

[Open in app](#)Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)

Snort Challenge — Live Attack

8 min read · Aug 28, 2025



Saeed Ahmed

Introduction

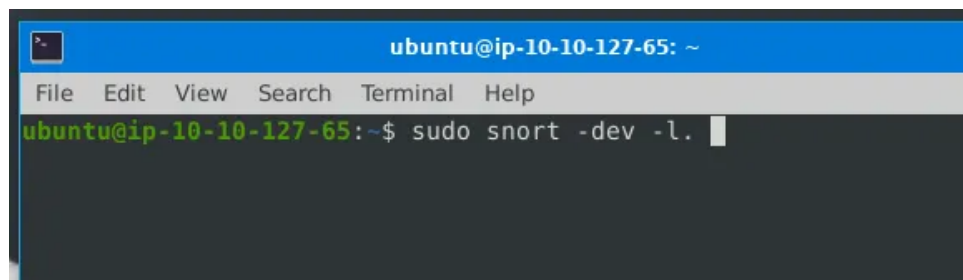
This challenge on TryHackMe is to investigate a series of traffic data and neutralise malicious activity across two distinct scenarios. Using Snort to analyse both live and captured traffic to identify and stop threats. This challenge uses two dedicated Virtual Machines, one for each task, which are accessible directly through the platform's split-screen view, eliminating the need for SSH or RDP connections.

Scenario 1 | Brute-Force

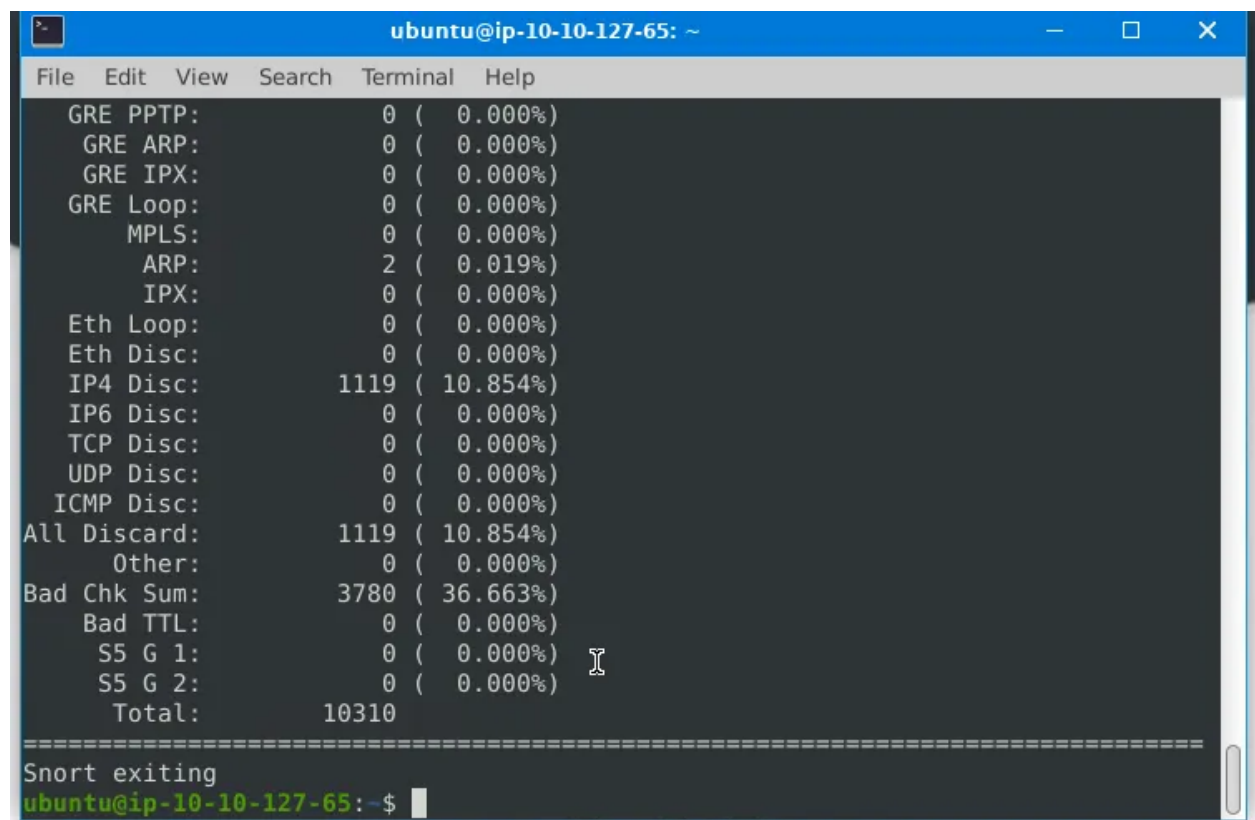
J&Y Enterprise, a world-renowned tech-coffee retailer, is facing a critical security threat to its latest recipe, "Shot4J," whose secret formula is stored in a digital safe. After experiencing multiple cyberattacks, they have hired me to protect their valuable digital assets. The AI assistant, J.A.V.A., has just alerted you to an active brute-force attack. My mission is to use the provided VM to first start Snort in sniffer mode to identify the attack's source, the targeted service, and the port. Once you have analysed the traffic, I will write and test an IPS rule with the ``-A console`` mode to stop the brute-force attempt. To complete the objective and receive the flag on your desktop, I must then implement this rule by running Snort in IPS mode with ``-A full``, using the default log path, and ensuring the malicious traffic is blocked for at least one minute.

A Snort command was executed with elevated privileges (sudo), incorporating flags for verbose output and packet logging. The inclusion of the -dev flags (-d for payload data, -e for the link layer, and -v for verbose output) indicates an intention to capture detailed information about each packet.

Concurrently, the -l . flag directed Snort to operate primarily in packet logger mode. This configuration stipulates that captured packets will not be displayed live on the console but will instead be saved to a log file within the current directory (.), thereby enabling subsequent detailed and persistent analysis.



```
ubuntu@ip-10-10-127-65: ~  
File Edit View Search Terminal Help  
ubuntu@ip-10-10-127-65:~$ sudo snort -dev -l.
```

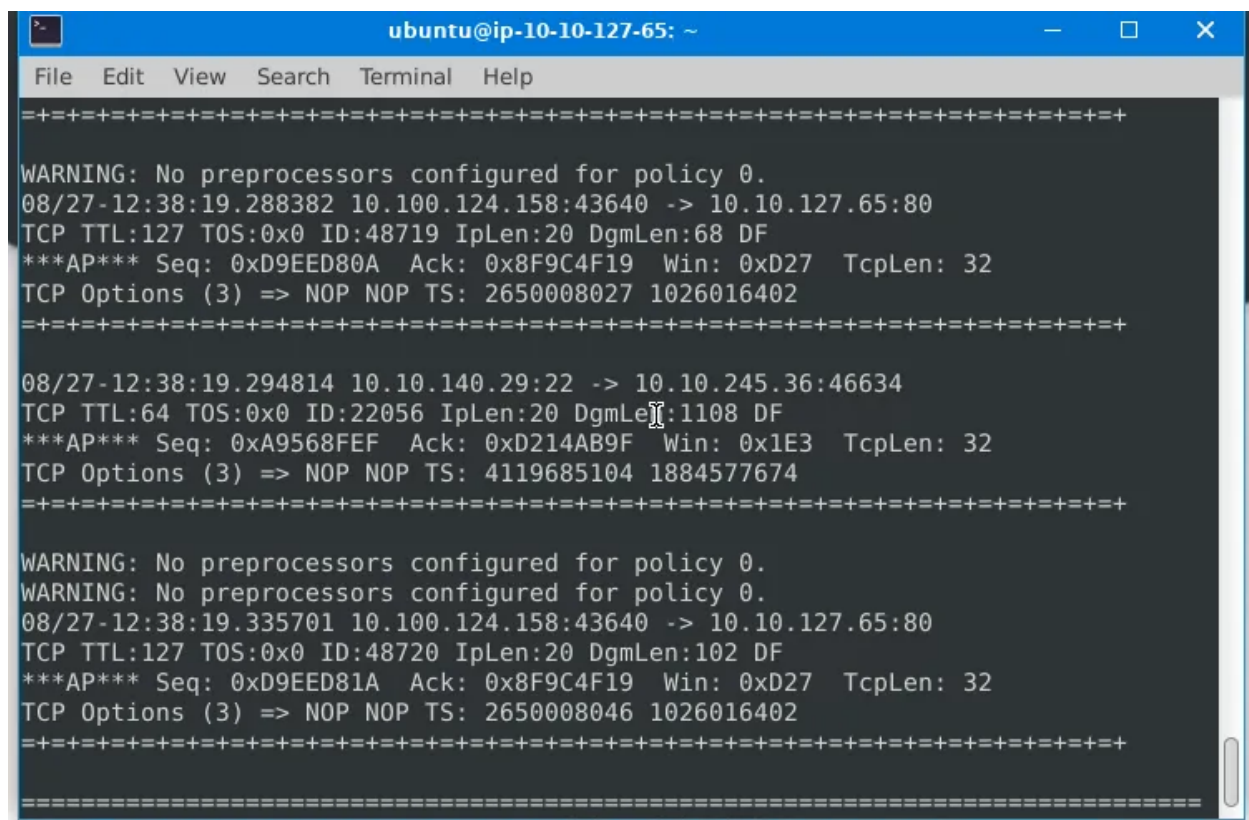


```
ubuntu@ip-10-10-127-65: ~  
File Edit View Search Terminal Help  
GRE PTP: 0 ( 0.000%)  
GRE ARP: 0 ( 0.000%)  
GRE IPX: 0 ( 0.000%)  
GRE Loop: 0 ( 0.000%)  
MPLS: 0 ( 0.000%)  
ARP: 2 ( 0.019%)  
IPX: 0 ( 0.000%)  
Eth Loop: 0 ( 0.000%)  
Eth Disc: 0 ( 0.000%)  
IP4 Disc: 1119 ( 10.854%)  
IP6 Disc: 0 ( 0.000%)  
TCP Disc: 0 ( 0.000%)  
UDP Disc: 0 ( 0.000%)  
ICMP Disc: 0 ( 0.000%)  
All Discard: 1119 ( 10.854%)  
Other: 0 ( 0.000%)  
Bad Chk Sum: 3780 ( 36.663%)  
Bad TTL: 0 ( 0.000%)  
S5 G 1: 0 ( 0.000%)  
S5 G 2: 0 ( 0.000%)  
Total: 10310  
=====
```

