PROJECT NAME

Analyzing Network Traffic with Moloch and Elastic

Domain Name

Cybersecurity, Network Traffic Analysis, Data Visualization

CDAC, Noida

CYBER GYAN VIRTUAL INTERNSHIP PROGRAM

Submitted By:

Yatharth Kumar Saxena
Project Trainee, (July-August) 2025
Batch No: 18

BONAFIDE CERTIFICATE

This is to certify that this project report entitled

Analyzing Network Traffic with Moloch and Elastic

submitted to CDAC Noida, is a Bonafide record of work done by

Yatharth Kumar Saxena, under my supervision from

11th July 2025 to **25th July 2025**.

Declaration by Author(s):

This is to declare that this report has been written by me.

No part of the report is plagiarized from other sources.

All information taken from external sources has been duly acknowledged.

I aver that if any part of the report is found to be plagiarized,

I shall take full responsibility for it.

Name of Author: Yatharth Kumar Saxena

TABLE OF CONTENTS:

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to all those who supported and guided me throughout the course of this project titled “Analyzing Network Traffic using Moloch and Elastic Stack.”

I am especially thankful to our faculty and lab instructors for providing the essential knowledge, infrastructure, and a motivating environment to explore real-world cybersecurity tools. Their continuous encouragement made this technical exploration a valuable learning experience.

I would also like to thank the open-source communities behind **Moloch (Arkime)**, **Wireshark**, and the **Elastic Stack**, whose powerful tools and extensive documentation played a key role in the successful execution of this project.

Lastly, I extend my appreciation to my peers for their collaborative spirit and insightful discussions, which enriched the practical understanding of network behavior and analysis.

1. Introduction

1.1 PROBLEM STATEMENT:

Analyzing Network Traffic with Moloch and Elastic

1.2 Learning Objectives:

a. Understand full-packet capture systems like **Moloch**.
b. Learn how to integrate **Moloch with Elasticsearch and Kibana**.
c. Perform hands-on network behavior analysis and anomaly detection.
d. Learn to visualize network data and interpret signs of threats or performance issues.

1.3 Problem Addressed:

During the implementation of this project, I encountered several practical challenges that required deep system-level troubleshooting and research:

a. Environment Dependency:
The tools used in this project, especially Moloch (now Arkime), are not supported on Windows. As a result, I had to set up a **dual-boot environment with Ubuntu** on my personal laptop, which was itself a technically demanding task involving partitioning and bootloader configurations.

b. Installation Challenges:
The installation of Arkime required multiple dependencies, and due to broken or missing package links, many commands failed in the terminal. I had to **rely heavily on community forums and official documentation** to navigate through the issues and configure each component manually. No pre-configured Docker setup was used — all linkages were done natively through the terminal and shell scripting.

c. Data Visualization Issues in Kibana:
While trying to analyze security threats, I initially faced an error in Kibana's **Discover** tab stating:

> 3 of 4 shards failed
> Despite having valid pcap input, the dashboard was not reflecting meaningful data. This error required deeper understanding of **ElasticSearch indexing and shard management**.

d. Security Threat Detection Difficulties:
Extracting security insights from the captured traffic was another significant hurdle. Initially, neither the Discover tab nor visual dashboards showed relevant data. Only after carefully creating **custom dashboards**, and using **filters and field analysis**, I was able to extract some meaningful Indicators of Compromise (IoCs). Additionally, I used **Wireshark with deep packet inspection on interface wlan0** to cross-verify traffic patterns and security events manually.

2. Approach

This project was executed entirely on a standalone Linux system (Ubuntu) without using Docker or virtual machines. All tools were configured and linked manually via terminal using the default wlan0 network interface for capturing and communication. The analysis stack included Moloch (Arkime) and the Elastic Stack (Elasticsearch + Kibana), with Wireshark used for deep packet inspection.

2.1. Tools and Technologies Used:

Moloch (Arkime) – For full packet capture and metadata indexing.
Elasticsearch – Backend engine storing Moloch and Logstash data.

