Title: Networking and Security Operations with SIEM, Forensics, and Traffic Analysis

1. Introduction

This task focuses on creating a subnet for small network and analyze the network traffic using packet capture tools . Troubleshoot network protocol issue in simulated environment, Setting up SIEM using ELK stack for log monitoring and analysis, then to simulate a incident and perform network forensics to investigate. Performing threat hunting using network log data

2. Tools Used

- Operating system: Kali Linux
- ELK stack
- Wireshark
- Cisco Packet Tracker
- Docker

3. Subnet Design

IP range

We can use this subnet 192.168.10.0/27 with subnet mask 255.255.255.224, giving 32 total addresses and 30 usable hosts, The remaining one address is for Broadcast.

- Total addresses: 32 (192.168.10.0 – 192.168.10.31)
- Network address: 192.168.10.0
- Usable host range: 192.168.10.1 – 192.168.10.30
- Broadcast address: 192.168.10.31
- Default gateway: 192.168.10.1
- Devices (PCs, printers, etc.): 192.168.10.2 – 192.168.10.30

This gives 29 device IPs + 1 gateway (30 usable in total)

Subnetmask

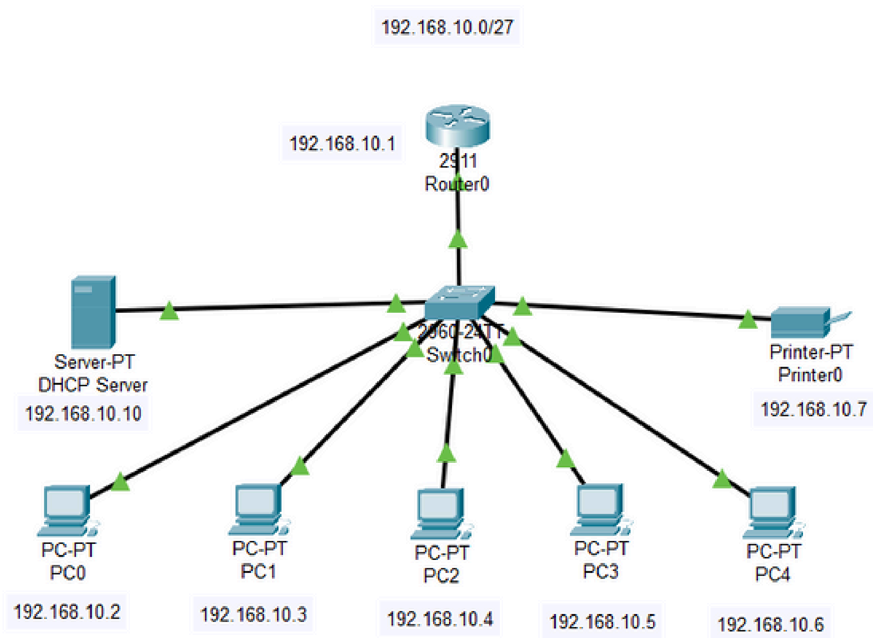
- Network ID: 192.168.10.0
- CIDR: /27
- Subnet mask: 255.255.255.224

Calculations

Number of devices: 20
Need: at least 20 usable IPs, plus 1 for the default gateway and some extra room
We can use this formula to calculate the subnet: 2^(32−prefix)−2
For /27:

=2^(32−27)−2
=2^5−2
=32−2
=30 usable hosts

Subnet diagram



Small office network with 192.168.10.0/27 subnet

Simple network with the subnet of **192.168.10.0/27** , here i have created a small office network and tested the connection between then and verified a successful connection among the network

