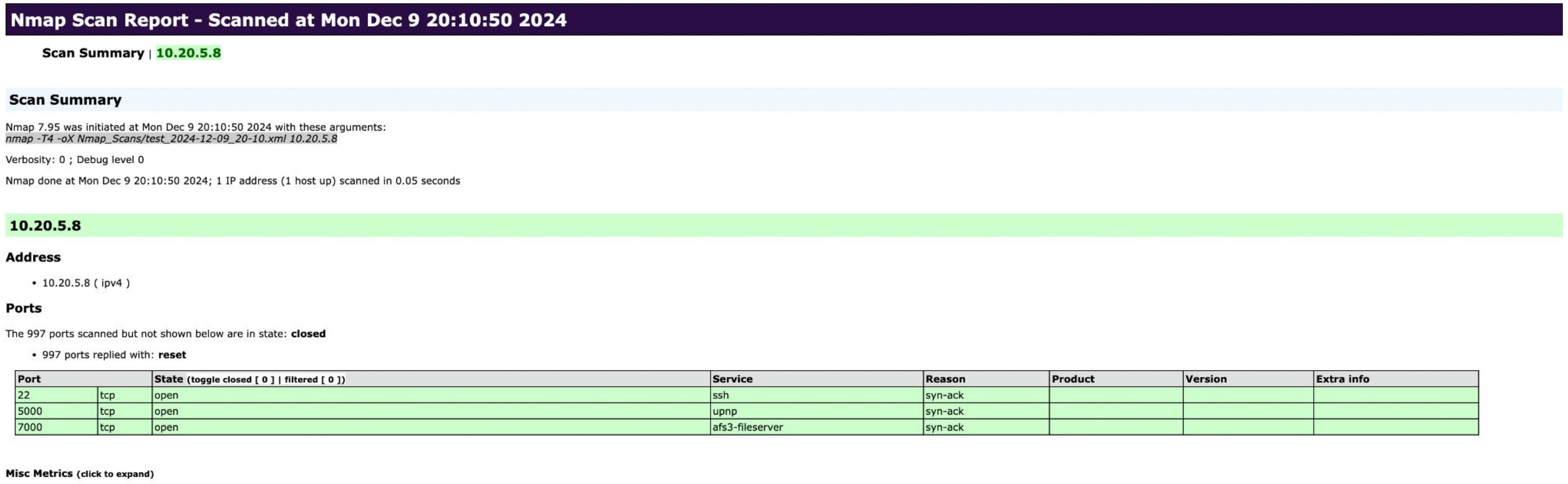


```
(base) davisburrill@Davis-and-Brittany-iMac Downloads % sudo python spoofscan.py
Enter the target IP address or hostname: 10.20.5.8
Do you want to use custom Nmap options? (yes/no): yes
Enter your custom Nmap options, or press Enter for default options: -T4
Do you want to spoof your source IP? (yes/no): 
```

How the Enhancements Work

- 

```
├── Nmap_Scans
│   ├── network_scan_2024-12-09_20-10.html
│   ├── network_scan_2024-12-09_20-10.xml
│   ├── network_scan_2024-12-09_20-11.html
│   └── network_scan_2024-12-09_20-11.xml
```

- 

Nmap Scan Report - Scanned at Mon Dec 9 20:10:50 2024

Scan Summary | 10.20.5.8

Scan Summary

Nmap 7.95 was initiated at Mon Dec 9 20:10:50 2024 with these arguments:
nmap -T4 -oX Nmap_Scans/test_2024-12-09_20-10.xml 10.20.5.8

Verbosity: 0 ; Debug level 0

Nmap done at Mon Dec 9 20:10:50 2024; 1 IP address (1 host up) scanned in 0.05 seconds

10.20.5.8

Address

- 10.20.5.8 (ipv4)

Ports

The 997 ports scanned but not shown below are in state: **closed**

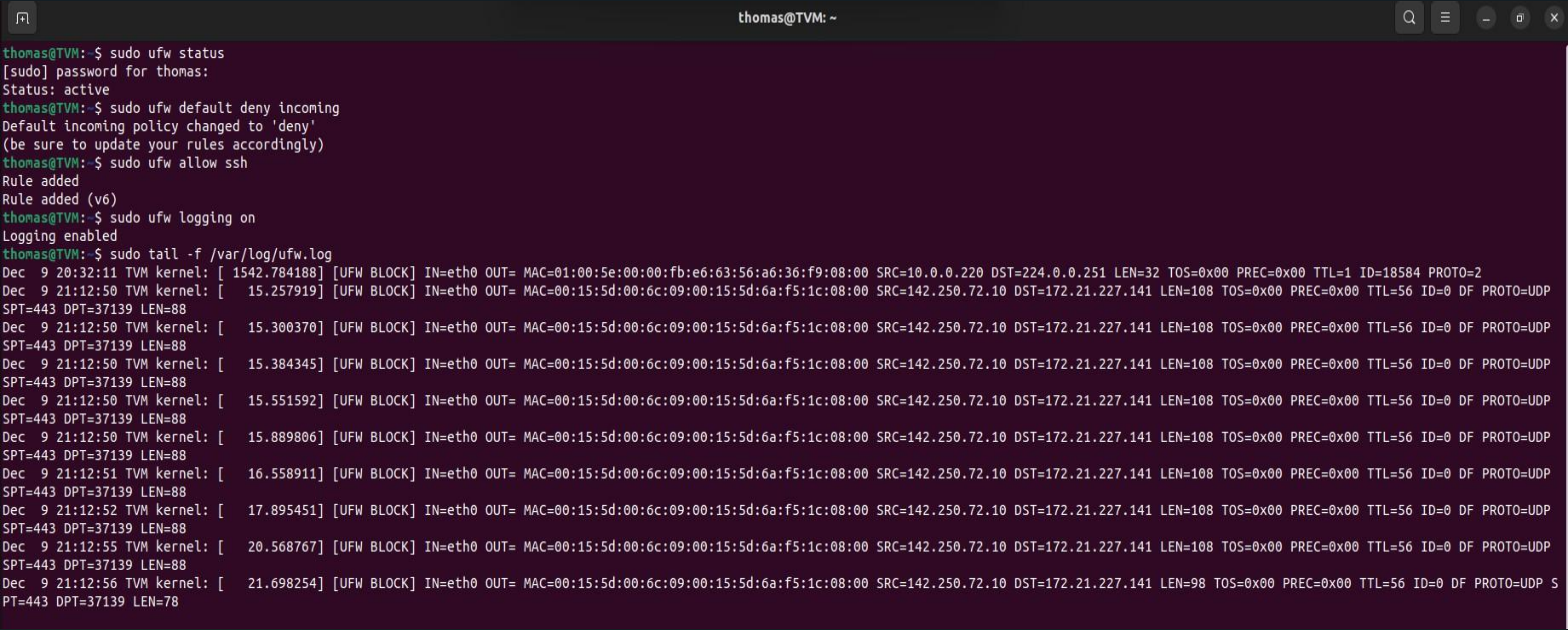
- 997 ports replied with: **reset**

Port		State (toggle closed [0] | filtered [0])	Service	Reason	Product	Version	Extra info
22	tcp	open	ssh	syn-ack			
5000	tcp	open	upnp	syn-ack			
7000	tcp	open	afs3-fileserver	syn-ack			

Misc Metrics (click to expand)