Kibana – Visualization of packet insights and analytics.
Wireshark – Validation of captured packets from .pcap files.
PCAP Files – As data source for simulation.
Ubuntu (with wlan0**)** – Host machine for the entire setup.

2.2. System Setup:

Everything was installed and configured directly on a single Ubuntu machine using terminal commands.
No virtualization (like VMware/VirtualBox) or containerization (Docker) was used.
Network interface wlan0 was used for both data ingestion and tool communication.

2.3. Access Points Used in Firefox:

Component	Purpose	Firefox Access URL
Moloch Web Interface	Packet metadata viewer	http://localhost:8005
Kibana Dashboard	Visualization of logs/packets	http://localhost:5601
Elasticsearch Health	Verifying backend connectivity	http://localhost:9200

2.4. Data Flow & Command-Driven Linking:

All components were started from the terminal — no GUI tool or container service was involved.
Network traffic was captured via wlan0, either from live stream or .pcap files.
Captured data was parsed and indexed by **Moloch**, stored into **Elasticsearch**, and visualized in **Kibana**.
Wireshark was used to cross-check specific .pcap files separately.

3. Implementation

3.1. Configuration Steps:

Step 1: Ubuntu Installation & Setup

Installed Ubuntu via **dual boot**. From my **475 GB C Drive**, I allocated:

- **108 GB for Ubuntu**, split as:
  - **16 GB** → *Swap area*
  - **92 GB** → */ (root)*
- Rest of the space reserved for Windows OS.

Post-installation:

- Opened **Terminal** and ran

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo apt update && sudo apt upgrade -y
[sudo] password for yatharth:
Get:1 http://in.archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
```

Step 2: Elasticsearch Installation

a. Added Elasticsearch GPG key & repo

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
[sudo] password for yatharth:
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
```

b. Fetched Repository that contain elastic with GPG Key

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-7.x.list
[sudo] password for yatharth:
deb https://artifacts.elastic.co/packages/7.x/apt stable main
```

c. Installed via sudo apt install elasticsearch and then update

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo apt install elasticsearch -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo apt update
Get:1 https://artifacts.elastic.co/packages/7.x/apt stable InRelease [13.7 kB]
```

d. Enabled & started with sudo systemctl enable –now elasticsearch and verified it is running perfectly

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo systemctl enable elasticsearch.service
Synchronizing state of elasticsearch.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable elasticsearch
Created symlink /etc/systemd/system/multi-user.target.wants/elasticsearch.service → /lib/systemd/system/elasticsearch.service.
```

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo systemctl start elasticsearch.service
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo systemctl status elasticsearch.service
● elasticsearch.service - Elasticsearch
     Loaded: loaded (/lib/systemd/system/elasticsearch.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2025-07-18 17:58:22 IST; 2min 5s ago
       Docs: https://www.elastic.co
```

e. Curl Installation

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo apt install curl -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ curl -X GET "localhost:9200"
{
  "name" : "yatharth-VivoBook-ASUSLaptop-X515EA-X515EA",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "Xd4soh9cRZK7LkhoYUEUng",
  "version" : {
    "number" : "7.17.29",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "580aff1a0064ce4c93293aaab6fcc55e22c10d1c",
    "build_date" : "2025-06-19T01:37:57.847711500Z",
    "build_snapshot" : false,
    "lucene_version" : "8.11.3",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

Step 3: Kibana Installation

a. Installed via sudo apt install kibana

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo apt install kibana -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

b. Enabled & started with sudo systemctl enable --now kibana and verified it running perfectly

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo systemctl enable kibana.service
Synchronizing state of kibana.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable kibana
Created symlink /etc/systemd/system/multi-user.target.wants/kibana.service → /etc/systemd/system/kibana.service.
```

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo systemctl start kibana.service
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo systemctl status kibana.service
● kibana.service - Kibana
     Loaded: loaded (/etc/systemd/system/kibana.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2025-07-18 18:18:49 IST; 34s ago
       Docs: https://www.elastic.co
```