Setup Verification

```
C:\>ping 192.168.10.1

Pinging 192.168.10.1 with 32 bytes of data:

Reply from 192.168.10.1: bytes=32 time<1ms TTL=255
Reply from 192.168.10.1: bytes=32 time<1ms TTL=255
Reply from 192.168.10.1: bytes=32 time<1ms TTL=255

Ping statistics for 192.168.10.1:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

Control-C
^C
C:\>ping 192.168.10.10

Pinging 192.168.10.10 with 32 bytes of data:

Reply from 192.168.10.10: bytes=32 time<1ms TTL=128
Reply from 192.168.10.10: bytes=32 time<1ms TTL=128
Reply from 192.168.10.10: bytes=32 time<1ms TTL=128
```

Ping Verification

Setup has successfully verified by pinging the Gateway and the DHCP server

4. Traffic Analysis

Captured Traffic

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	8592	100.0	2386985	156 k	0	0	0	8592
Ethernet	100.0	8592	5.0	120288	7897	0	0	0	8592
Internet Protocol Version 4	99.9	8582	7.2	171640	11 k	0	0	0	8582
User Datagram Protocol	86.9	7469	2.5	59752	3922	0	0	0	7469
Real-time Transport Control Protocol	4.1	355	0.6	14496	951	341	12908	847	372
Malformed Packet	0.2	14	0.0	0	0	14	0	0	14
QUIC IETF	14.7	1260	38.4	915744	60 k	1260	904837	59 k	1290
Multicast Domain Name System	0.1	6	0.0	1016	66	6	1016	66	6
Domain Name System	1.0	90	0.3	6608	433	90	6608	433	90
Data	67.0	5758	34.5	823782	54 k	5758	823782	54 k	5758
Transmission Control Protocol	12.9	1107	1.0	22800	1496	573	12120	795	1107
Transport Layer Security	6.1	522	9.6	228761	15 k	522	135395	8889	532
Hypertext Transfer Protocol	0.0	4	0.0	846	55	4	846	55	4
Data	0.1	8	0.0	8	0	8	8	0	8
Internet Control Message Protocol	0.1	6	0.0	240	15	6	240	15	6
Address Resolution Protocol	0.1	10	0.0	280	18	10	280	18	10
802.1Q Virtual LAN	11.5	992	0.2	3968	260	0	0	0	992

Protocol Hierarchy

tcp

No.	Time	Source	Destination	Protocol	Length	Info
5	0.012970	10.179.107.68	172.66.161.212	TLSv1.2	85	Application Data
9	0.028519	172.66.161.212	10.179.107.68	TCP	58	443 → 61372 [ACK] Seq=1 Ack=32 Win=17 Len=0
11	0.089892	104.18.34.135	10.179.107.68	TLSv1.2	118	Application Data
15	0.143301	10.179.107.68	104.18.34.135	TCP	54	58822 → 443 [ACK] Seq=1 Ack=61 Win=254 Len=0
24	0.269801	172.66.161.212	10.179.107.68	TLSv1.2	83	Application Data
26	0.314165	10.179.107.68	172.66.161.212	TCP	54	61372 → 443 [ACK] Seq=32 Ack=26 Win=255 Len=0
65	1.022785	10.179.107.68	18.97.36.44	TLSv1.2	294	Application Data
82	1.308495	18.97.36.44	10.179.107.68	TCP	54	443 → 65310 [ACK] Seq=1 Ack=241 Win=452 Len=0
83	1.308600	18.97.36.44	10.179.107.68	TLSv1.2	172	Application Data
84	1.349203	10.179.107.68	18.97.36.44	TCP	54	65310 → 443 [ACK] Seq=241 Ack=119 Win=252 Len=0
99	1.587931	10.179.107.68	104.16.103.112	TLSv1.2	2591	Application Data

TCP protocol

udp

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	104.29.139.88	10.179.107.68	RTCP	94	Receiver Report
2	0.000000	104.29.139.88	10.179.107.68	RTCP	218	Sender Report
3	0.000000	104.29.139.88	10.179.107.68	UDP	132	19300 → 62352 Len=90
4	0.003628	104.29.139.88	10.179.107.68	UDP	132	19300 → 62352 Len=90
6	0.022959	104.29.139.88	10.179.107.68	UDP	153	19300 → 62352 Len=111
7	0.026760	104.29.139.88	10.179.107.68	UDP	249	19300 → 62352 Len=207
8	0.026760	104.29.139.88	10.179.107.68	UDP	249	19300 → 62352 Len=207