Protocol	Count	Percentage
GRE PTP	0	0.000%
GRE ARP	0	0.000%
GRE IPX	0	0.000%
GRE Loop	0	0.000%
MPLS	0	0.000%
ARP	2	0.019%
IPX	0	0.000%
Eth Loop	0	0.000%
Eth Disc	0	0.000%
IP4 Disc	1119	10.854%
IP6 Disc	0	0.000%
TCP Disc	0	0.000%
UDP Disc	0	0.000%
ICMP Disc	0	0.000%
All Discard	1119	10.854%
Other	0	0.000%
Bad Chk Sum	3780	36.663%
Bad TTL	0	0.000%
S5 G 1	0	0.000%
S5 G 2	0	0.000%
Total	10310	

```
Snort exiting  
ubuntu@ip-10-10-127-65:~$
```

Following the execution of the Snort command in packet logger mode, the `ls` command was utilised to verify the creation of the log file, identified as `snort.log.1756298222`. To proceed with the analysis, this file was subsequently opened using Snort's read-back mode via the command `sudo snort -r snort.log.1756298222`. This action facilitates a comprehensive, packet-by-packet examination of the previously captured network traffic, furnishing the raw data essential for identifying anomalies and understanding communication patterns.



```
ubuntu@ip-10-10-127-65: ~
File Edit View Search Terminal Help

=====
WARNING: No preprocessors configured for policy 0.
08/27-12:38:19.288382 10.100.124.158:43640 -> 10.10.127.65:80
TCP TTL:127 TOS:0x0 ID:48719 IpLen:20 DgmLen:68 DF
***AP*** Seq: 0xD9EED80A Ack: 0x8F9C4F19 Win: 0xD27 TcpLen: 32
TCP Options (3) => NOP NOP TS: 2650008027 1026016402
=====

08/27-12:38:19.294814 10.10.140.29:22 -> 10.10.245.36:46634
TCP TTL:64 TOS:0x0 ID:22056 IpLen:20 DgmLen:1108 DF
***AP*** Seq: 0xA9568FEF Ack: 0xD214AB9F Win: 0x1E3 TcpLen: 32
TCP Options (3) => NOP NOP TS: 4119685104 1884577674
=====

WARNING: No preprocessors configured for policy 0.
WARNING: No preprocessors configured for policy 0.
08/27-12:38:19.335701 10.100.124.158:43640 -> 10.10.127.65:80
TCP TTL:127 TOS:0x0 ID:48720 IpLen:20 DgmLen:102 DF
***AP*** Seq: 0xD9EED81A Ack: 0x8F9C4F19 Win: 0xD27 TcpLen: 32
TCP Options (3) => NOP NOP TS: 2650008046 1026016402
=====
=====
```

To better understand the suspicious outbound connection, the captured log file (snort.log.1756298222) was filtered specifically for traffic involving port 22. Using the command `sudo snort -r snort.log.1756298222 -n 10 port 22`, the analysis was narrowed to the first ten packets matching this criterion. This targeted approach allows for a focused examination of the communication pattern on the port of interest, helping to quickly confirm the nature of the suspicious activity without reviewing irrelevant data.

```
ubuntu@ip-10-10-127-65:~$ sudo snort -r snort.log.1756298222 port 22 -n 10
Exiting after 10 packets
Running in packet dump mode
```

```

ubuntu@ip-10-10-127-65: ~
File Edit View Search Terminal Help
***AP*** Seq: 0x18E82855 Ack: 0xE3D46C4E Win: 0x1E1 TcpLen: 32
TCP Options (3) => NOP NOP TS: 1884573938 4119681369
=====
08/27-12:37:02.625332 10.10.140.29:22 -> 10.10.245.36:46604
TCP TTL:64 TOS:0x0 ID:55922 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xE3D46C4E Ack: 0x18E82899 Win: 0x1E3 TcpLen: 32
TCP Options (3) => NOP NOP TS: 4119681369 1884573938
=====
08/27-12:37:02.625343 10.10.140.29:22 -> 10.10.245.36:46590
TCP TTL:64 TOS:0x0 ID:9393 IpLen:20 DgmLen:104 DF
***AP*** Seq: 0xDDCD3509 Ack: 0x2EDC6677 Win: 0x1E3 TcpLen: 32
TCP Options (3) => NOP NOP TS: 4119681375 1884570301
=====
WARNING: No preprocessors configured for policy 0.
08/27-12:37:02.625353 10.10.245.36:46590 -> 10.10.140.29:22
TCP TTL:64 TOS:0x0 ID:54821 IpLen:20 DgmLen:52 DF
***A***F Seq: 0x2EDC6677 Ack: 0xDDCD353D Win: 0x1E1 TcpLen: 32
TCP Options (3) => NOP NOP TS: 1884573945 4119681375
=====
WARNING: No preprocessors configured for policy 0.

```

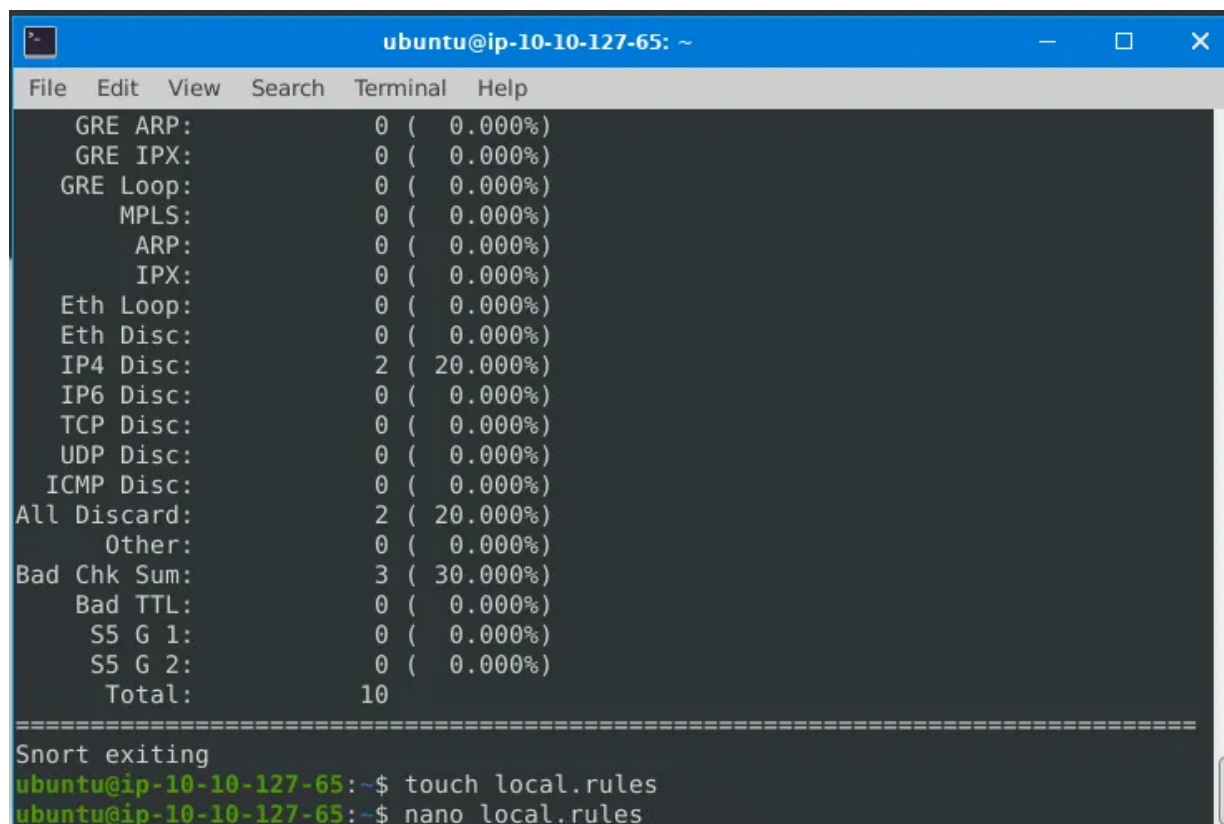
```

ubuntu@ip-10-10-127-65: ~
File Edit View Search Terminal Help
08/27-12:37:02.625232 10.10.127.65:80 -> 10.100.124.158:43640
TCP TTL:64 TOS:0x0 ID:9067 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x8F172F0A Ack: 0xD9EDB6C6 Win: 0x1BA TcpLen: 32
TCP Options (3) => NOP NOP TS: 1025939740 2649931308
=====
WARNING: No preprocessors configured for policy 0.
08/27-12:37:02.625265 10.100.124.158:43640 -> 10.10.127.65:80
TCP TTL:127 TOS:0x0 ID:44763 IpLen:20 DgmLen:68 DF
***AP*** Seq: 0xD9EDB6C6 Ack: 0x8F172F0A Win: 0xD27 TcpLen: 32
TCP Options (3) => NOP NOP TS: 2649931309 1025939661
=====
08/27-12:37:02.625276 10.10.127.65:80 -> 10.100.124.158:43640
TCP TTL:64 TOS:0x0 ID:9068 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x8F172F0A Ack: 0xD9EDB6D6 Win: 0x1BA TcpLen: 32
TCP Options (3) => NOP NOP TS: 1025939740 2649931309
=====
WARNING: No preprocessors configured for policy 0.
=====
Run time for packet processing was 0.632 seconds
Snort processed 10 packets.

```

In order to confirm that the suspicious traffic was indeed a brute-force attack, the `sudo snort -r snort.log.1756298222 -n 10 port 22` command was executed. This command specifically filtered the

captured snort.log.1756298222 file for the first ten packets involving port 22 to display only those lines containing “22”. The analysis revealed multiple SSH connection attempts originating from the same source IP address (10.10.231.10) targeting the same destination IP address (10.10.198.125). The repetition of “22” entries, combined with various status messages (e.g., “invalid user”, “disconnected”), strongly indicated a brute-force attempt against the SSH service.