Step 4: Arkime(Molosch) Installation

a. Installed Arkime deb from Github and checking it exists in downloads

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ cd ~/Downloads
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~/Downloads$ sudo dpkg -i arkime_5.7.0-1.ubuntu2204_amd64.deb
[sudo] password for yatharth:
(Reading database ... 277861 files and directories currently installed.)
Preparing to unpack arkime_5.7.0-1.ubuntu2204_amd64.deb ...
Unpacking arkime (5.7.0-1) over (5.7.0-1) ...
Setting up arkime (5.7.0-1) ...
Arkime systemd files copied
```

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ dpkg -l | grep arkime
ii  arkime                                5.7.0-1                                  amd64        Arkime Full Packet System
```

b. Now linking Molosch(Arkime) to Elastic:

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo /opt/arkime/db/db.pl http://localhost:9200 init
[sudo] password for yatharth:
It is STRONGLY recommended that you stop ALL Arkime captures and viewers before proceeding.  Use 'db.pl http://localhost:9200 backup' to backup db first.

There is 1 OpenSearch/Elasticsearch data node, if you expect more please fix first before proceeding.

This is a fresh Arkime install
Erasing
Creating
Finished
```

c. Initialized the Arkime configuration file (config.ini)

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo nano /opt/arkime/etc/config.ini
```

d. Start Arkime (Molosch) in Server and checking it is running perfectly

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo systemctl start arkimecapture
[sudo] password for yatharth:
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo systemctl start arkimeviewer
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo systemctl status arkimecapture
[sudo] password for yatharth:
● arkimecapture.service - Arkime Capture
     Loaded: loaded (/etc/systemd/system/arkimecapture.service; disabled; vendo>
     Active: active (running) since Sat 2025-07-19 19:04:54 IST; 43min ago
    Process: 5360 ExecStartPre=/opt/arkime/bin/arkime_config_interfaces.sh -c />
```

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo systemctl status arkimeviewer
● arkimeviewer.service - Arkime Viewer
     Loaded: loaded (/etc/systemd/system/arkimeviewer.service; disabled; vendor preset: enabled)
     Active: active (running) since Sat 2025-07-19 19:05:42 IST; 43min ago
   Main PID: 5414 (sh)
      Tasks: 12 (limit: 9067)
     Memory: 122.8M
        CPU: 1.321s
```

e. Enable Arkime(Molosch) Capture and Viewer

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo systemctl enable arkimecapture
Created symlink /etc/systemd/system/multi-user.target.wants/arkimecapture.service → /etc/systemd/system/arkimecapture.service.
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo systemctl enable arkimeviewer
Created symlink /etc/systemd/system/multi-user.target.wants/arkimeviewer.service → /etc/systemd/system/arkimeviewer.service.
```

Step 5: Wireshark Installation

a. Install Wireshark-Common

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo apt install wireshark-common
[sudo] password for yatharth:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
```