UDP Protocol

dns

No.	Time	Source	Destination	Protocol	Length	Info
97	1.587183	10.179.107.68	10.179.107.37	DNS	73	Standard query 0x2ff9 HTTPS www.canva.com
98	1.587505	10.179.107.68	10.179.107.37	DNS	73	Standard query 0x7bc3 A www.canva.com
101	1.593252	10.179.107.37	10.179.107.68	DNS	151	Standard query response 0x7bc3 A www.canva.com CNAME www.canva.com.
108	1.631422	10.179.107.37	10.179.107.68	DNS	161	Standard query response 0x2ff9 HTTPS www.canva.com CNAME www.canva.
321	9.393888	10.179.107.68	10.179.107.37	DNS	77	Standard query 0x1b46 A cdn.growthbook.io
323	9.439202	10.179.107.68	10.179.107.37	DNS	77	Standard query 0x1b46 A cdn.growthbook.io

DNS Protocol

ip.addr == 10.179.107.68

No.	Time	Source	Destination	Protocol	Length	Info
84	1.349203	10.179.107.68	18.97.36.44	TCP	54	65310 → 443 [ACK] Seq=241 Ack=119 Win=252 Len=0
85	1.356033	104.29.139.88	10.179.107.68	UDP	216	19300 → 62352 Len=174
86	1.362462	104.29.139.88	10.179.107.68	UDP	196	19300 → 62352 Len=154
87	1.374351	104.29.139.88	10.179.107.68	UDP	199	19300 → 62352 Len=157
88	1.379543	104.29.139.88	10.179.107.68	UDP	192	19300 → 62352 Len=150
89	1.430491	104.29.139.88	10.179.107.68	UDP	163	19300 → 62352 Len=121
90	1.445072	104.29.139.88	10.179.107.68	UDP	174	19300 → 62352 Len=132
91	1.476121	104.29.139.88	10.179.107.68	UDP	168	19300 → 62352 Len=126
92	1.479890	104.29.139.88	10.179.107.68	UDP	172	19300 → 62352 Len=130
93	1.494819	104.29.139.88	10.179.107.68	UDP	167	19300 → 62352 Len=125
94	1.534361	104.29.139.88	10.179.107.68	UDP	153	19300 → 62352 Len=111

IP based Filter

5. Protocol Troubleshooting

UDP Traffic

The UDP view shows a remote host 104.29.139.88 sending a stream of UDP packets to 10.179.107.68 on high ports (e.g., 19300 → 62352), and some RTCP packets.

DNS Traffic

The DNS filter shows 10.179.107.68 querying 10.179.107.37 for domains such as www.canva.com and cdn.growthbook.io, with corresponding responses from the same DNS server.

TCP Traffic

In the TCP/TLS view, 10.179.107.68 establishes TLSv1.2 sessions with multiple remote servers (e.g., 172.66.161.212, 18.97.36.44, 104.18.34.135) using destination ports 443 and 65310

---

6. SIEM Setup

Installation & Configurations

1. I used Docker for easier setup and configuring the ELK stack
2. Install docker for Windows
3. Create a separate folder "ELK-lab"
4. Create a docker-compose.yaml file with the below content:

```
version: "3.8"

services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:8.12.0
    container_name: es-lab
    environment:
      - discovery.type=single-node
      - ES_JAVA_OPTS=-Xms1g -Xmx1g
      - xpack.security.enabled=false
    ports:
      - "9200:9200"
    volumes:
      - esdata:/usr/share/elasticsearch/data

  kibana:
    image: docker.elastic.co/kibana/kibana:8.12.0
    container_name: kibana-lab
    environment:
      - ELASTICSEARCH_HOSTS=http://elasticsearch:9200
```

```
    ports:
      - "5601:5601"
    depends_on:
      - elasticsearch

  logstash:
    image: docker.elastic.co/logstash/logstash:8.12.0
    container_name: logstash-lab
    ports:
      - "5140:5140/udp"
    volumes:
      - ./logstash-pipeline:/usr/share/logstash/pipeline
    depends_on:
      - elasticsearch

volumes:
  esdata:
```