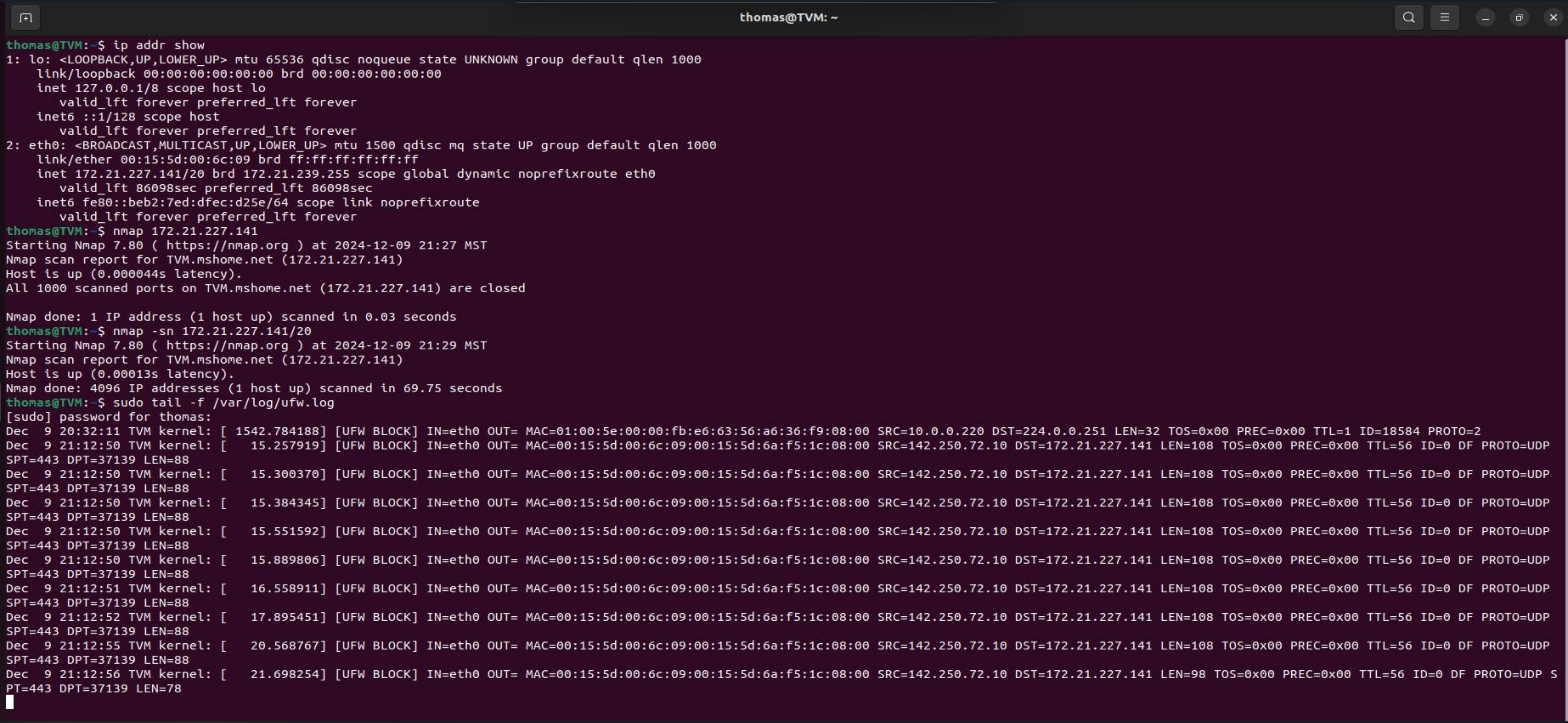
Firewalls: Defending the Network

(THOMAS)



```
thomas@TVM:~$ sudo ufw status
[sudo] password for thomas:
Status: active
thomas@TVM:~$ sudo ufw default deny incoming
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
thomas@TVM:~$ sudo ufw allow ssh
Rule added
Rule added (v6)
thomas@TVM:~$ sudo ufw logging on
Logging enabled
thomas@TVM:~$ sudo tail -f /var/log/ufw.log
Dec  9 20:32:11 TVM kernel: [ 1542.784188] [UFW BLOCK] IN=eth0 OUT= MAC=01:00:5e:00:00:fb:e6:63:56:a6:36:f9:08:00 SRC=10.0.0.220 DST=224.0.0.251 LEN=32 TOS=0x00 PREC=0x00 TTL=1 ID=18584 PROTO=2
Dec  9 21:12:50 TVM kernel: [   15.257919] [UFW BLOCK] IN=eth0 OUT= MAC=00:15:5d:00:6c:09:00:15:5d:6a:f5:1c:08:00 SRC=142.250.72.10 DST=172.21.227.141 LEN=108 TOS=0x00 PREC=0x00 TTL=56 ID=0 DF PROTO=UDP
SPT=443 DPT=37139 LEN=88
Dec  9 21:12:50 TVM kernel: [   15.300370] [UFW BLOCK] IN=eth0 OUT= MAC=00:15:5d:00:6c:09:00:15:5d:6a:f5:1c:08:00 SRC=142.250.72.10 DST=172.21.227.141 LEN=108 TOS=0x00 PREC=0x00 TTL=56 ID=0 DF PROTO=UDP
SPT=443 DPT=37139 LEN=88
Dec  9 21:12:50 TVM kernel: [   15.384345] [UFW BLOCK] IN=eth0 OUT= MAC=00:15:5d:00:6c:09:00:15:5d:6a:f5:1c:08:00 SRC=142.250.72.10 DST=172.21.227.141 LEN=108 TOS=0x00 PREC=0x00 TTL=56 ID=0 DF PROTO=UDP
SPT=443 DPT=37139 LEN=88
Dec  9 21:12:50 TVM kernel: [   15.551592] [UFW BLOCK] IN=eth0 OUT= MAC=00:15:5d:00:6c:09:00:15:5d:6a:f5:1c:08:00 SRC=142.250.72.10 DST=172.21.227.141 LEN=108 TOS=0x00 PREC=0x00 TTL=56 ID=0 DF PROTO=UDP
SPT=443 DPT=37139 LEN=88
Dec  9 21:12:50 TVM kernel: [   15.889806] [UFW BLOCK] IN=eth0 OUT= MAC=00:15:5d:00:6c:09:00:15:5d:6a:f5:1c:08:00 SRC=142.250.72.10 DST=172.21.227.141 LEN=108 TOS=0x00 PREC=0x00 TTL=56 ID=0 DF PROTO=UDP
SPT=443 DPT=37139 LEN=88
Dec  9 21:12:51 TVM kernel: [   16.558911] [UFW BLOCK] IN=eth0 OUT= MAC=00:15:5d:00:6c:09:00:15:5d:6a:f5:1c:08:00 SRC=142.250.72.10 DST=172.21.227.141 LEN=108 TOS=0x00 PREC=0x00 TTL=56 ID=0 DF PROTO=UDP
SPT=443 DPT=37139 LEN=88
Dec  9 21:12:52 TVM kernel: [   17.895451] [UFW BLOCK] IN=eth0 OUT= MAC=00:15:5d:00:6c:09:00:15:5d:6a:f5:1c:08:00 SRC=142.250.72.10 DST=172.21.227.141 LEN=108 TOS=0x00 PREC=0x00 TTL=56 ID=0 DF PROTO=UDP
SPT=443 DPT=37139 LEN=88
Dec  9 21:12:55 TVM kernel: [   20.568767] [UFW BLOCK] IN=eth0 OUT= MAC=00:15:5d:00:6c:09:00:15:5d:6a:f5:1c:08:00 SRC=142.250.72.10 DST=172.21.227.141 LEN=108 TOS=0x00 PREC=0x00 TTL=56 ID=0 DF PROTO=UDP
SPT=443 DPT=37139 LEN=88
Dec  9 21:12:56 TVM kernel: [   21.698254] [UFW BLOCK] IN=eth0 OUT= MAC=00:15:5d:00:6c:09:00:15:5d:6a:f5:1c:08:00 SRC=142.250.72.10 DST=172.21.227.141 LEN=98 TOS=0x00 PREC=0x00 TTL=56 ID=0 DF PROTO=UDP S
PT=443 DPT=37139 LEN=78
```

One mitigation option is the built in firewall in Ubuntu named UFW. There is no cost associated with using UFW and it is fairly easy to set up. First you must ensure that UFW is active and running. In our example, the first configuration step I took was to deny all incoming traffic. This way we can ensure there are no random open ports. Next I opened port 22, the TCP port. Next I enabled logging so we can see details about network traffic that is ether allowed or blocked. More specifically, the logs will show the date, time, port, as well as the source and destination IP address. These logs are stored in a file on the system, but that can be changed.