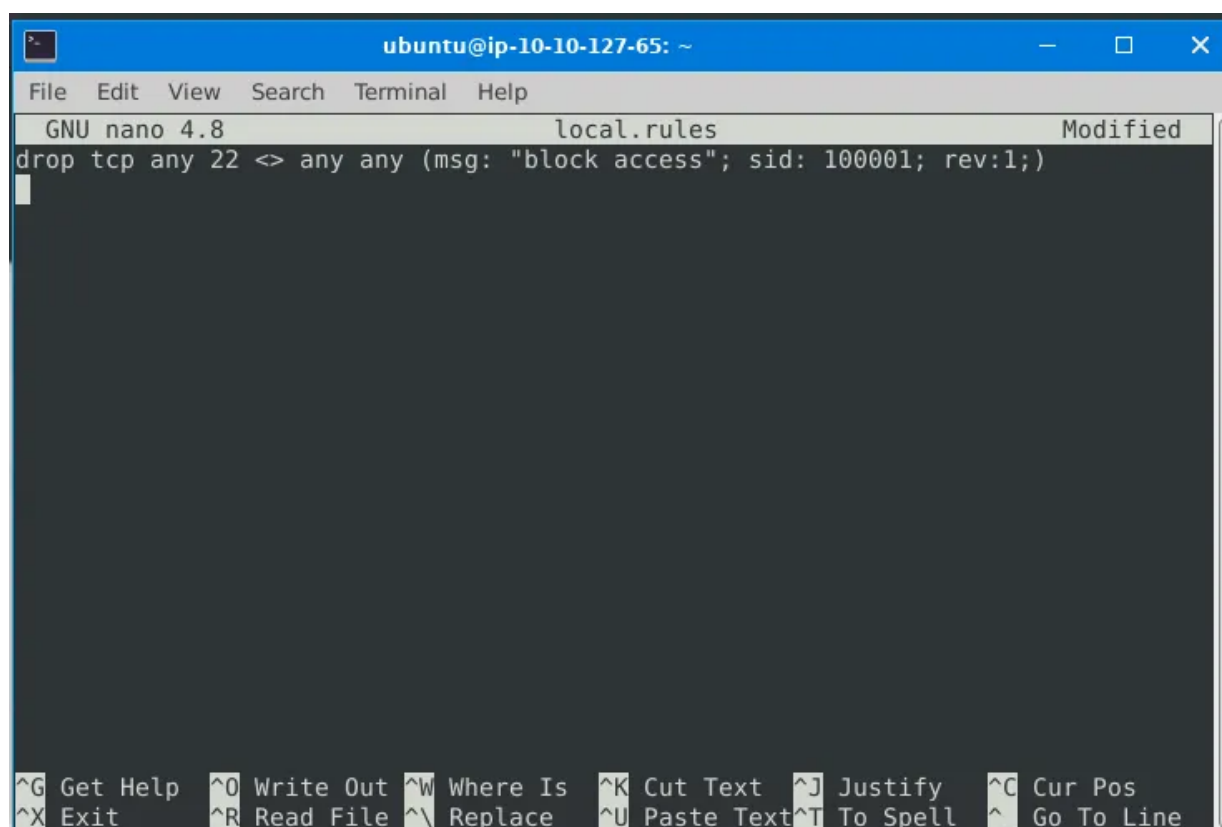


The screenshot shows a terminal window titled "ubuntu@ip-10-10-127-65: ~". The window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The main content area displays the following statistics:

Protocol	Count	Percentage
GRE ARP:	0	(0.000%)
GRE IPX:	0	(0.000%)
GRE Loop:	0	(0.000%)
MPLS:	0	(0.000%)
ARP:	0	(0.000%)
IPX:	0	(0.000%)
Eth Loop:	0	(0.000%)
Eth Disc:	0	(0.000%)
IP4 Disc:	2	(20.000%)
IP6 Disc:	0	(0.000%)
TCP Disc:	0	(0.000%)
UDP Disc:	0	(0.000%)
ICMP Disc:	0	(0.000%)
All Discard:	2	(20.000%)
Other:	0	(0.000%)
Bad Chk Sum:	3	(30.000%)
Bad TTL:	0	(0.000%)
S5 G 1:	0	(0.000%)
S5 G 2:	0	(0.000%)
Total:	10	

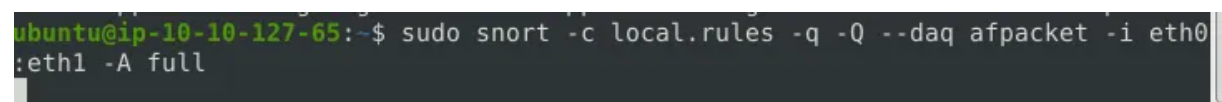
Below the statistics, the text "Snort exiting" is displayed. The terminal prompt shows the user has executed the following commands:

```
ubuntu@ip-10-10-127-65:~$ touch local.rules
ubuntu@ip-10-10-127-65:~$ nano local.rules
```



```
ubuntu@ip-10-10-127-65: ~  
File Edit View Search Terminal Help  
GNU nano 4.8 local.rules Modified  
drop tcp any 22 <> any any (msg: "block access"; sid: 100001; rev:1;)  
  
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos  
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line
```

Now that we know what port is being used we can create a rule which can be used to drop this port access. Now that we have successfully identified the specific port being utilised by the malicious activity, we possess the crucial information needed to implement a robust defensive measure. Our next logical step is to formulate a precise Snort rule designed to effectively block or “drop” any network traffic attempting to access this particular port. This rule will act as a digital barrier, preventing unauthorized connections and mitigating the potential for further exploitation or data exfiltration through this vulnerable point. By strategically deploying this rule, we can significantly enhance the security posture of our system and disrupt the attacker’s ability to maintain persistence or launch subsequent phases of their attack.



```
ubuntu@ip-10-10-127-65:~$ sudo snort -c local.rules -q -Q --daq afpacket -i eth0 :eth1 -A full
```

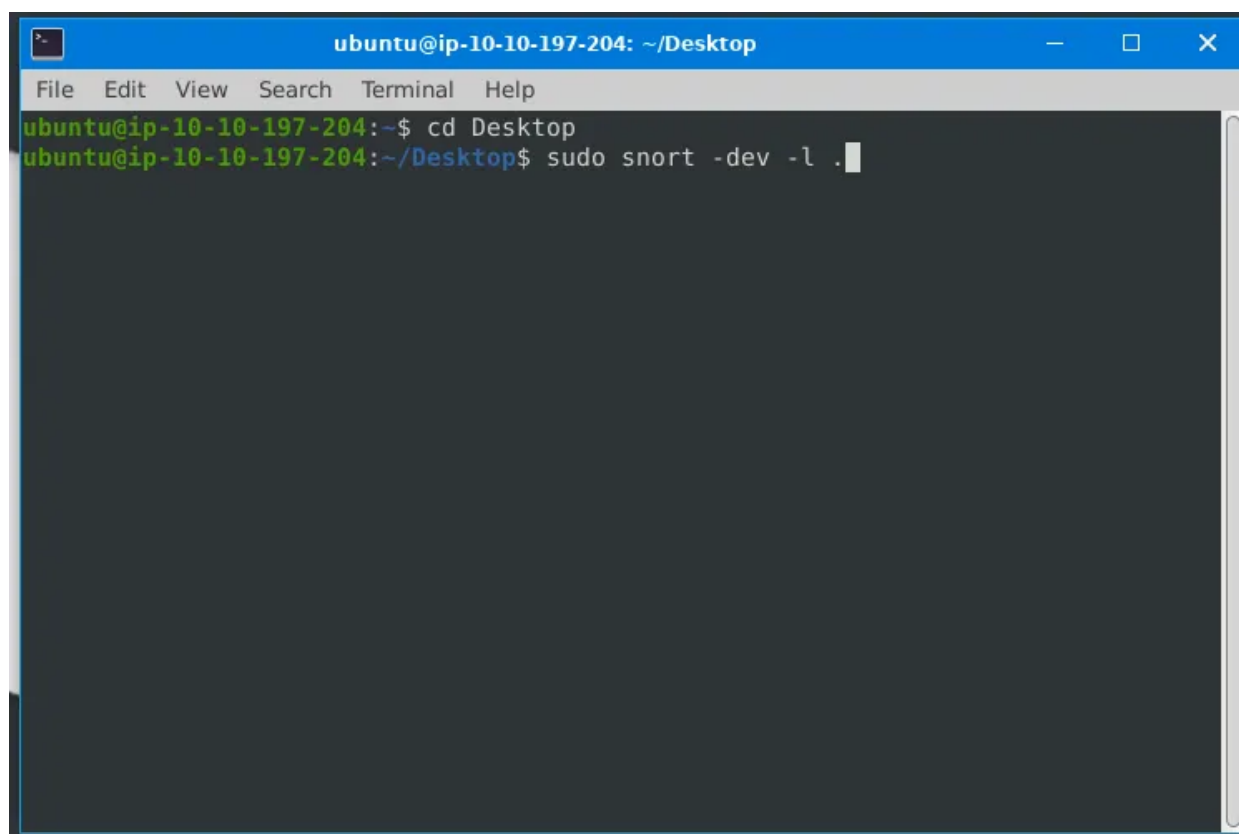
Running this command we can now rest ensure that the unauthorised access to the system has now been stopped.