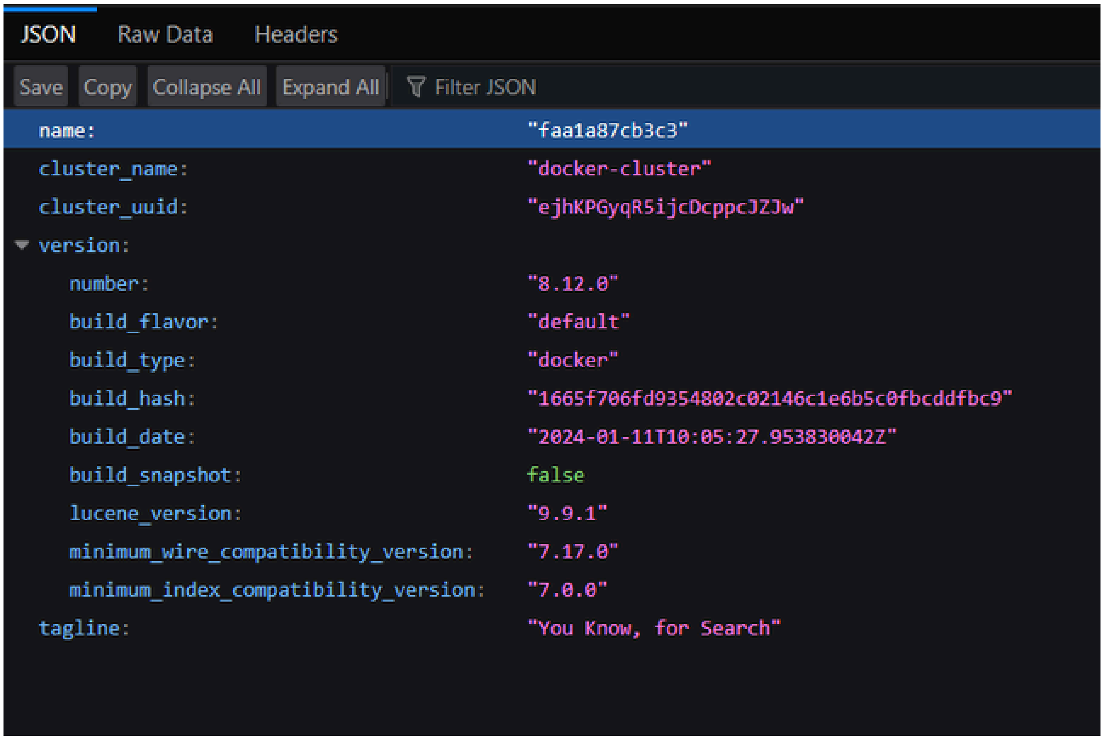
5. Create logstash-pipeline/logstash.conf in that folder and add this configurations in that file:

```
input {
  udp {
    port => 5140
    type => "syslog"
  }
}

filter { }

output {
  elasticsearch {
    hosts => ["http://elasticsearch:9200"]
    index => "syslog-%{+YYYY.MM.dd}"
  }
}
```