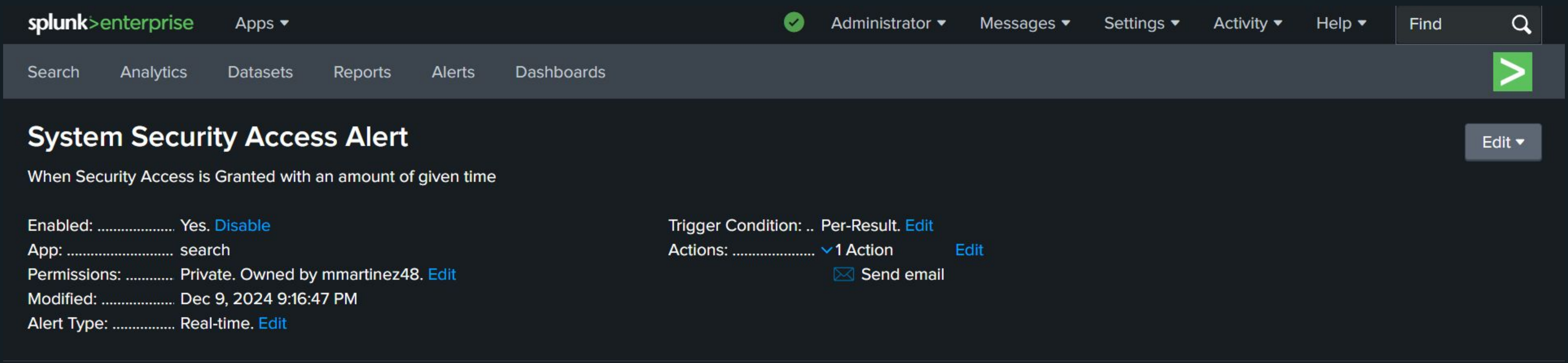
```
thomas@TVM:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:15:5d:00:6c:09 brd ff:ff:ff:ff:ff:ff
    inet 172.21.227.141/20 brd 172.21.239.255 scope global dynamic noprefixroute eth0
       valid_lft 86098sec preferred_lft 86098sec
    inet6 fe80::beb2:7ed:dfec:d25e/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
thomas@TVM:~$ nmap 172.21.227.141
Starting Nmap 7.80 ( https://nmap.org ) at 2024-12-09 21:27 MST
Nmap scan report for TVM.mshome.net (172.21.227.141)
Host is up (0.000044s latency).
All 1000 scanned ports on TVM.mshome.net (172.21.227.141) are closed

Nmap done: 1 IP address (1 host up) scanned in 0.03 seconds
thomas@TVM:~$ nmap -sn 172.21.227.141/20
Starting Nmap 7.80 ( https://nmap.org ) at 2024-12-09 21:29 MST
Nmap scan report for TVM.mshome.net (172.21.227.141)
Host is up (0.00013s latency).
Nmap done: 4096 IP addresses (1 host up) scanned in 69.75 seconds
thomas@TVM:~$ sudo tail -f /var/log/ufw.log
[sudo] password for thomas:
Dec  9 20:32:11 TVM kernel: [ 1542.784188] [UFW BLOCK] IN=eth0 OUT= MAC=01:00:5e:00:00:fb:e6:63:56:a6:36:f9:08:00 SRC=10.0.0.220 DST=224.0.0.251 LEN=32 TOS=0x00 PREC=0x00 TTL=1 ID=18584 PROTO=2
Dec  9 21:12:50 TVM kernel: [   15.257919] [UFW BLOCK] IN=eth0 OUT= MAC=00:15:5d:00:6c:09:00:15:5d:6a:f5:1c:08:00 SRC=142.250.72.10 DST=172.21.227.141 LEN=108 TOS=0x00 PREC=0x00 TTL=56 ID=0 DF PROTO=UDP
SPT=443 DPT=37139 LEN=88
Dec  9 21:12:50 TVM kernel: [   15.300370] [UFW BLOCK] IN=eth0 OUT= MAC=00:15:5d:00:6c:09:00:15:5d:6a:f5:1c:08:00 SRC=142.250.72.10 DST=172.21.227.141 LEN=108 TOS=0x00 PREC=0x00 TTL=56 ID=0 DF PROTO=UDP
SPT=443 DPT=37139 LEN=88
Dec  9 21:12:50 TVM kernel: [   15.384345] [UFW BLOCK] IN=eth0 OUT= MAC=00:15:5d:00:6c:09:00:15:5d:6a:f5:1c:08:00 SRC=142.250.72.10 DST=172.21.227.141 LEN=108 TOS=0x00 PREC=0x00 TTL=56 ID=0 DF PROTO=UDP
SPT=443 DPT=37139 LEN=88
Dec  9 21:12:50 TVM kernel: [   15.551592] [UFW BLOCK] IN=eth0 OUT= MAC=00:15:5d:00:6c:09:00:15:5d:6a:f5:1c:08:00 SRC=142.250.72.10 DST=172.21.227.141 LEN=108 TOS=0x00 PREC=0x00 TTL=56 ID=0 DF PROTO=UDP
SPT=443 DPT=37139 LEN=88
Dec  9 21:12:50 TVM kernel: [   15.889806] [UFW BLOCK] IN=eth0 OUT= MAC=00:15:5d:00:6c:09:00:15:5d:6a:f5:1c:08:00 SRC=142.250.72.10 DST=172.21.227.141 LEN=108 TOS=0x00 PREC=0x00 TTL=56 ID=0 DF PROTO=UDP
SPT=443 DPT=37139 LEN=88
Dec  9 21:12:51 TVM kernel: [   16.558911] [UFW BLOCK] IN=eth0 OUT= MAC=00:15:5d:00:6c:09:00:15:5d:6a:f5:1c:08:00 SRC=142.250.72.10 DST=172.21.227.141 LEN=108 TOS=0x00 PREC=0x00 TTL=56 ID=0 DF PROTO=UDP
SPT=443 DPT=37139 LEN=88
Dec  9 21:12:52 TVM kernel: [   17.895451] [UFW BLOCK] IN=eth0 OUT= MAC=00:15:5d:00:6c:09:00:15:5d:6a:f5:1c:08:00 SRC=142.250.72.10 DST=172.21.227.141 LEN=108 TOS=0x00 PREC=0x00 TTL=56 ID=0 DF PROTO=UDP
SPT=443 DPT=37139 LEN=88
Dec  9 21:12:55 TVM kernel: [   20.568767] [UFW BLOCK] IN=eth0 OUT= MAC=00:15:5d:00:6c:09:00:15:5d:6a:f5:1c:08:00 SRC=142.250.72.10 DST=172.21.227.141 LEN=108 TOS=0x00 PREC=0x00 TTL=56 ID=0 DF PROTO=UDP
SPT=443 DPT=37139 LEN=88
Dec  9 21:12:56 TVM kernel: [   21.698254] [UFW BLOCK] IN=eth0 OUT= MAC=00:15:5d:00:6c:09:00:15:5d:6a:f5:1c:08:00 SRC=142.250.72.10 DST=172.21.227.141 LEN=98 TOS=0x00 PREC=0x00 TTL=56 ID=0 DF PROTO=UDP S
PT=443 DPT=37139 LEN=78
```

In this screenshot you can see an NMAP scan that was run & the last 10 lines of the resulting log.

Splunk: Alerts!

(Michael Martinez)

1. Log Collection
2. Log Parsing and Analysis
3. Alerting



System Security Access Alert

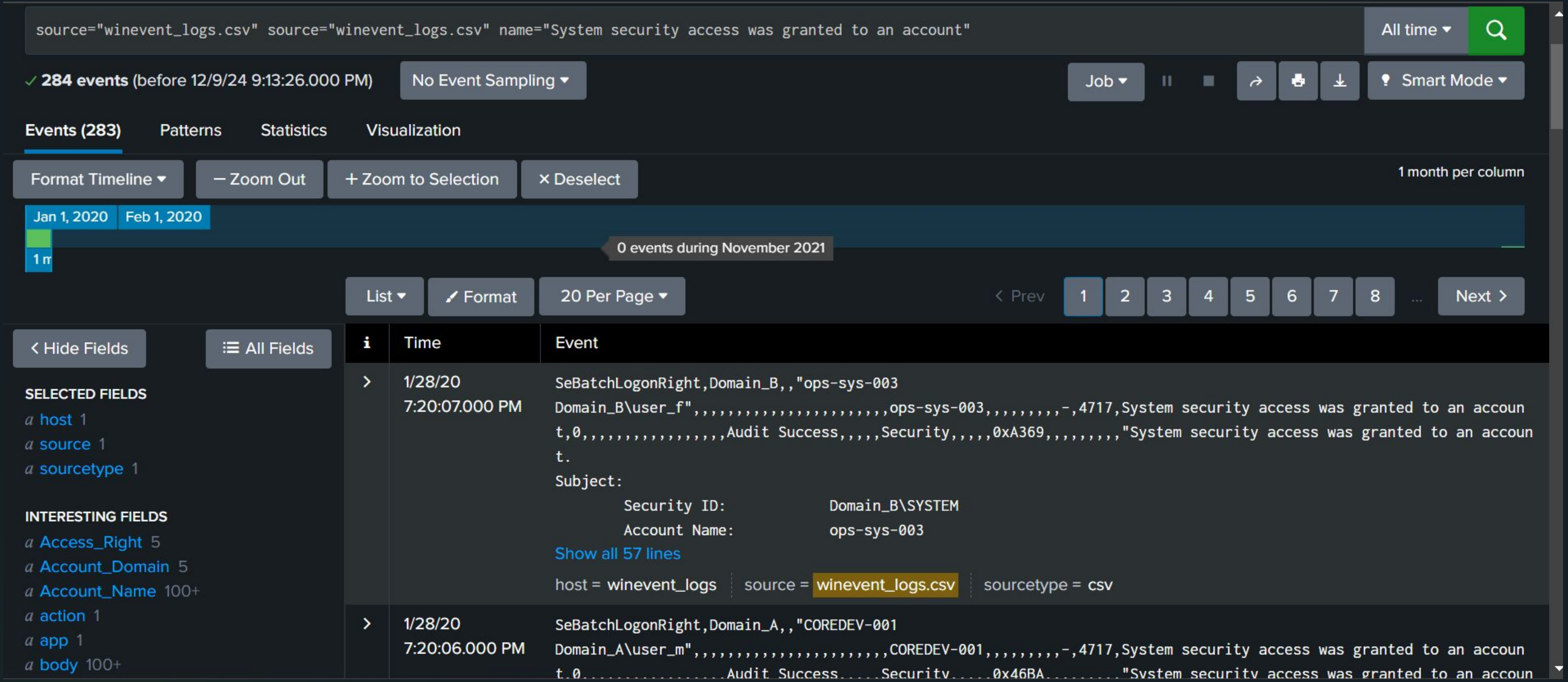
When Security Access is Granted with an amount of given time

Enabled: .... Yes. Disable
App: .... search
Permissions: .... Private. Owned by mmartinez48. Edit
Modified: .... Dec 9, 2024 9:16:47 PM
Alert Type: .... Real-time. Edit

Trigger Condition: .. Per-Result. Edit
Actions: .... 1 Action Edit
Send email

There are no fired events for this alert.