Scenario 2 | Reverse-Shell

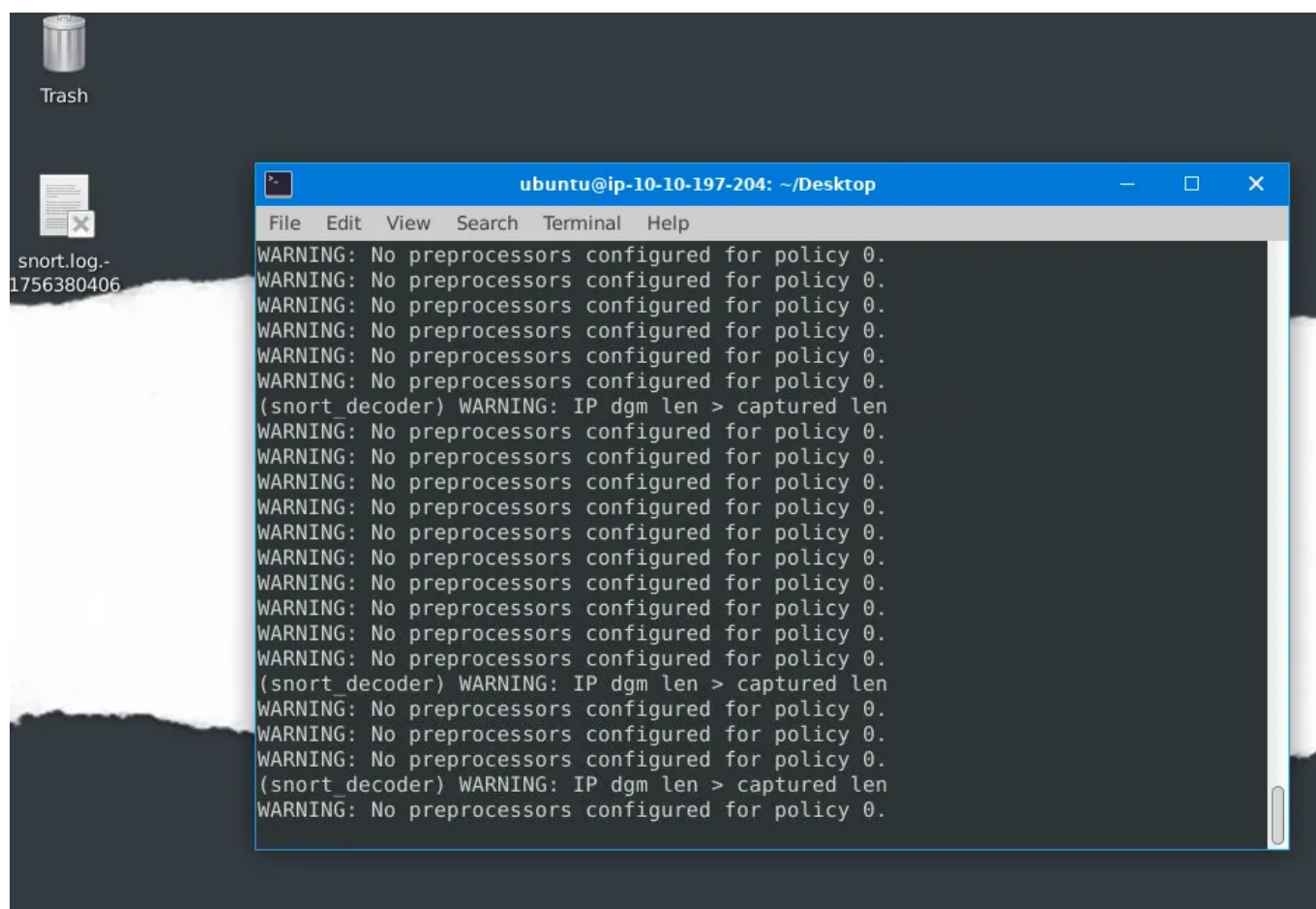
Upon successfully mitigating several inbound access attempts, the focus of the investigation pivoted to outbound network traffic. This proactive step was taken to address the significant risk of pre-existing compromises, such as those originating from insider threats or adversaries with a long dwell time. An initial scan of egress traffic immediately identified a persistent, anomalous connection. This activity

was assessed as being highly indicative of a reverse shell, suggesting a compromised host within the network was communicating with an external command-and-control (C2) server. The immediate next step is to use Snort for deep packet inspection to precisely identify the malicious traffic signature and subsequently develop a custom rule to block the unauthorised communication.

The investigation began by initiating a live network traffic capture on the host machine. The command `sudo snort -dev -l .` was executed from the Desktop directory, instructing Snort to run in packet-logging mode. This configuration captures all network packets, displays application-level data (-dev), and saves the entire session to a log file (-l .) in the current directory for subsequent, more detailed offline analysis.

A screenshot of a terminal window titled 'ubuntu@ip-10-10-197-204: ~/Desktop'. The terminal has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The command prompt shows the user has navigated to the Desktop directory and executed the command 'sudo snort -dev -l .'.

After the live packet capture was stopped, the `ls` command was used to confirm the creation of the log file, named `snort.log.1756380406`. To begin the inspection, this file was opened using Snort's read-back mode with the command `sudo snort -r snort.log.1756380406`. This function allows for a detailed, packet-by-packet review of the previously captured traffic, providing the raw data necessary to identify anomalies.



The initial review of the captured packets immediately revealed suspicious activity. The log output showed a successful TCP handshake and subsequent communication involving the IP address 10.10.144.156 on port 4444. This non-standard port, combined with persistent communication, strongly suggested the presence of an unauthorised connection, such as a reverse shell or a backdoor, communicating with an external entity.

```

ubuntu@ip-10-10-197-204: ~/Desktop
File Edit View Search Terminal Help
TCP TTL:64 TOS:0x0 ID:37992 IpLen:20 DgmLen:368 DF
***AP*** Seq: 0xDE9E6AA Ack: 0x5E6845FA Win: 0x1BA TcpLen: 32
TCP Options (3) => NOP NOP TS: 2229604568 527512238
=====
WARNING: No preprocessors configured for policy 0.
08/28-11:28:00.667888 10.100.107.200:52762 -> 10.10.197.204:80
TCP TTL:127 TOS:0x0 ID:40652 IpLen:20 DgmLen:68 DF
***AP*** Seq: 0x5E6845FA Ack: 0xDE9E7E6 Win: 0xA22 TcpLen: 32
TCP Options (3) => NOP NOP TS: 527512257 2229604568
=====

Run time for packet processing was 4.810176 seconds
Snort processed 13025 packets.
Snort ran for 0 days 0 hours 0 minutes 4 seconds
  Pkts/sec:          3256
=====

Memory usage summary:
  Total non-mmapped bytes (arena):      786432
  Bytes in mapped regions (hblkhd):     12906496
  Total allocated space (uordblks):     678160
  Total free space (fordblks):          108272
  Topmost releasable block (keepcost):  102224

```

```

ubuntu@ip-10-10-197-204: ~/Desktop
File Edit View Search Terminal Help
=====
WARNING: No preprocessors configured for policy 0.
08/28-11:28:00.586853 10.10.144.156:4444 -> 10.10.196.55:54120
TCP TTL:64 TOS:0x0 ID:9999 IpLen:20 DgmLen:60 DF
***A**S* Seq: 0x2D7325F9 Ack: 0xE99CA159 Win: 0xF4B3 TcpLen: 40
TCP Options (5) => MSS: 8961 SackOK TS: 1980336320 2358168621 NOP WS: 7
=====

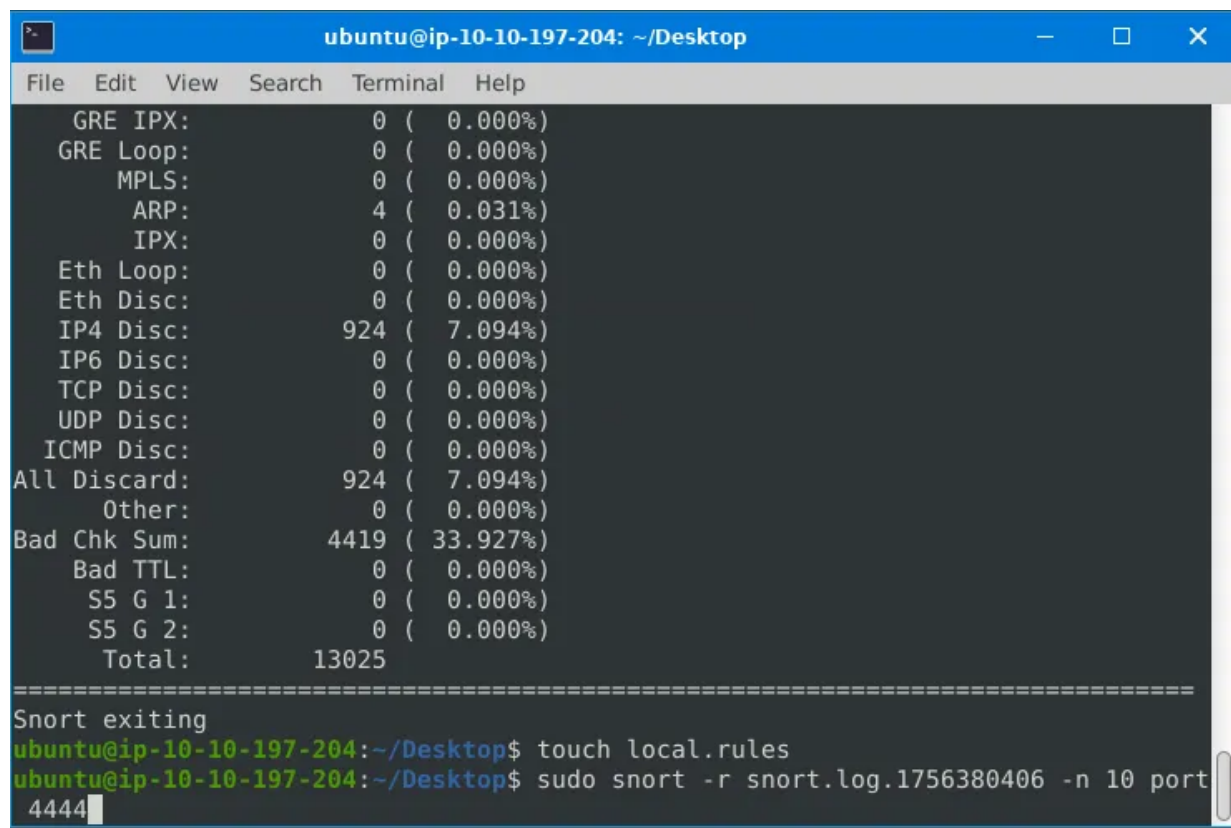
WARNING: No preprocessors configured for policy 0.
WARNING: No preprocessors configured for policy 0.
08/28-11:28:00.596554 10.10.196.55:54120 -> 10.10.144.156:4444
TCP TTL:64 TOS:0x0 ID:9999 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xE99CA159 Ack: 0x2D7325FA Win: 0x1EB TcpLen: 32
TCP Options (3) => NOP NOP TS: 2358168621 1980336320
=====

WARNING: No preprocessors configured for policy 0.
08/28-11:28:00.606450 10.100.107.200:52762 -> 10.10.197.204:80
TCP TTL:127 TOS:0x0 ID:40650 IpLen:20 DgmLen:68 DF
***AP*** Seq: 0x5E6845DA Ack: 0xDE9D87F Win: 0xA22 TcpLen: 32
TCP Options (3) => NOP NOP TS: 527512238 2229604549
=====