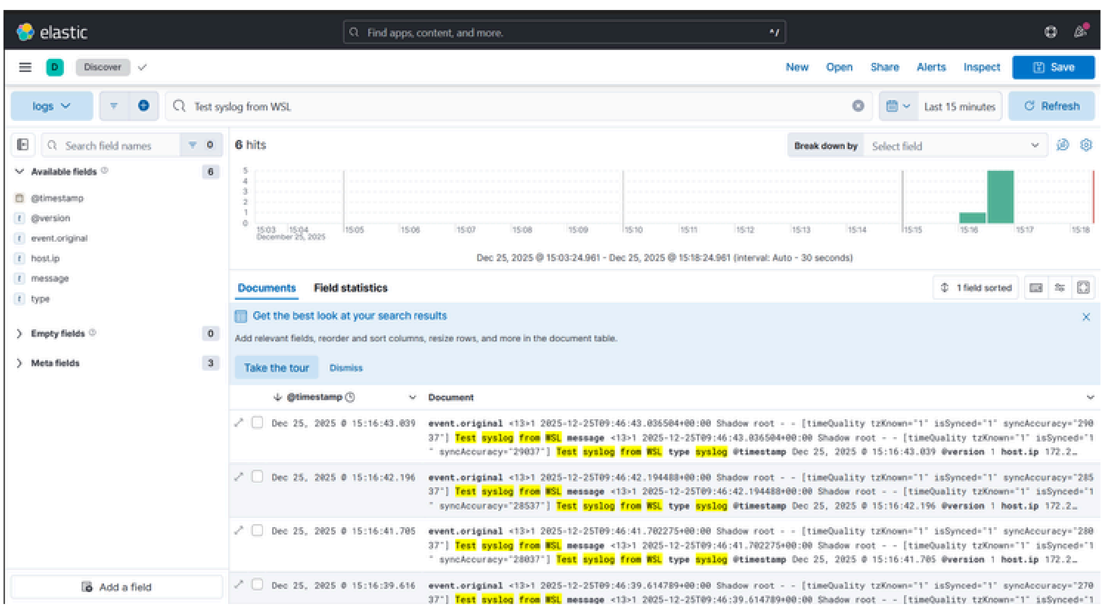
6. Then start the ELK stack using this command :
   docker compose up -d