This screenshot is from an Alert that I created that will show Successful Access Granted Log-ins more allotted within a certain time.



```
source="winevent_logs.csv" source="winevent_logs.csv" name="System security access was granted to an account"
```

✓ **284 events** (before 12/9/24 9:13:26.000 PM)

SELECTED FIELDS
- a host 1
- a source 1
- a sourcetype 1

INTERESTING FIELDS
- a Access_Right 5
- a Account_Domain 5
- a Account_Name 100+
- a action 1
- a app 1
- a body 100+

1/28/20 7:20:07.000 PM

```
SeBatchLogonRight,Domain_B,,"ops-sys-003
Domain_B\user_f",,,,,,,,,,,,,,,,,,,,,,,ops-sys-003,,,,,,,,,-,4717,System security access was granted to an accoun
t,0,,,,,,,,,,,,,,,,Audit Success,,,,,Security,,,,,0xA369,,,,,,,,,"System security access was granted to an accoun
t.
Subject:
        Security ID:            Domain_B\SYSTEM
        Account Name:           ops-sys-003
```

host = winevent_logs | source = winevent_logs.csv | sourcetype = csv

1/28/20 7:20:06.000 PM

```
SeBatchLogonRight,Domain_A,,"COREDEV-001
Domain_A\user_m",,,,,,,,,,,,,,,,,,,,,,,COREDEV-001,,,,,,,,,-,4717,System security access was granted to an accoun
t,0,,,,,,,,,,,,,,,,Audit Success,,,,,Security,,,,,0x46BA,,,,,,,,,"System security access was granted to an accoun
```

Screen shot of the all the Successful Event Log ins

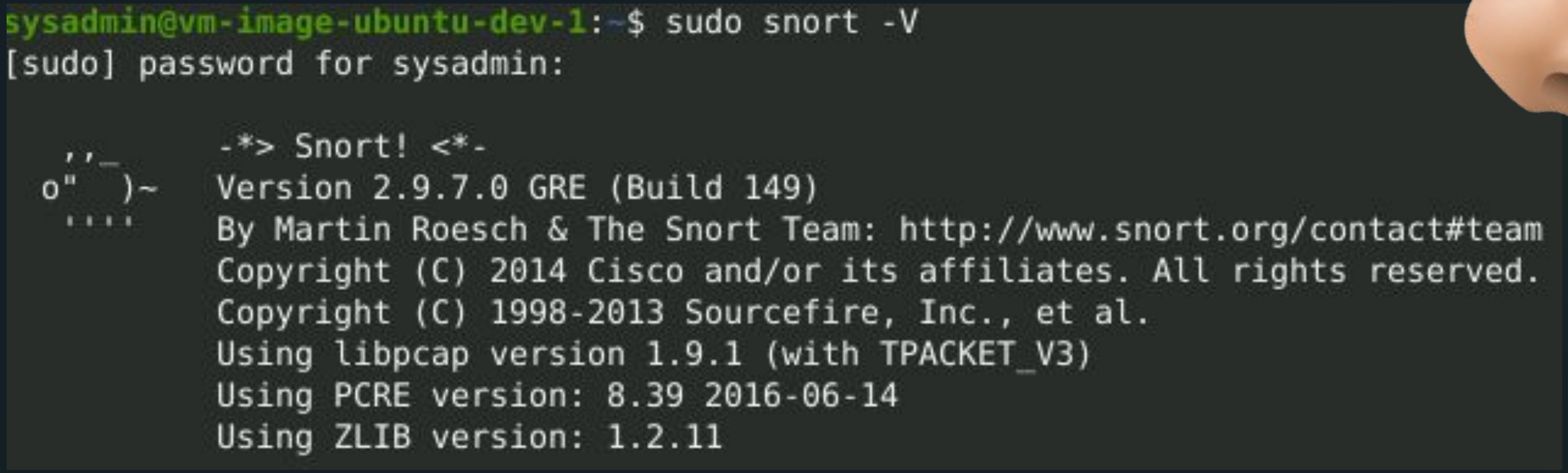
Snort: Sniffing (Danna)

1. Packet Sniffing
2. Intrusion Detection
3. Intrusion Prevention

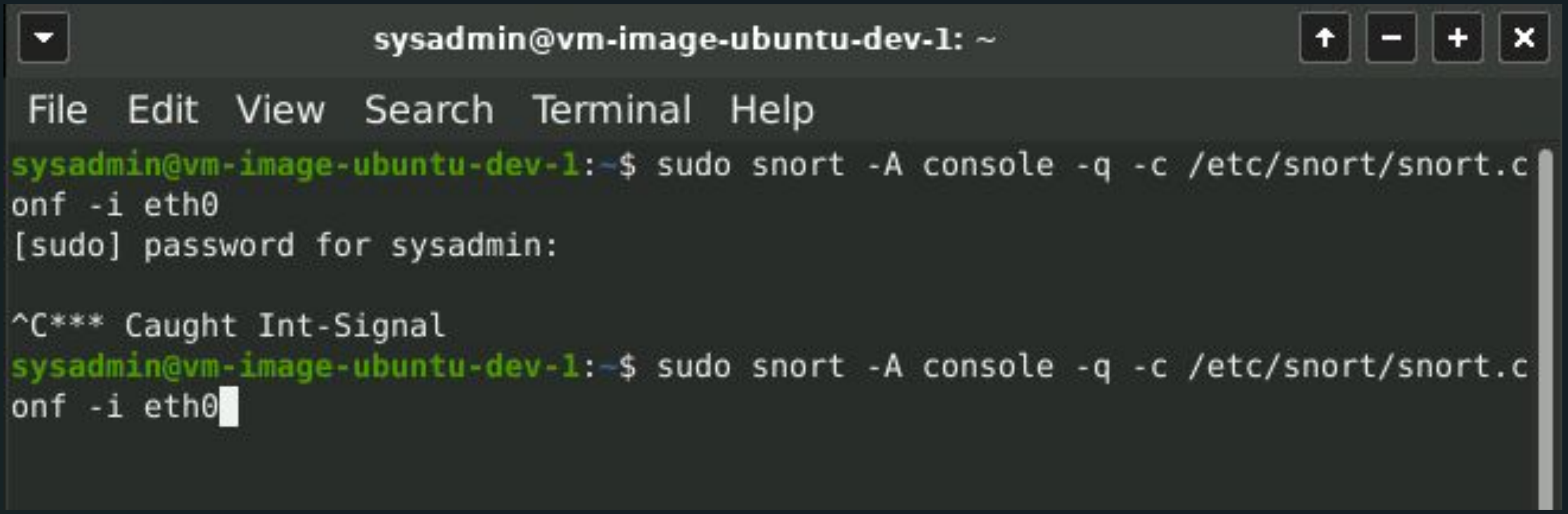
Snort

Running snort through the terminal to detect intrusion via nmap.

sniff sniff



```
sysadmin@vm-image-ubuntu-dev-1:~$ sudo snort -V
[sudo] password for sysadmin:

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.7.0 GRE (Build 149)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.9.1 (with TPACKET_V3)
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.11
```