```

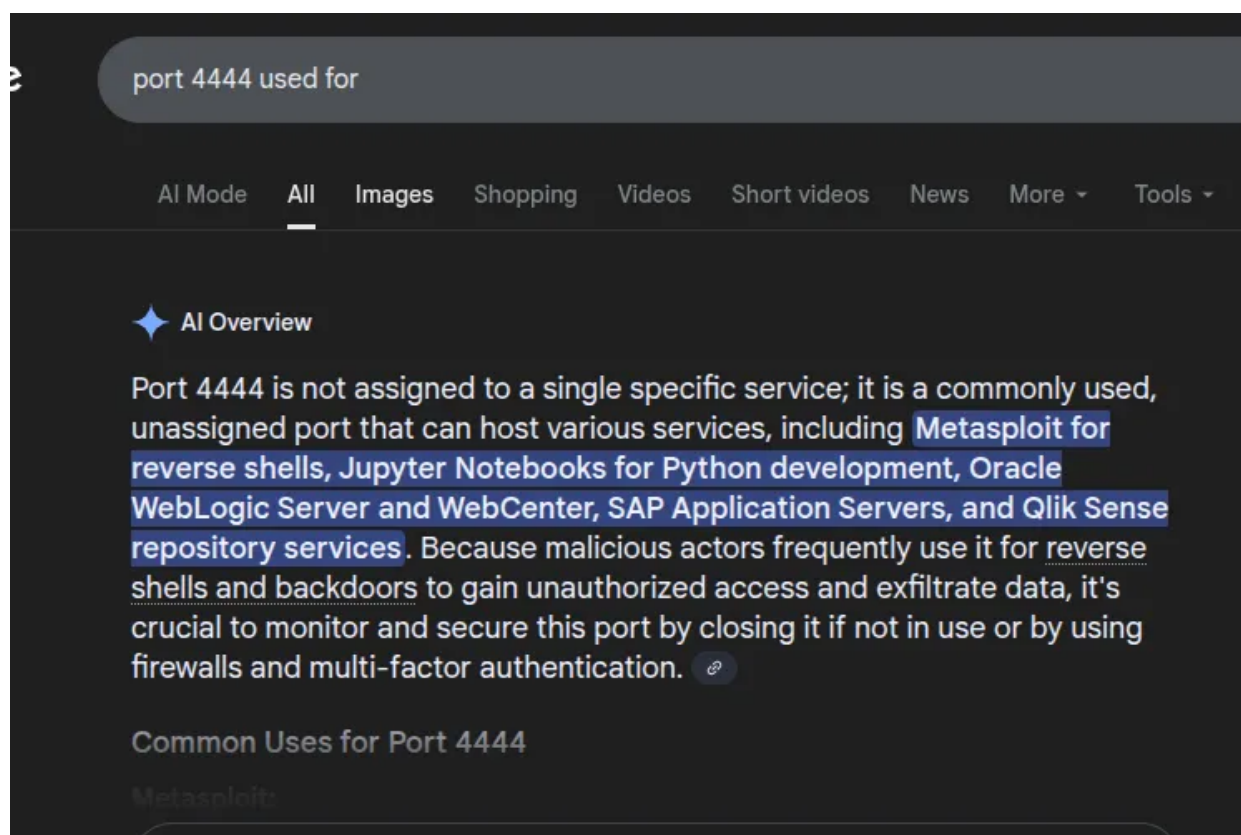
To better understand the suspicious outbound connection, the captured log file (snort.log.1756380406)

was filtered specifically for traffic involving port 4444. Using the command `sudo snort -r snort.log.1756380406 -n 10 port 4444`, the analysis was narrowed to the first ten packets matching this criterion. This targeted approach allows for a focused examination of the communication pattern on the port of interest, helping to quickly confirm the nature of the suspicious activity without reviewing irrelevant data.

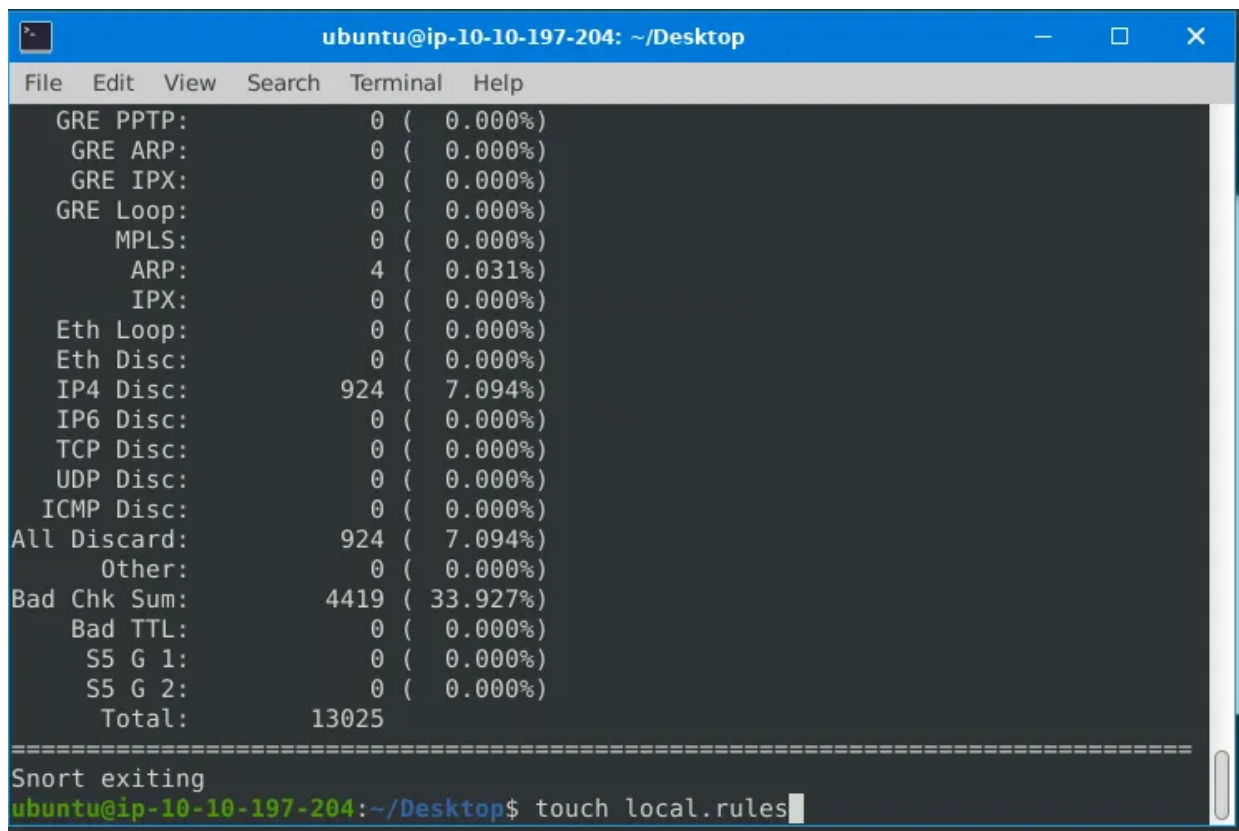


```
ubuntu@ip-10-10-197-204: ~/Desktop
File Edit View Search Terminal Help
GRE IPX: 0 ( 0.000%)
GRE Loop: 0 ( 0.000%)
MPLS: 0 ( 0.000%)
ARP: 4 ( 0.031%)
IPX: 0 ( 0.000%)
Eth Loop: 0 ( 0.000%)
Eth Disc: 0 ( 0.000%)
IP4 Disc: 924 ( 7.094%)
IP6 Disc: 0 ( 0.000%)
TCP Disc: 0 ( 0.000%)
UDP Disc: 0 ( 0.000%)
ICMP Disc: 0 ( 0.000%)
All Discard: 924 ( 7.094%)
Other: 0 ( 0.000%)
Bad Chk Sum: 4419 ( 33.927%)
Bad TTL: 0 ( 0.000%)
S5 G 1: 0 ( 0.000%)
S5 G 2: 0 ( 0.000%)
Total: 13025
=====
Snort exiting
ubuntu@ip-10-10-197-204:~/Desktop$ touch local.rules
ubuntu@ip-10-10-197-204:~/Desktop$ sudo snort -r snort.log.1756380406 -n 10 port
4444
```

To validate the hypothesis that port 4444 was being used for malicious purposes, external research was conducted. An online search confirmed that port 4444 is an unassigned port that is **commonly used by the Metasploit Framework as a default port for reverse shells and backdoors**. This intelligence strongly supports the conclusion that the observed traffic is a command-and-control channel from a compromised internal host.