7. Verify the Elastic setup by visiting
   http://localhost:9200 .which provide a Elasticsearch json data

8. Verify the Kibana setup by visiting
   http://localhost:5601 .which provide a Elasticsearch json data



Elasticsearch

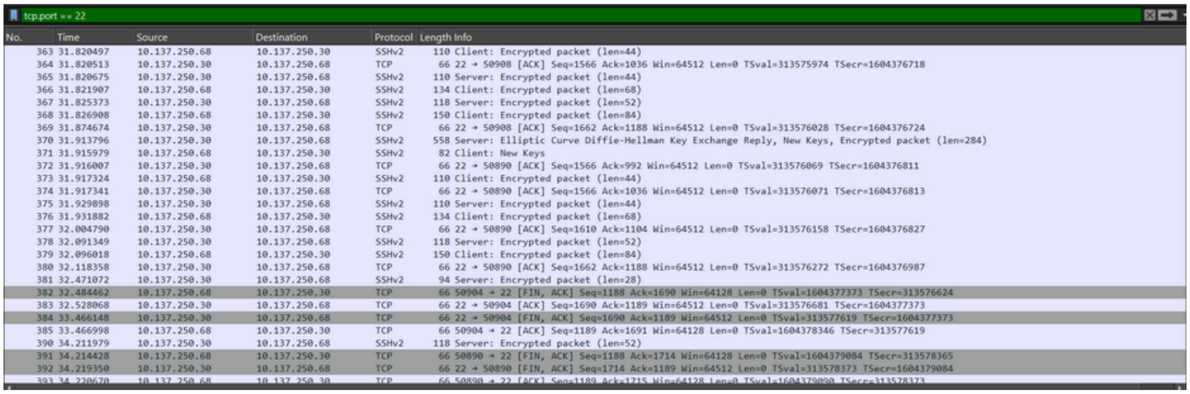


Kibana

7. Incident Forensics

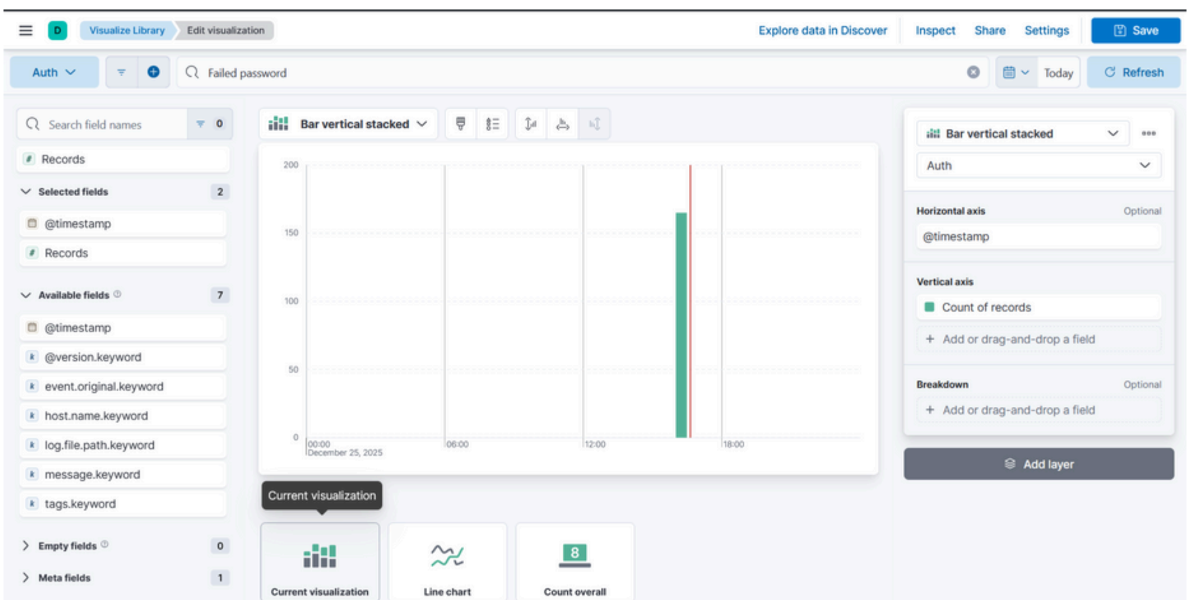
For Testing i performed a SSH brute-force attack on my target machine which is on the lab environment and captured the traffic in my target machine using **Wireshark**

- Performed a Brute-force attack on the Target
- Captured the Traffic using Wireshark
- Analyzing the traffic by filtering the port 22
  - **tcp.port == 22**



Wireshark

I configured and ingested the auth.log into kibana to provide a dashboard view using the rsyslog, where it send the captured log file to my host machine where i had a ELK setup, Then configure the docker-compose.yaml and other config files to use my auth.logs to create a data view in Kibana with timeline



Kibana

8. Threat Hunting

Tools used

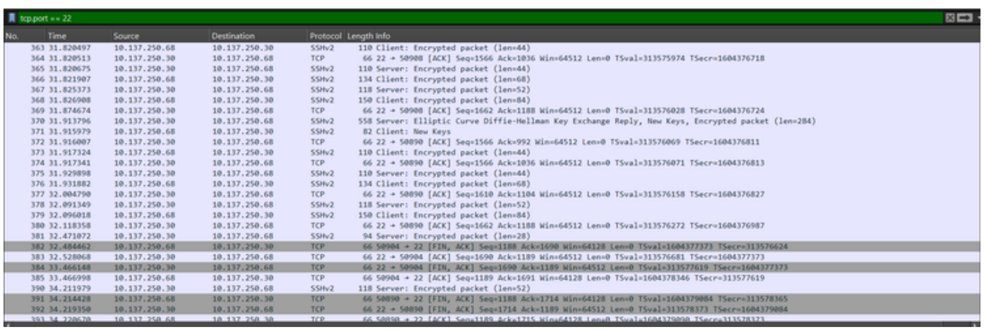
- Wireshark
- Kibana (ELK Stack)