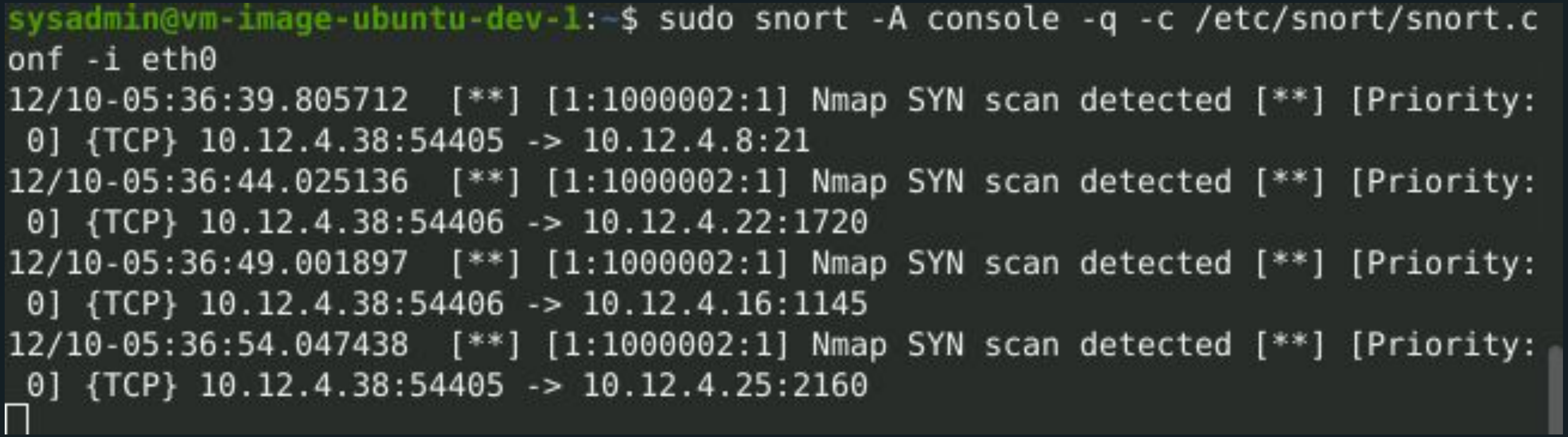


sysadmin@vm-image-ubuntu-dev-1: ~
File Edit View Search Terminal Help

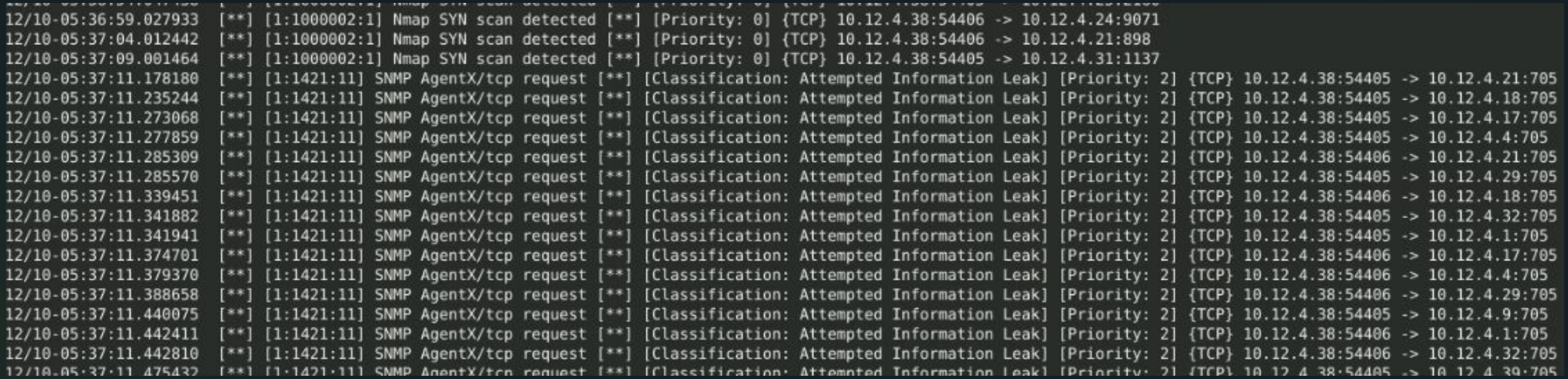
```
sysadmin@vm-image-ubuntu-dev-1:~$ sudo snort -A console -q -c /etc/snort/snort.c
onf -i eth0
[sudo] password for sysadmin:

^C*** Caught Int-Signal
sysadmin@vm-image-ubuntu-dev-1:~$ sudo snort -A console -q -c /etc/snort/snort.c
onf -i eth0
```

IDS

Results of the scans, showing the Nmap scans.



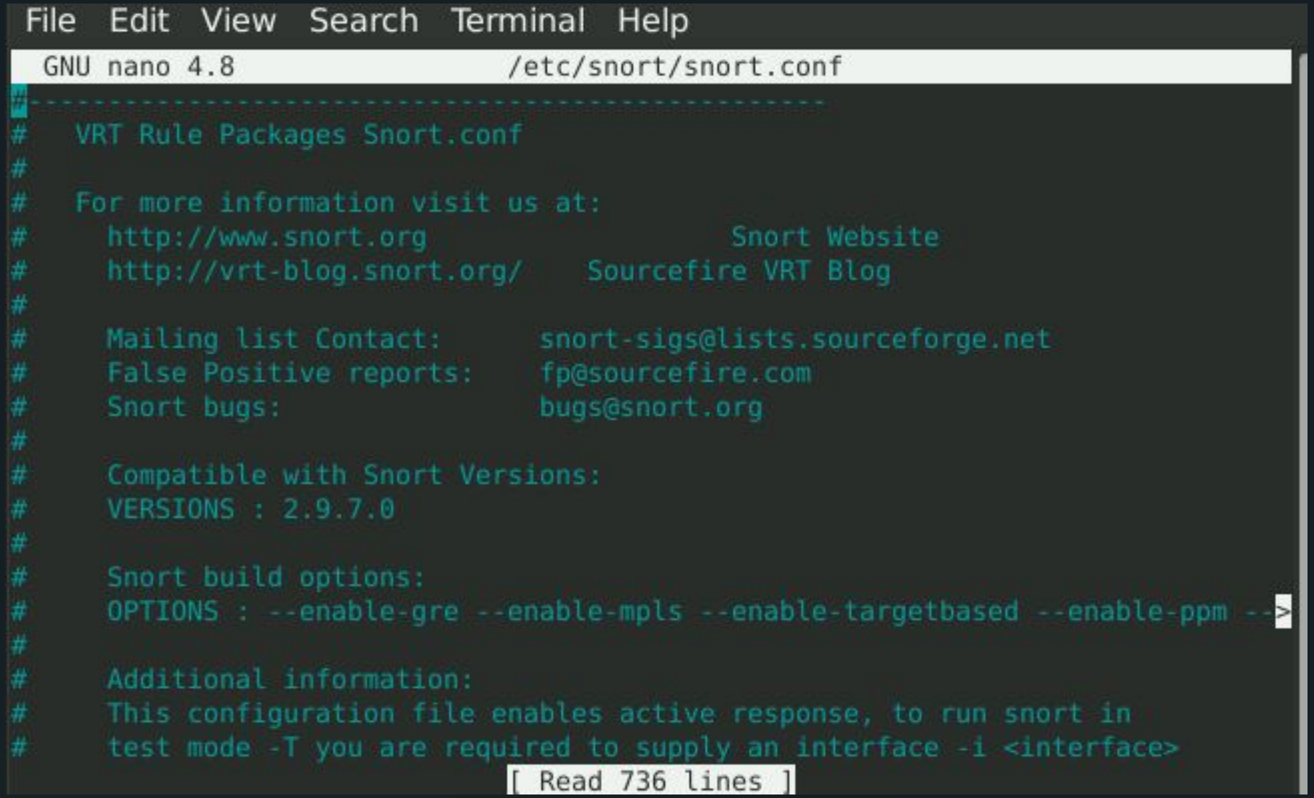
```
sysadmin@vm-image-ubuntu-dev-1:~$ sudo snort -A console -q -c /etc/snort/snort.c
onf -i eth0
12/10-05:36:39.805712  [**] [1:1000002:1] Nmap SYN scan detected [**] [Priority:
 0] {TCP} 10.12.4.38:54405 -> 10.12.4.8:21
12/10-05:36:44.025136  [**] [1:1000002:1] Nmap SYN scan detected [**] [Priority:
 0] {TCP} 10.12.4.38:54406 -> 10.12.4.22:1720
12/10-05:36:49.001897  [**] [1:1000002:1] Nmap SYN scan detected [**] [Priority:
 0] {TCP} 10.12.4.38:54406 -> 10.12.4.16:1145
12/10-05:36:54.047438  [**] [1:1000002:1] Nmap SYN scan detected [**] [Priority:
 0] {TCP} 10.12.4.38:54405 -> 10.12.4.25:2160
```



```
12/10-05:36:59.027933  [**] [1:1000002:1] Nmap SYN scan detected [**] [Priority: 0] {TCP} 10.12.4.38:54406 -> 10.12.4.24:9071
12/10-05:37:04.012442  [**] [1:1000002:1] Nmap SYN scan detected [**] [Priority: 0] {TCP} 10.12.4.38:54406 -> 10.12.4.21:898
12/10-05:37:09.001464  [**] [1:1000002:1] Nmap SYN scan detected [**] [Priority: 0] {TCP} 10.12.4.38:54405 -> 10.12.4.31:1137
12/10-05:37:11.178180  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.12.4.38:54405 -> 10.12.4.21:705
12/10-05:37:11.235244  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.12.4.38:54405 -> 10.12.4.18:705
12/10-05:37:11.273068  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.12.4.38:54405 -> 10.12.4.17:705
12/10-05:37:11.277859  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.12.4.38:54405 -> 10.12.4.4:705
12/10-05:37:11.285309  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.12.4.38:54406 -> 10.12.4.21:705
12/10-05:37:11.285570  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.12.4.38:54405 -> 10.12.4.29:705
12/10-05:37:11.339451  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.12.4.38:54406 -> 10.12.4.18:705
12/10-05:37:11.341882  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.12.4.38:54405 -> 10.12.4.32:705
12/10-05:37:11.341941  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.12.4.38:54405 -> 10.12.4.1:705
12/10-05:37:11.374701  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.12.4.38:54406 -> 10.12.4.17:705
12/10-05:37:11.379370  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.12.4.38:54406 -> 10.12.4.4:705
12/10-05:37:11.388658  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.12.4.38:54406 -> 10.12.4.29:705
12/10-05:37:11.440075  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.12.4.38:54405 -> 10.12.4.9:705
12/10-05:37:11.442411  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.12.4.38:54406 -> 10.12.4.1:705
12/10-05:37:11.442810  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.12.4.38:54406 -> 10.12.4.32:705
12/10-05:37:11.475432  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.12.4.38:54405 -> 10.12.4.39:705
```