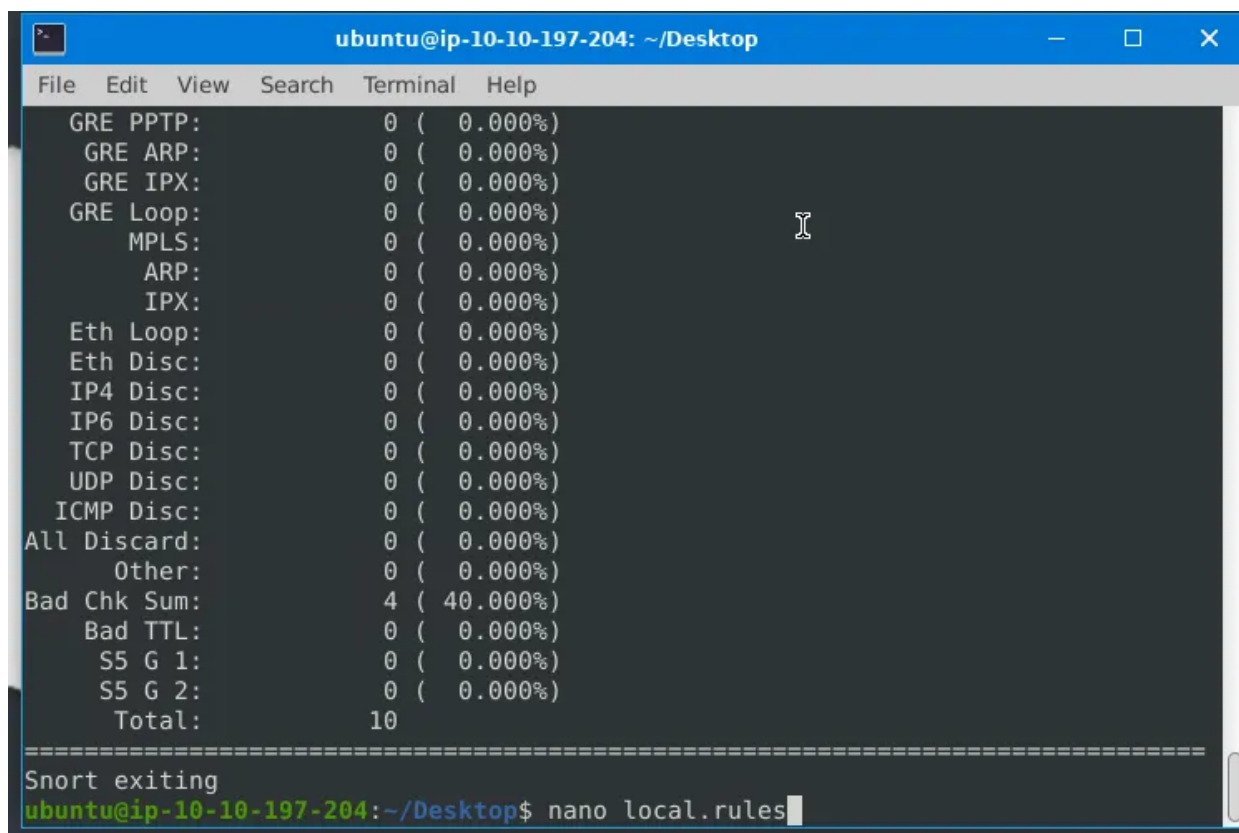


In preparation for actively mitigating the identified threat, a new file named `local.rules` was created with the `touch` command. This empty file serves as the standard location for writing custom Snort rules. The next step involves populating this file with a specific rule designed to detect and block the malicious traffic observed on port 4444, thereby operationalising the intelligence gathered from the packet analysis.



```
ubuntu@ip-10-10-197-204: ~/Desktop
File Edit View Search Terminal Help
GRE PPTP: 0 ( 0.000%)
GRE ARP: 0 ( 0.000%)
GRE IPX: 0 ( 0.000%)
GRE Loop: 0 ( 0.000%)
MPLS: 0 ( 0.000%)
ARP: 4 ( 0.031%)
IPX: 0 ( 0.000%)
Eth Loop: 0 ( 0.000%)
Eth Disc: 0 ( 0.000%)
IP4 Disc: 924 ( 7.094%)
IP6 Disc: 0 ( 0.000%)
TCP Disc: 0 ( 0.000%)
UDP Disc: 0 ( 0.000%)
ICMP Disc: 0 ( 0.000%)
All Discard: 924 ( 7.094%)
Other: 0 ( 0.000%)
Bad Chk Sum: 4419 ( 33.927%)
Bad TTL: 0 ( 0.000%)
S5 G 1: 0 ( 0.000%)
S5 G 2: 0 ( 0.000%)
Total: 13025
=====
Snort exiting
ubuntu@ip-10-10-197-204:~/Desktop$ touch local.rules
```

With the malicious traffic pattern identified and correlated with known threat actor TTPs, the next step was to create a custom detection signature. The nano local.rules command was used to open the previously created rules file in a text editor. The objective is to write a specific Snort rule that will alert on any inbound TCP traffic destined for the internal network on port 4444. This rule will serve as the active countermeasure, enabling the intrusion detection system to automatically identify and flag any future reverse shell connection attempts.

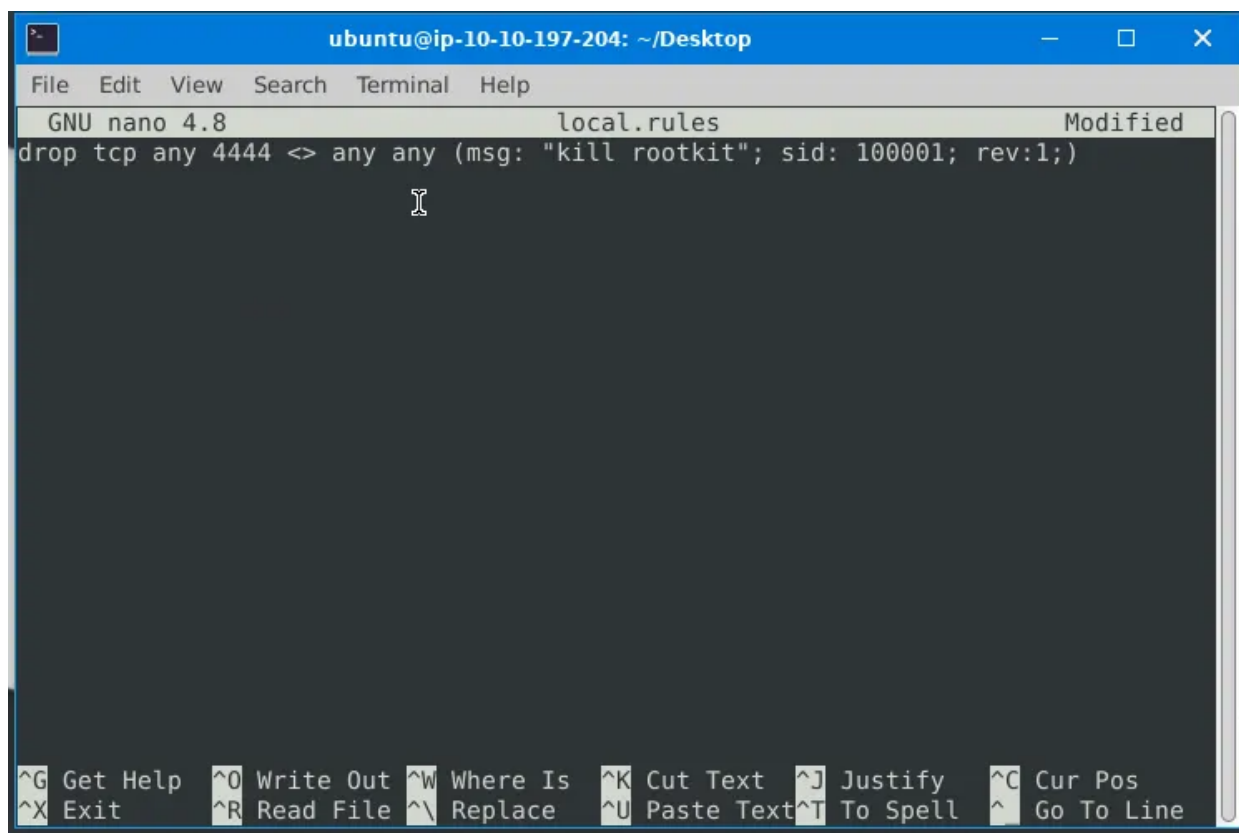


```
ubuntu@ip-10-10-197-204: ~/Desktop
File Edit View Search Terminal Help
GRE PPTP: 0 ( 0.000%)
GRE ARP: 0 ( 0.000%)
GRE IPX: 0 ( 0.000%)
GRE Loop: 0 ( 0.000%)
MPLS: 0 ( 0.000%)
ARP: 0 ( 0.000%)
IPX: 0 ( 0.000%)
Eth Loop: 0 ( 0.000%)
Eth Disc: 0 ( 0.000%)
IP4 Disc: 0 ( 0.000%)
IP6 Disc: 0 ( 0.000%)
TCP Disc: 0 ( 0.000%)
UDP Disc: 0 ( 0.000%)
ICMP Disc: 0 ( 0.000%)
All Discard: 0 ( 0.000%)
Other: 0 ( 0.000%)
Bad Chk Sum: 4 ( 40.000%)
Bad TTL: 0 ( 0.000%)
S5 G 1: 0 ( 0.000%)
S5 G 2: 0 ( 0.000%)
Total: 10
=====
Snort exiting
ubuntu@ip-10-10-197-204:~/Desktop$ nano local.rules
```

A custom Snort rule was authored and added to the `local.rules` file to actively block the malicious traffic. The following rule was implemented:

```
drop tcp any 4444 <> any any (msg:"kill rootkit"; sid:100001; rev:1;)
```

This rule is configured to act as an inline prevention measure. The drop action instructs Snort to discard any packet that matches the signature. It targets the tcp protocol and uses a bidirectional operator (<>) to block traffic where port 4444 is used as either the source or destination. This ensures the connection is severed regardless of direction. The rule includes a descriptive message ("kill rootkit") for logging purposes, a unique signature ID (sid:100001) for identification, and a revision number (rev:1). Once activated, this rule will effectively terminate the reverse shell communication channel.



```
ubuntu@ip-10-10-197-204: ~/Desktop
File Edit View Search Terminal Help
GNU nano 4.8 local.rules Modified
drop tcp any 4444 <> any any (msg: "kill rootkit"; sid: 100001; rev:1;)
I
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line
```

To deploy the countermeasure, an attempt was made to run Snort in an inline Intrusion Prevention System (IPS) mode using the afpacket DAQ module. The goal was to position Snort between two network interfaces, eth0 and eth1, to actively inspect and drop malicious traffic in real-time. The initial command executed was `sudo snort -c local.rules -Q — daq afpacket -i eth0:eth1 -A full`. However, this command failed to initialize, returning a fatal error: “Invalid interface specification.” This indicates that the DAQ module did not accept the syntax for defining the inline interface pair, necessitating a correction to the startup command before the drop rule could be successfully activated.

```
ubuntu@ip-10-10-197-204: ~/Desktop
File Edit View Search Terminal Help
Eth Disc: 0 ( 0.000%)
IP4 Disc: 0 ( 0.000%)
IP6 Disc: 0 ( 0.000%)
TCP Disc: 0 ( 0.000%)
UDP Disc: 0 ( 0.000%)
ICMP Disc: 0 ( 0.000%)
All Discard: 0 ( 0.000%)
Other: 0 ( 0.000%)
Bad Chk Sum: 4 ( 40.000%)
Bad TTL: 0 ( 0.000%)
S5 G 1: 0 ( 0.000%)
S5 G 2: 0 ( 0.000%)
Total: 10
=====
Snort exiting
ubuntu@ip-10-10-197-204:~/Desktop$ nano local.rules
ubuntu@ip-10-10-197-204:~/Desktop$ sudo snort -c local.rules -q -Q --daq afpacke
t -i eth0: eth1 -A full
ERROR: Can't initialize DAQ afpacket (-1) - afpacket_daq_initialize: Invalid int
erface specification: 'eth0:'!
Fatal Error, Quitting..
ubuntu@ip-10-10-197-204:~/Desktop$ nano local.rules
ubuntu@ip-10-10-197-204:~/Desktop$ sudo snort -c local.rules -q -Q --daq afpacke
t -i eth0:eth1 -A full
```