Methods

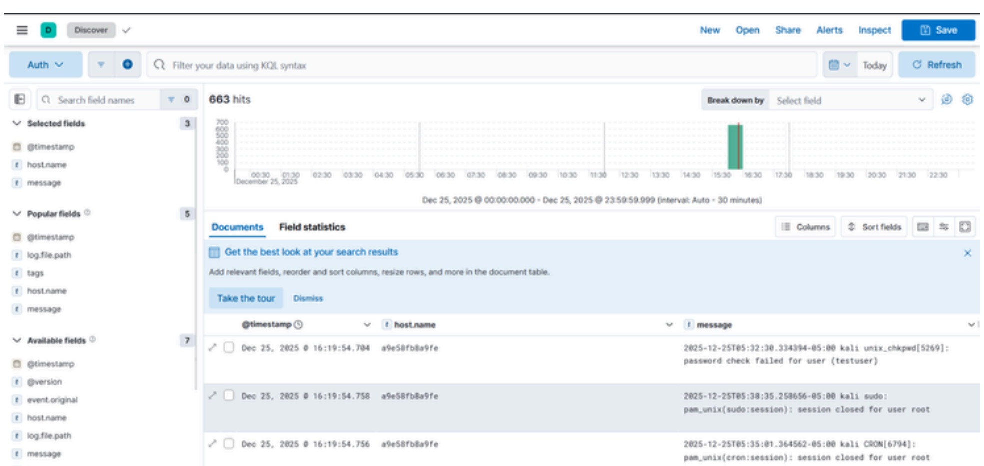
- Using Wireshark on the host to inspect captured traffic (incident_ssh_bruteforce.pcap) and apply protocol-level filters such as tcp.port == 22, dns, and http for deep packet inspection.
- Using Kibana (ELK Stack) to query imported SSH authentication logs for failed logins, aggregate events by source IP and username, and visualize spikes over time using KQL and Lens visualizations.

Findings and IOCs

- Identified a single attacker IP (e.g., 192.168.10.50) generating dozens of Failed password events against user testuser on the SSH service of the lab host in a short period, matching the hydra brute-force run seen in Wireshark.
- Confirmed correlation between the time of the SSH connection burst in the PCAP and the spike of failed-login events in Kibana, treating the attacker IP (192.168.10.50), target IP (192.168.10.20 or WSL IP), username (testuser), and service (ssh/TCP 22) as indicators of compromise (IOCs) for this simulated incident.



Wireshark IOC



Kibana IOC

9. Key Learnings

- Learned how to design and validate a /27 subnet for a small office network, then reproduce that design in Packet Tracer and verify connectivity with tools like ping and simulation mode
- Gained practical experience capturing and interpreting real traffic (TCP, UDP, DNS, SSH) with Wireshark/tshark and correlating it with log data ingested into an ELK-based SIEM for incident reconstruction
- Understood how to simulate attacks such as SSH brute-force, ingest auth logs (via syslog/file import) into ELK, and perform basic threat hunting in Kibana using KQL queries to identify IOCs like attacker IPs, usernames, and time-based spikes.

10. Conclusion

Designed a small office network with the subnet of /27 and verified the connections between them, captured the network traffic using Wireshark and analyzed the captured packets using various filters, Successfully setup a SIEM using ELK stack in docker system and verified successfully. Simulated a Brute-force attack on the lab machine and captured the Traffic using wireshark , then used various filters and done a threat hunting in the captured data, Then ingested the ssh logs into the ELK stack and create a data view and visualize the data in the stack and performed threat hunting those data as well and provide a IOC for the findings

11. References

- IP Subnetting - https://ipcisco.com/lesson/ip-subnetting-and-subnetting-examples
- Wireshark - https://www.geeksforgeeks.org/ethical-hacking/protocol-hierarchy-window-in-wireshark/
- ELK Stack - https://www.ibm.com/docs/en/snips/4.6.0?topic=options-configuring-snort-configuration-rules
- Log Ingestion - https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elastic-stack-on-ubuntu-22-04