Intrusion Prevention

You can configure your Snort into an IPS.



```
  GNU nano 4.8                 /etc/snort/snort.conf
#--------------------------------------------------
#   VRT Rule Packages Snort.conf
#
#   For more information visit us at:
#     http://www.snort.org                   Snort Website
#     http://vrt-blog.snort.org/    Sourcefire VRT Blog
#
#     Mailing list Contact:      snort-sigs@lists.sourceforge.net
#     False Positive reports:    fp@sourcefire.com
#     Snort bugs:                bugs@snort.org
#
#     Compatible with Snort Versions:
#     VERSIONS : 2.9.7.0
#
#     Snort build options:
#     OPTIONS : --enable-gre --enable-mpls --enable-targetbased --enable-ppm -->
#
#     Additional information:
#     This configuration file enables active response, to run snort in
#     test mode -T you are required to supply an interface -i <interface>
                                [ Read 736 lines ]
```

Update Snort Configuration for Inline Mode

Must configure local.rules.

Configure the DAQ Module for Inline Mode

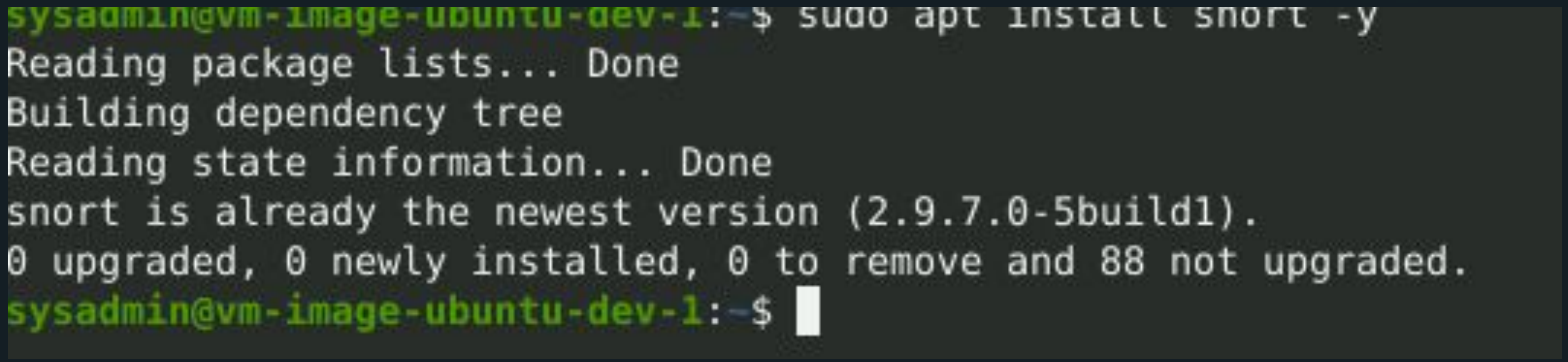
Must install afpacket DAQ module.

Add Rules to Drop Malicious Traffic

Must configure local.rules to block malicious traffic.

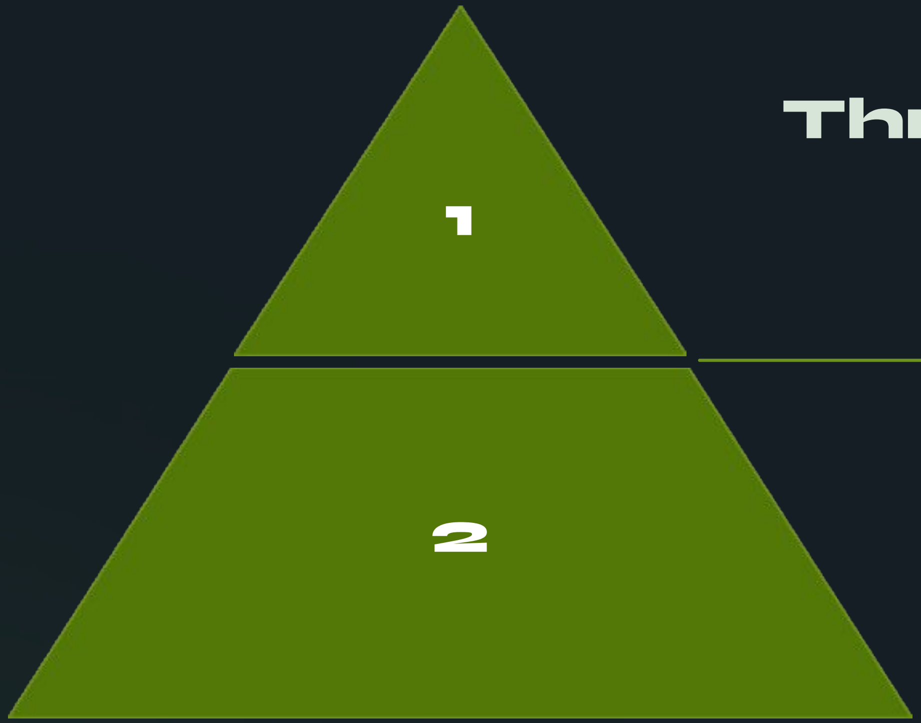
Test the IPS Setup

This helps ensure you have adjusted everything to how it goes.



```
sysadmin@vm-image-ubuntu-dev-1:~$ sudo apt install snort -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
snort is already the newest version (2.9.7.0-5build1).
0 upgraded, 0 newly installed, 0 to remove and 88 not upgraded.
sysadmin@vm-image-ubuntu-dev-1:~$
```

Security Onion: IDS (Precious)