[Snort](#)[Tryhackme Snort](#)[Follow](#)

Written by Saeed Ahmed

1 follower · 13 following

No responses yet



Itsjustme

More from Saeed Ahmed

```
ate; apt-get install -y python3-flask'  
mpute/v1/projects/qwiklabs-gcp-00-fb0991939da5/zones/us-w  
  
qwiklabs-gcp-00-fb0991939da5)$ gcloud compute firewall-ru  
  
.0/0
```

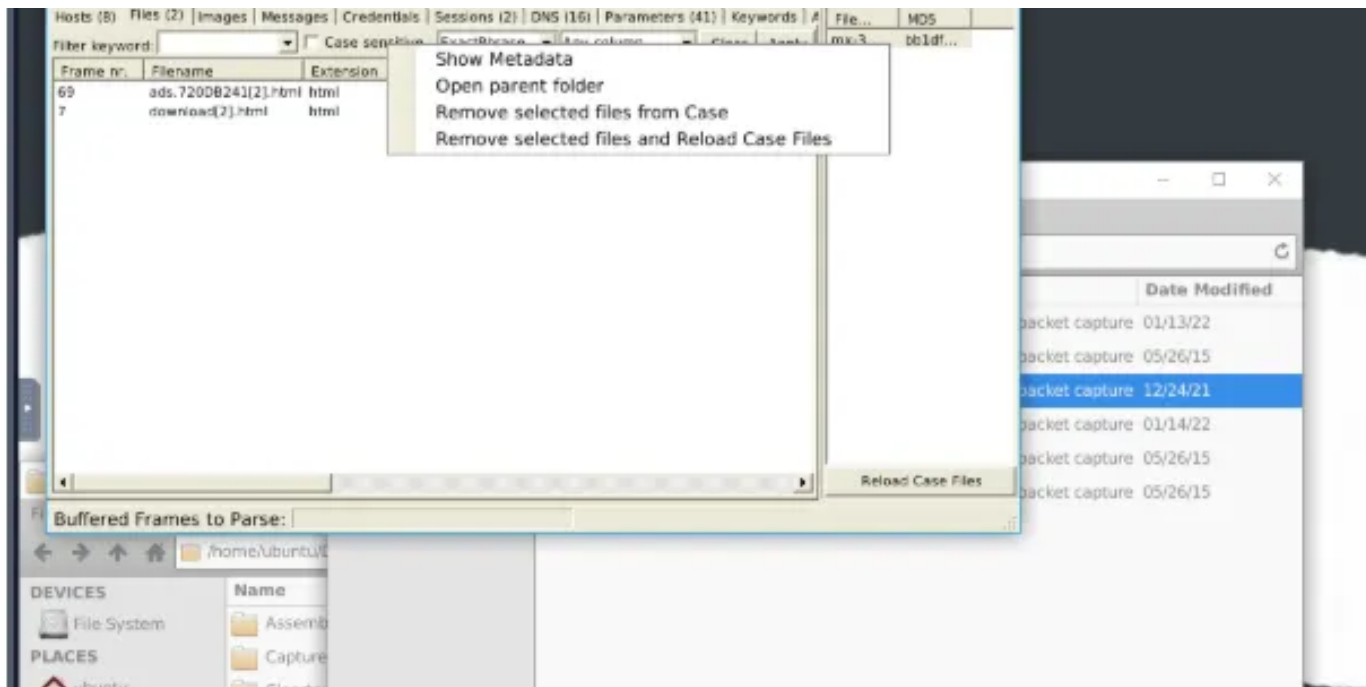



Saeed Ahmed

Identify vulnerabilities and remediation techniques

Vulnerability management overview

Nov 25, 2024

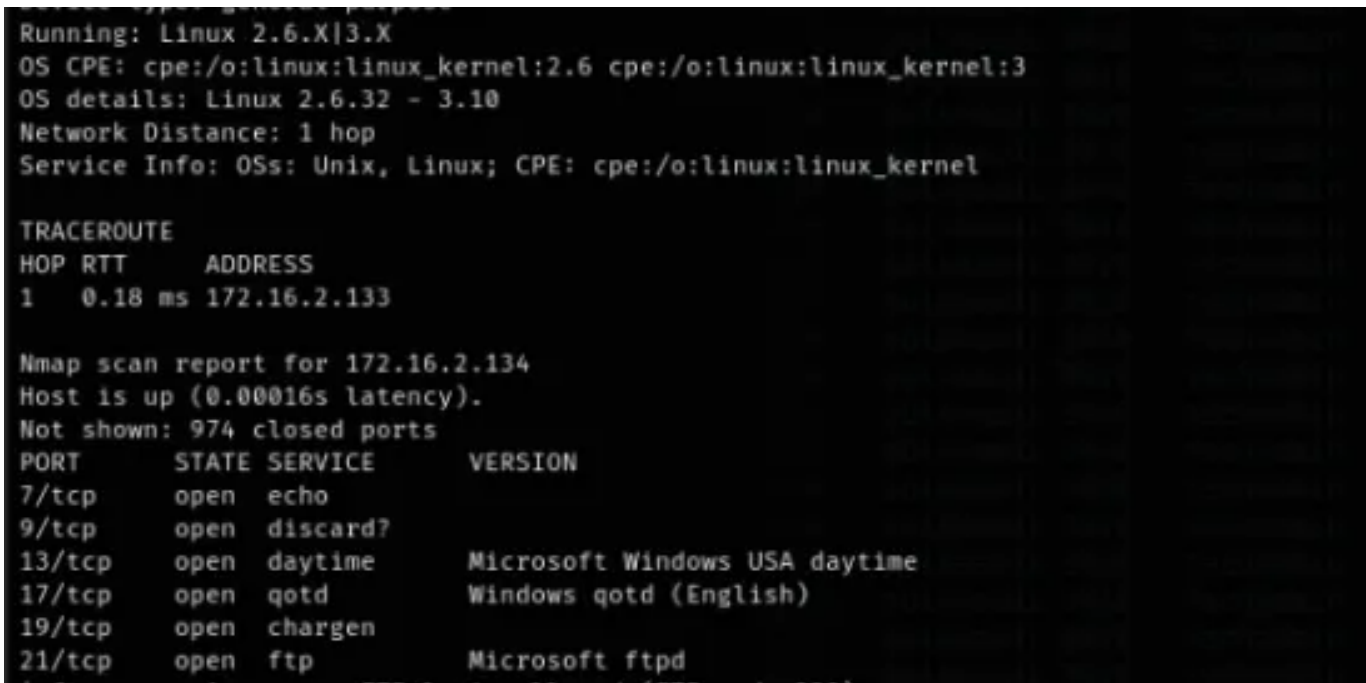



 Saeed Ahmed

NetworkMiner

NetworkMiner is a Network Forensics Analysis Tool (NFAT) designed primarily for parsing pre-captured traffic files, such as PCAPs, to...

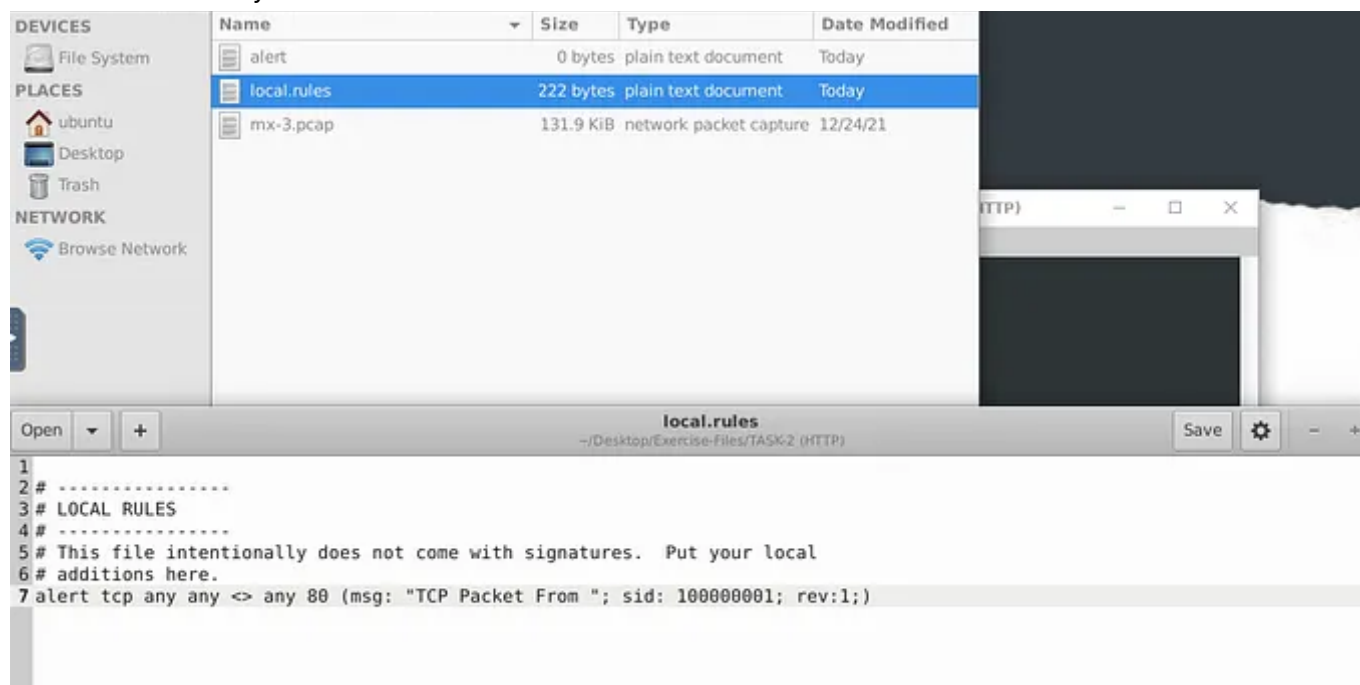
Sep 13



 Saeed Ahmed

Cyber Drill: Incident Investigation of a Compromised System

1. Executive Summary

 Saeed Ahmed

Snort Challenge — The Basics

Introduction

Aug 26

[See all from Saeed Ahmed](#)

Recommended from Medium

The screenshot displays the Resume Worded interface. On the left, a sidebar shows a score of 96 (Overall) and a list of fixes: Dates, Spelling & consistency, Growth signals, Job fit, and Filler words. The main area shows the 'LATEST SCORE' of 96 out of 100, indicating the resume is in the top 5% of all resumes analyzed. A progress bar shows the score has increased by ~80 points since the last upload. Below this, a section titled 'Steps to increase your score' is visible. On the right, a resume preview for 'Swetha Boddu' is shown, detailing Education, Technical Skills, Experience, and Projects.