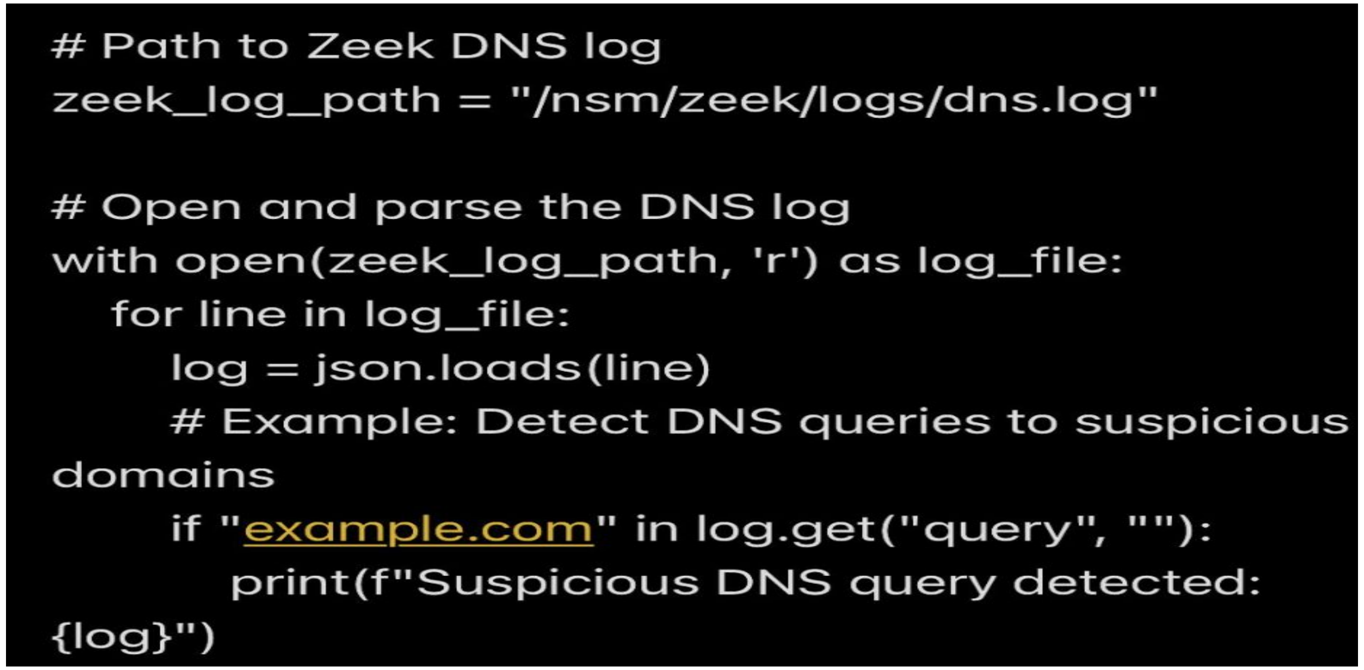
Threat Detection

Threat Prevention

Threat Detection

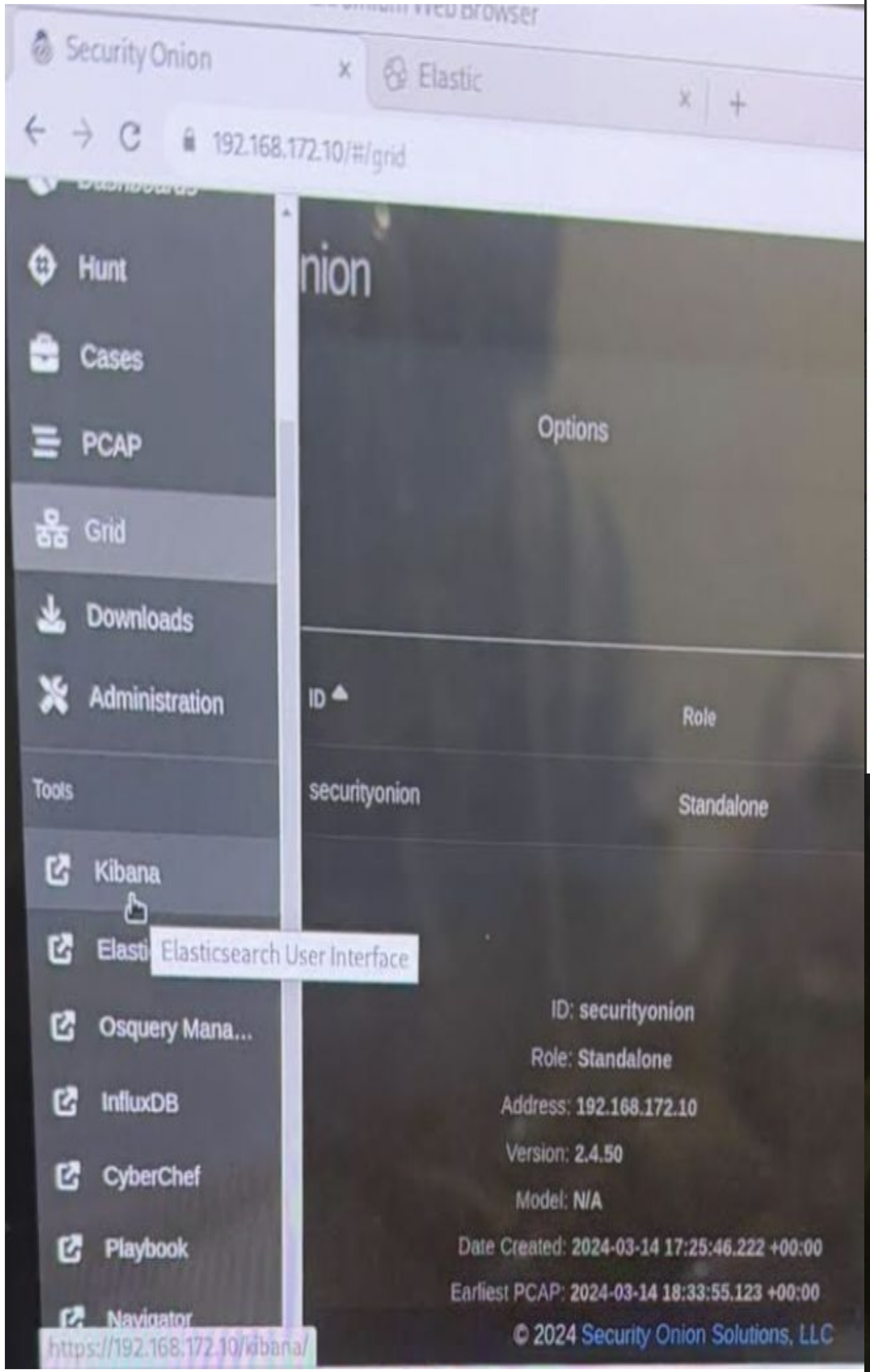
Security Onion uses kibana as its front-end for viewing secure data, there you'll be able to view pre-configured kibana dashboards that visualize traffic data, alerts and logs from suricata, zeek and other tools
Suricata(IDS/IPS) monitors network traffic and generates alerts for known attack patterns or suspicious behavior.
Zeek monitors network traffic, providing detailed logs and metadata about connections, protocols, and anomalous events.

how automating threat detection with zeek logs could look like:



```
# Path to Zeek DNS log
zeek_log_path = "/nsm/zeek/logs/dns.log"

# Open and parse the DNS log
with open(zeek_log_path, 'r') as log_file:
    for line in log_file:
        log = json.loads(line)
        # Example: Detect DNS queries to suspicious
domains
        if "example.com" in log.get("query", ""):
            print(f"Suspicious DNS query detected:
{log}")
```

This script reads Zeek DNS logs and flags queries to suspicious domains(like example.com). You can expand this logic to detect other anomalies such as unusual traffic patterns.



ID	Role
securityonion	Standalone

ID: securityonion

Role: Standalone

Address: 192.168.172.10

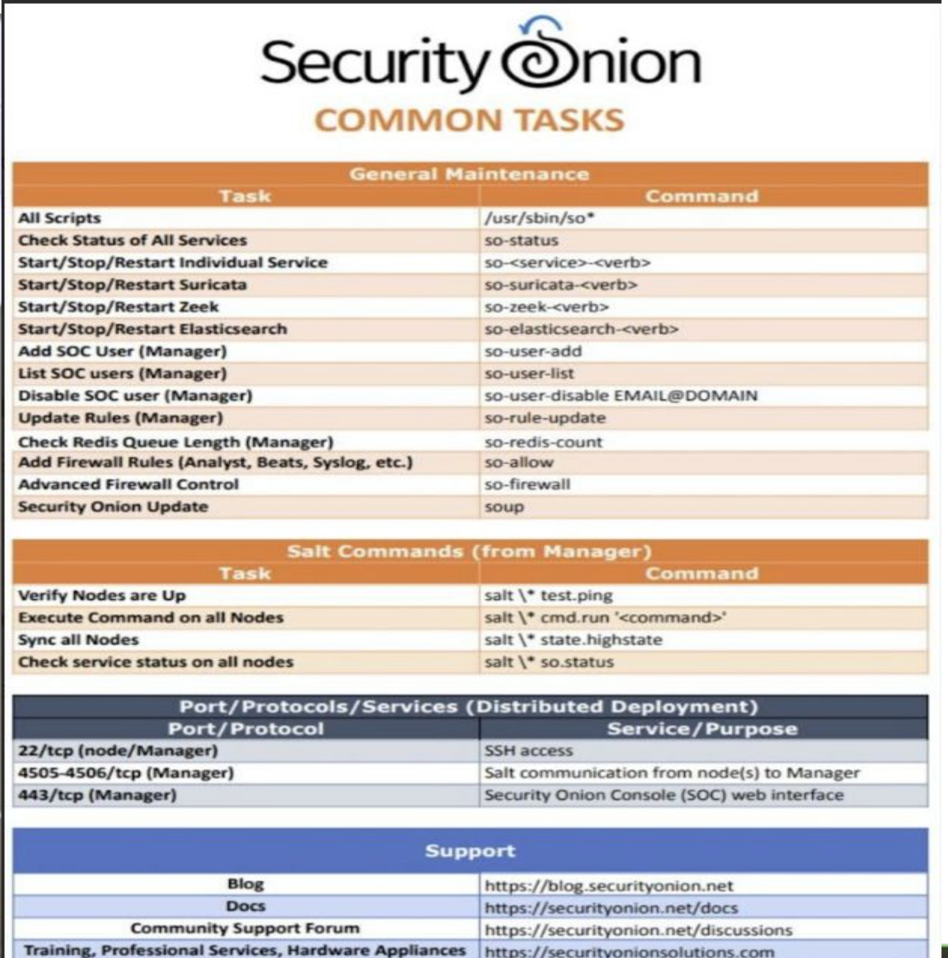
Version: 2.4.50

Model: N/A

Date Created: 2024-03-14 17:25:46.222 +00:00

Earliest PCAP: 2024-03-14 18:33:55.123 +00:00

© 2024 Security Onion Solutions, LLC



Security Onion

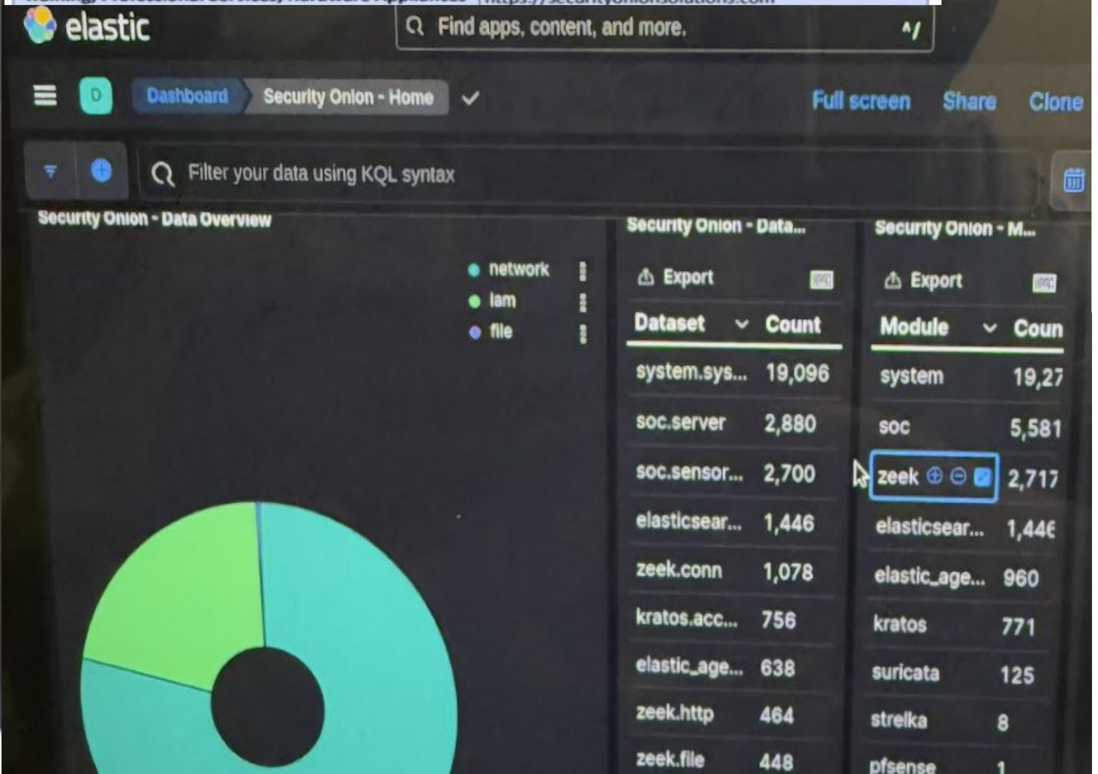
COMMON TASKS

General Maintenance	
Task	**Command**
All Scripts	/usr/sbin/so*
Check Status of All Services	so-status
Start/Stop/Restart Individual Service	so-<service>-<verb>
Start/Stop/Restart Suricata	so-suricata-<verb>
Start/Stop/Restart Zeek	so-zeek-<verb>
Start/Stop/Restart Elasticsearch	so-elasticsearch-<verb>
Add SOC User (Manager)	so-user-add
List SOC users (Manager)	so-user-list
Disable SOC user (Manager)	so-user-disable EMAIL@DOMAIN
Update Rules (Manager)	so-rule-update
Check Redis Queue Length (Manager)	so-redis-count
Add Firewall Rules (Analyst, Beats, Syslog, etc.)	so-allow
Advanced Firewall Control	so-firewall
Security Onion Update	soup

Salt Commands (from Manager)	
Task	**Command**
Verify Nodes are Up	salt * test.ping
Execute Command on all Nodes	salt * cmd.run '<command>'
Sync all Nodes	salt * state.highstate
Check service status on all nodes	salt * so.status

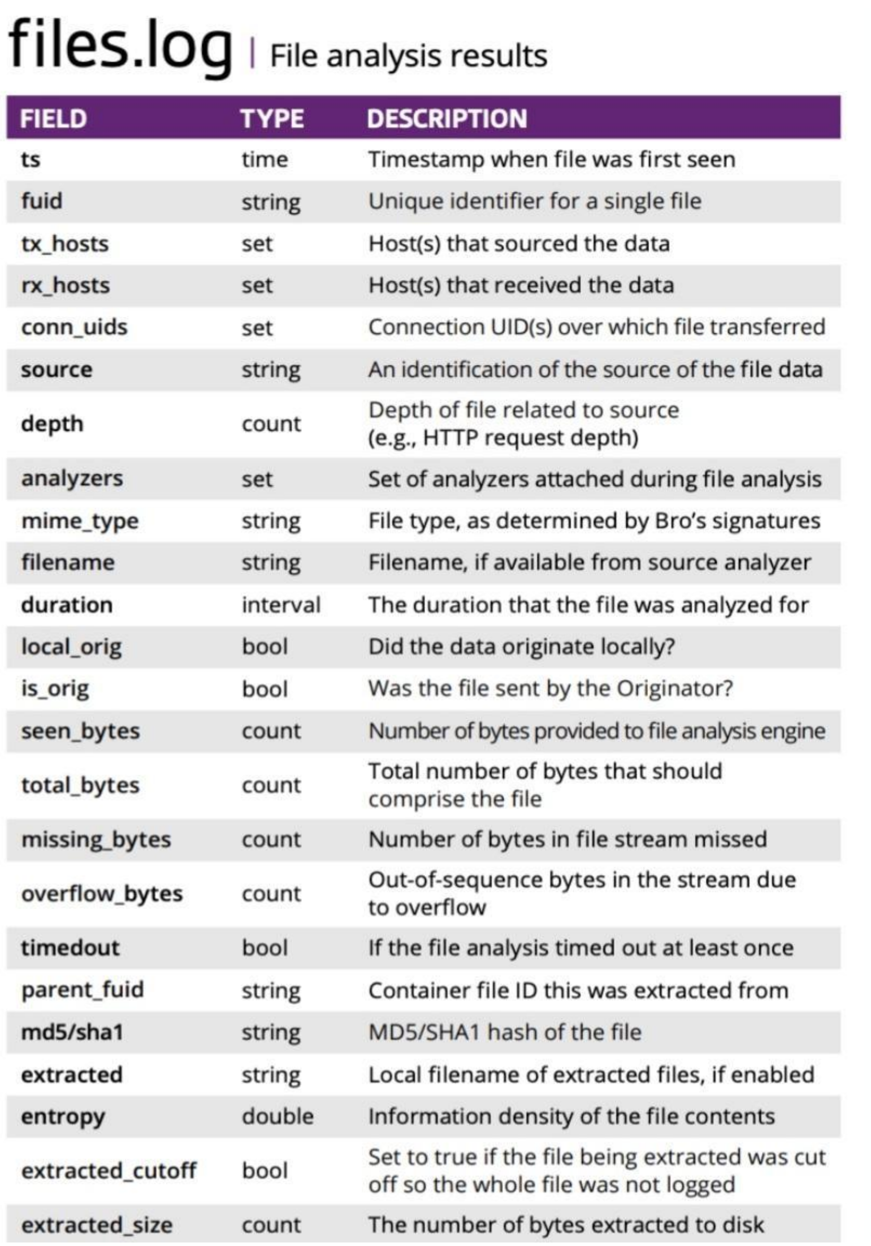
Port/Protocols/Services (Distributed Deployment)	
Port/Protocol	**Service/Purpose**
22/tcp (node/Manager)	SSH access
4505-4506/tcp (Manager)	Salt communication from node(s) to Manager
443/tcp (Manager)	Security Onion Console (SOC) web interface

Support	
Blog	https://blog.securityonion.net
Docs	https://securityonion.net/docs
Community Support Forum	https://securityonion.net/discussions
Training, Professional Services, Hardware Appliances	https://securityonionsolutions.com



Dataset	Count
system.sys...	19,096
soc.server	2,880
soc.sensor...	2,700
elasticsear...	1,446
zeek.conn	1,078
kratos.acc...	756
elastic_age...	638
zeek.http	464
zeek.file	448

Module	Coun
system	19,27
soc	5,581
zeek	2,717
elasticsear...	1,446
elastic_age...	960
kratos	771
suricata	125
strelka	8
pfsense	1



files.log | File analysis results

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when file was first seen
fuid	string	Unique identifier for a single file
tx_hosts	set	Host(s) that sourced the data
rx_hosts	set	Host(s) that received the data
conn_uids	set	Connection UID(s) over which file transferred
source	string	An identification of the source of the file data
depth	count	Depth of file related to source (e.g., HTTP request depth)
analyzers	set	Set of analyzers attached during file analysis
mime_type	string	File type, as determined by Bro's signatures
filename	string	Filename, if available from source analyzer
duration	interval	The duration that the file was analyzed for
local_orig	bool	Did the data originate locally?
is_orig	bool	Was the file sent by the Originator?
seen_bytes	count	Number of bytes provided to file analysis engine
total_bytes	count	Total number of bytes that should comprise the file
missing_bytes	count	Number of bytes in file stream missed
overflow_bytes	count	Out-of-sequence bytes in the stream due to overflow
timedout	bool	If the file analysis timed out at least once
parent_fuid	string	Container file ID this was extracted from
md5/sha1	string	MD5/SHA1 hash of the file
extracted	string	Local filename of extracted files, if enabled
entropy	double	Information density of the file contents
extracted_cutoff	bool	Set to true if the file being extracted was cut off so the whole file was not logged
extracted_size	count	The number of bytes extracted to disk

Threat Prevention

- Block malicious IP Addresses using iptables
- Implementation of strong access controls
- Identification of potential risks
- Encrypt data
- Regular monitoring and auditing
- Ensure compliance with data privacy regulations while designing automation solutions

QUESTIONS?