Resume Worded | SCORE MY RESUME

Welcome to your resume review.

96 OVERALL

↑ 79 POINTS

Home

IP FIXES

Dates

Spelling & consi...

Growth signals

Job fit

Filler words

1 MORE ISSUES

COMPLETED

Buzzwords 10

Unlock full report

LATEST SCORE

Your resume scored 96 out of 100.

Your resume is in the top 5% of all resumes we've analyzed - well done! Now, before applying for a job, remember to tailor your resume to the job you're applying to. Targeted Resume helps you do that in two minutes.

YOUR RESUME

0 TOP RESUMES

Your resume score has increased by ~80 points since your last upload. Excellent work on improving your resume.

Steps to increase your score

Swetha Boddu

EDUCATION

Bachelor of Technology in Data Science 2020 - 2024

A.B.T. # CYP4

TECHNICAL SKILLS

Programming Languages: Python, SQL, R, HTML, CSS.

Big Data & Visualization: Spark, Hadoop, Pig, Power BI, Tableau, Python (pandas, numpy, matplotlib, seaborn, sklearn).

Data Science & Miscellaneous Technologies: A/B testing, ETL, Statistics, Git, GitHub, Ms Office, MS Excel, Ms Word, Githab, Git, Hypothesis Testing, Census, UML, Jira, Emap.

EXPERIENCE

Oct 2024 - Present

- Maintained high emotional resilience while reviewing offensive and sensitive material across flexible shifts, ensuring 100% compliance with internal quality and policy standards.
- Reduced policy violations by 90% investigating flagged content (hate speech, illegal material), identifying patterns and misuse to refine content filtering systems.
- Increased team efficiency by 30% reporting content trends and analysis insights to stakeholders, which led to streamlined moderation workflows and improved decision accuracy.

May 2023 - Jul 2023

- Powered and maintained analytics frameworks, improving campaign targeting accuracy by 30% by transforming complex datasets into actionable insights for marketing and sales teams.
- Streamlined customer acquisition by 15% by analyzing complex datasets to identify key growth opportunities.
- Delivered bi-weekly progress reports, resulting in a 20% improvement in team collaboration and project delivery efficiency by collecting detailed project requirements.

Feb 2023 - Apr 2023

- Enhanced real-time insights and reduced manual data analysis time by 40% by designing interactive Tableau dashboards, increasing stakeholder engagement.
- Optimized data validation by 85% by processing extensive datasets using SQL and Excel, ensuring higher data quality and reliability for analysis.
- Standardized revenue growth by 20% within the first quarter by implementing a real-time performance dashboard providing detailed insights into customer acquisition and product trends.

PROJECTS

Covid-19 Detection using Chest X-ray | Python, Deep Learning Jan 2024 - May 2024

- Achieved a 90% accuracy rate in distinguishing COVID-19 cases from other respiratory illnesses by developing and implementing a CNN-based algorithm.

Swetha 🌟


How I Went from an ATS Resume Score of 43 to 96

This Is How It Started...

May 21

51




 In T3CH by Axoloth

TryHackMe | KQL (Kusto): Basic Queries | WriteUp

51

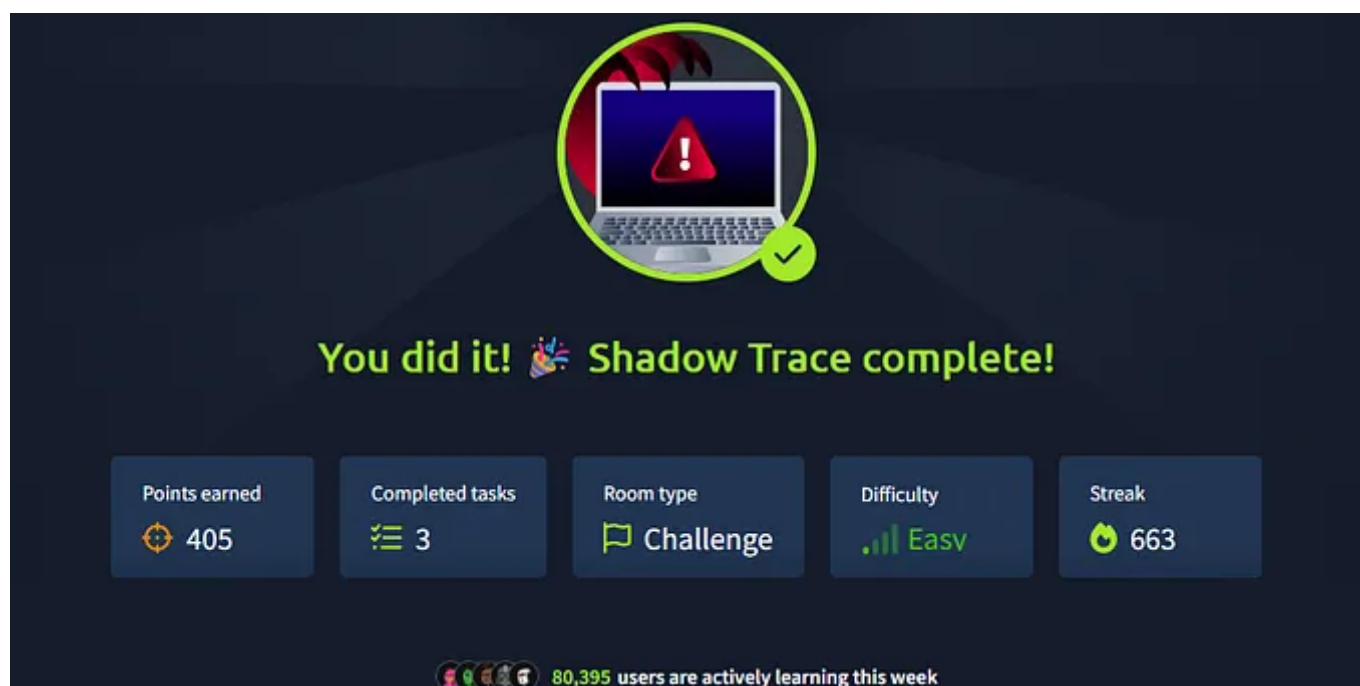


 In Towards AWS by bektiaw

AWS Project to get Hired: Smart Static Websites

Step by Step Guide to Host a Secure, Fast, Reliable Website on AWS to Impress Recruiters

Oct 6 1



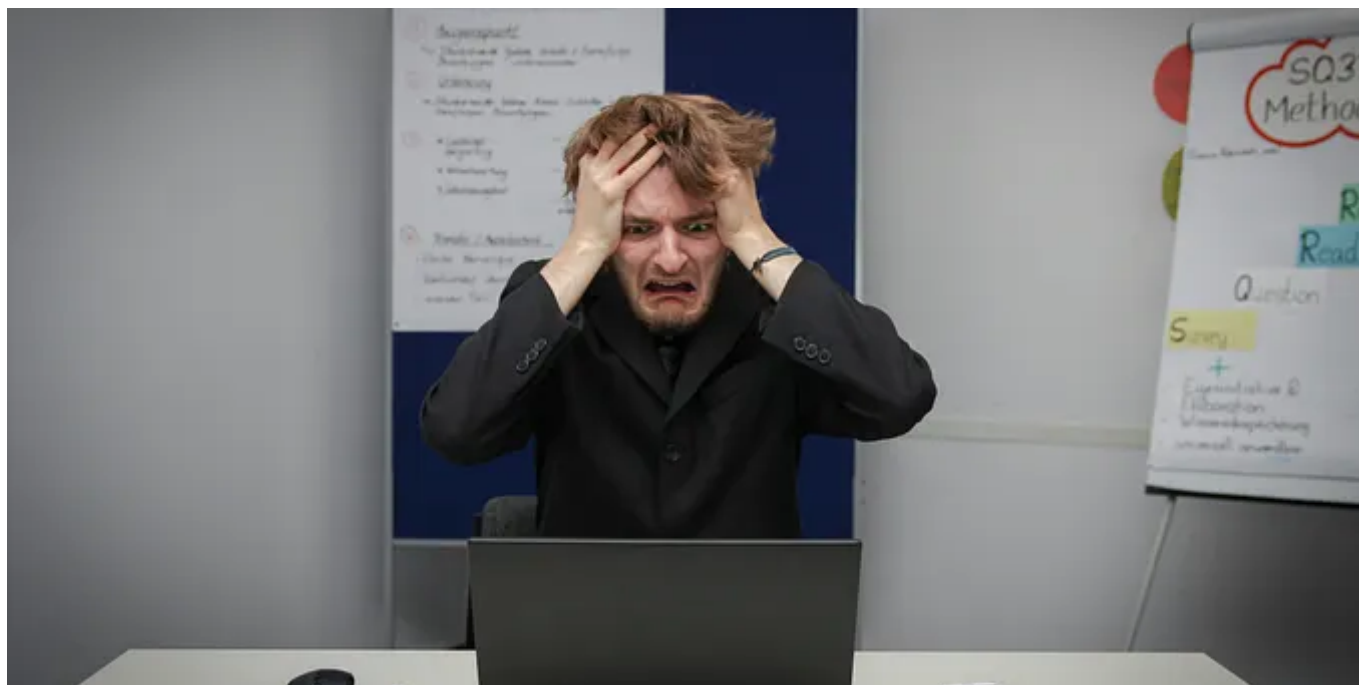
 Sle3pyHead 🤖


Shadow Trace CTF Notes | TryHackMe

CTF guide using PESTudio and PowerShell to find IOCs.

4d ago

11



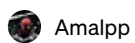
 Krishna Gupta

Why Do Employees Blame Their Bosses? A Hard Truth Every Professional Should Hear

“It’s not the boss versus the employee, it’s the team versus the problem.”

2d ago

19



Amalpp

Snort

Snort is a popular open-source network intrusion detection and prevention system (IDS/IPS) developed by Cisco.

Jun 9 3 1

See more recommendations