Step 6: Logstash Installation

a. Installed jdk version

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo apt install openjdk-11-jdk -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ java -version
openjdk version "11.0.27" 2025-04-15
OpenJDK Runtime Environment (build 11.0.27+6-post-Ubuntu-0ubuntu122.04)
Show Applications erver VM (build 11.0.27+6-post-Ubuntu-0ubuntu122.04, mixed mode, sharing)
```

b. Installed Logstash and verified it is installed

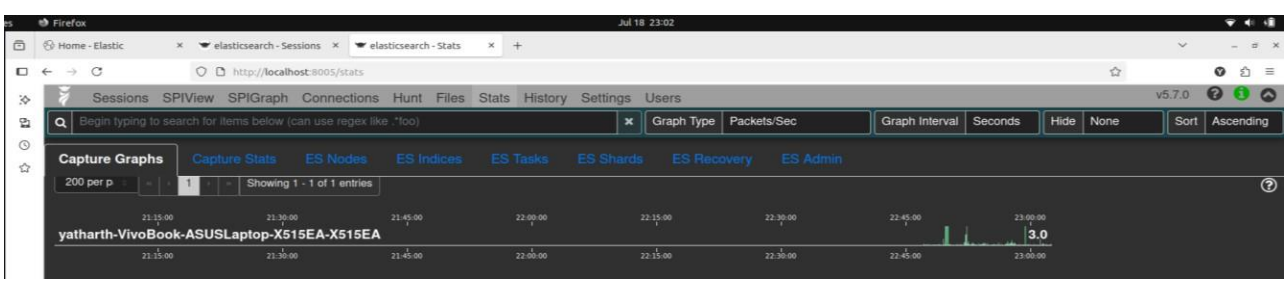
```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo apt install logstash -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

```
yatharth@yatharth-VivoBook-ASUSLaptop-X515EA-X515EA:~$ sudo /usr/share/logstash/bin/logstash --version
Using bundled JDK: /usr/share/logstash/jdk
logstash 8.18.3
```

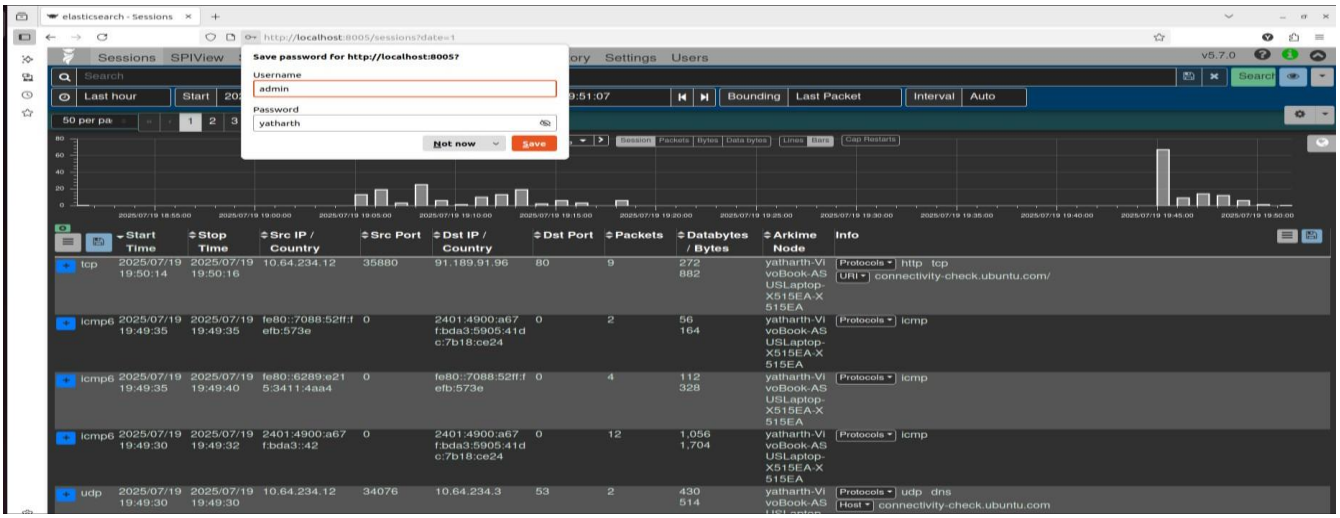
3.2. Network Analysis Screenshot:

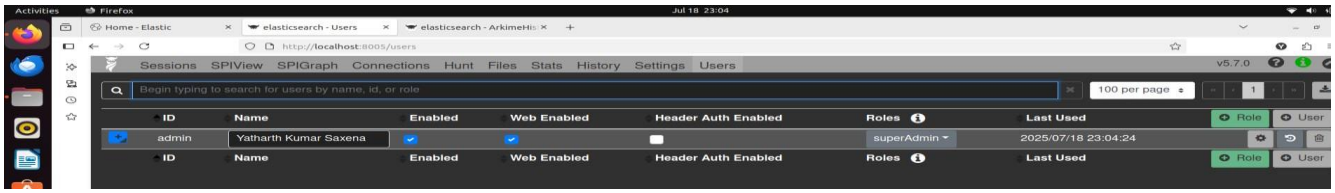
Step 1: Analysis through Molosch(Arkime)

a. Analysing Network Traffic on my live PC:

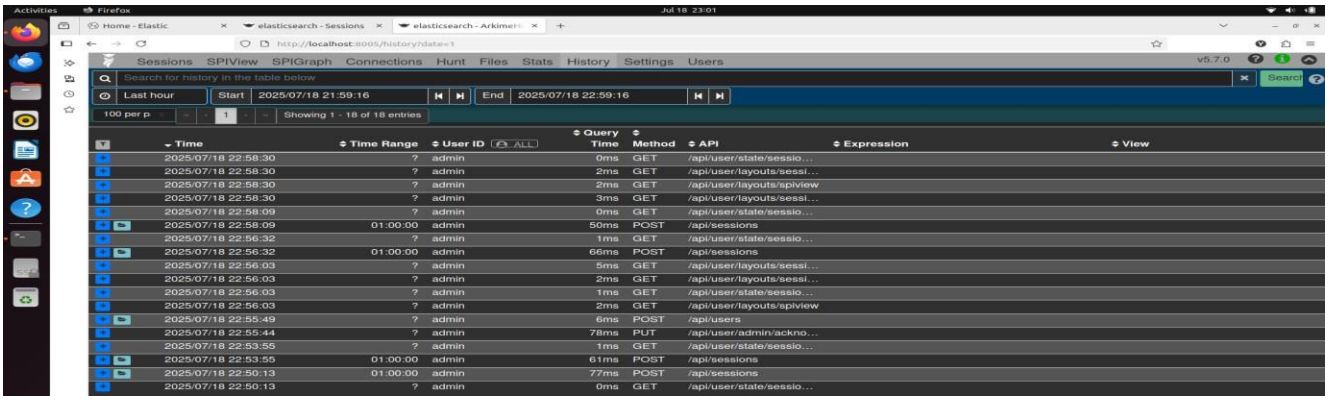


b. Arkime(Molosch) Capture and See Sign In Part:

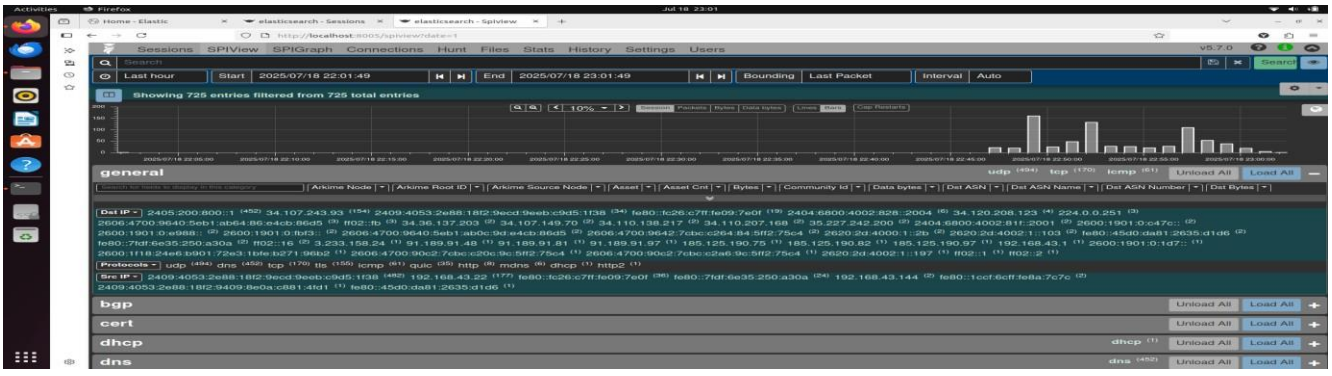




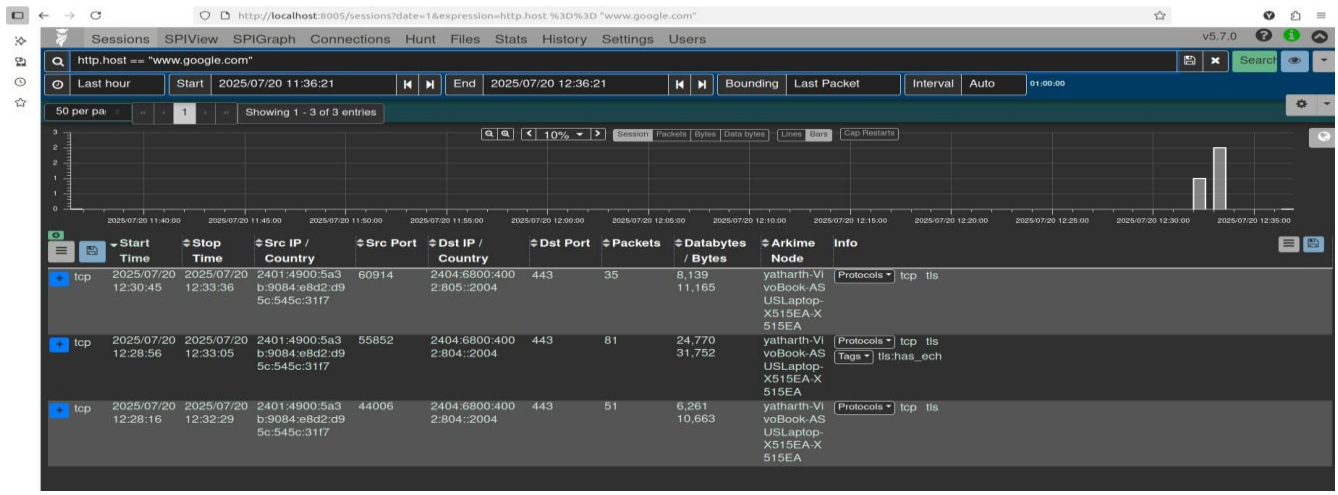
c. Analysis of Network Traffic using Arkime(Molosch):



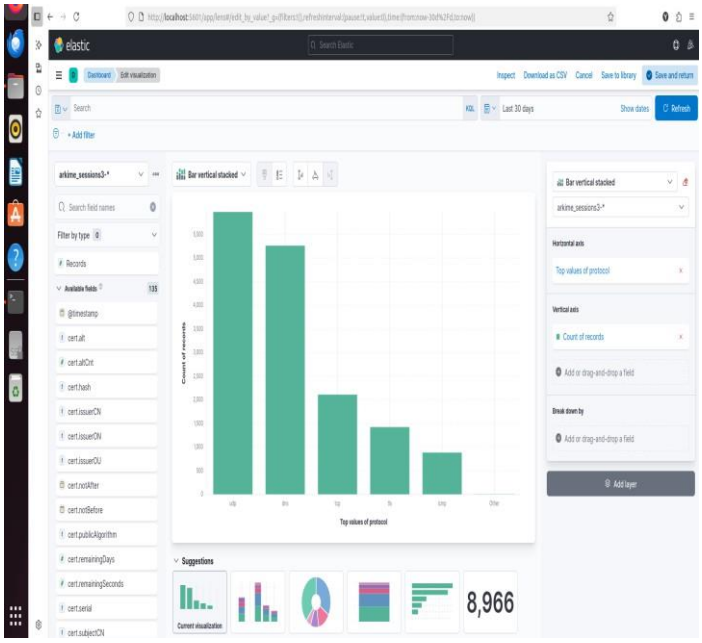
d. Deep Analysis of a Particular Session in Captured Network Traffic:

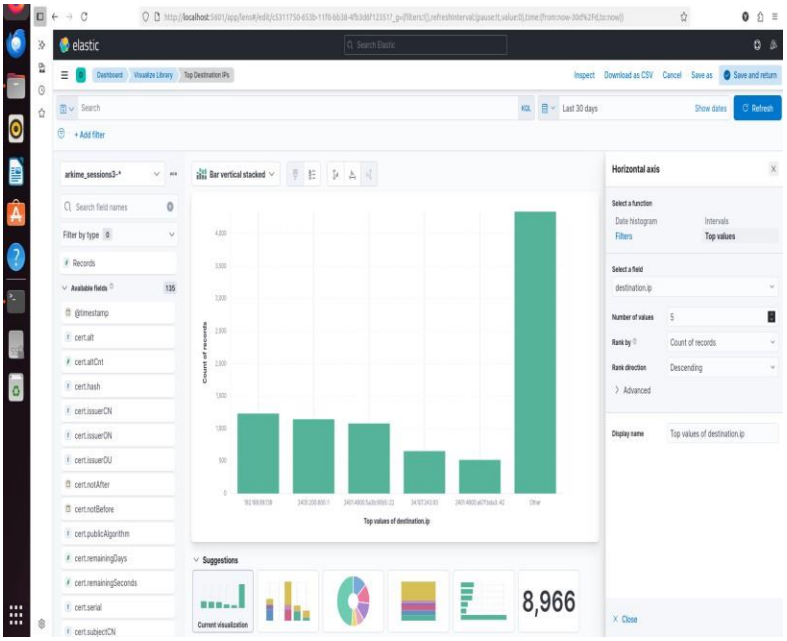


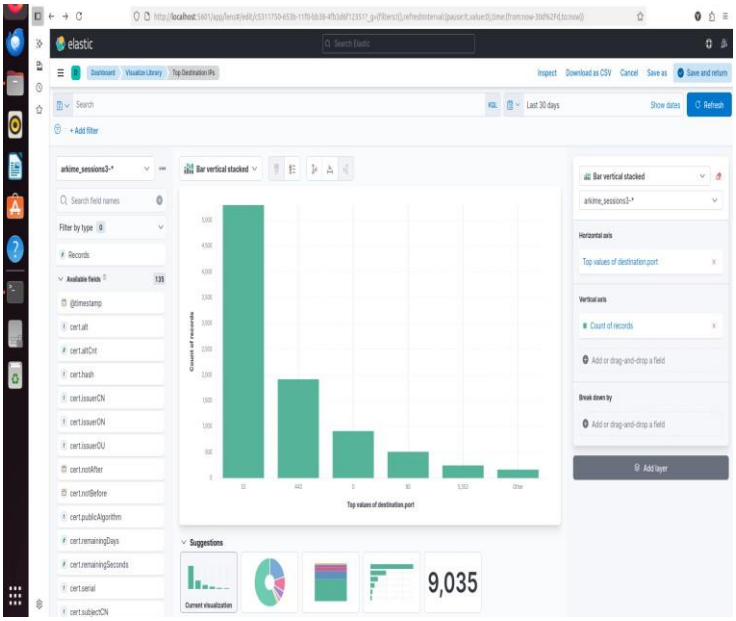
e. Deep Analysis by applying filters using search in Live Network Traffic:

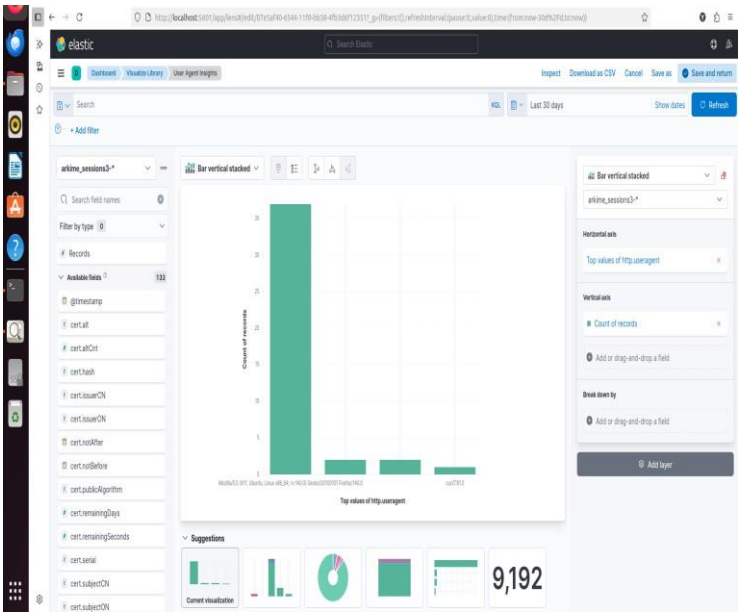


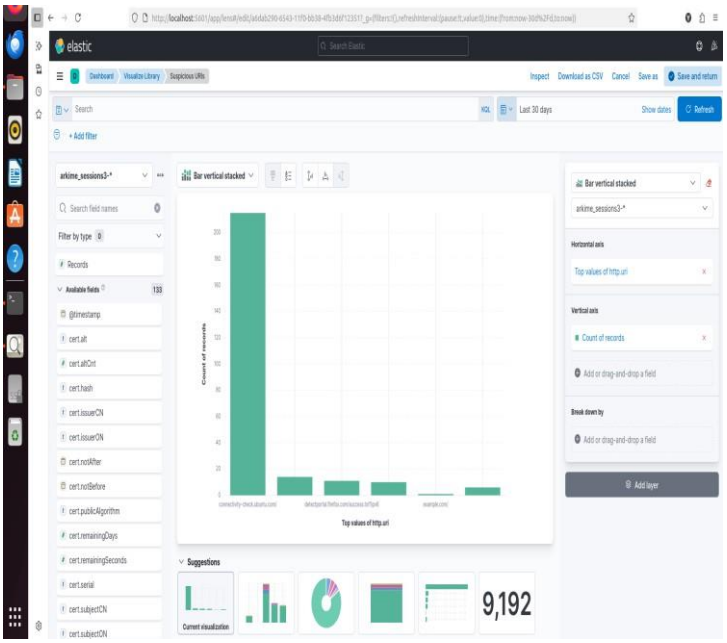
Step 2: Analysis through Kibana By Dashboard:

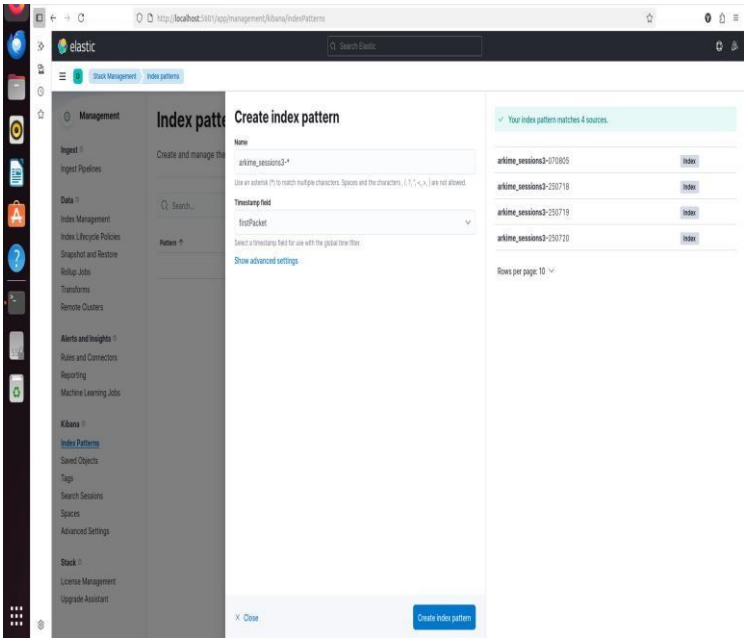








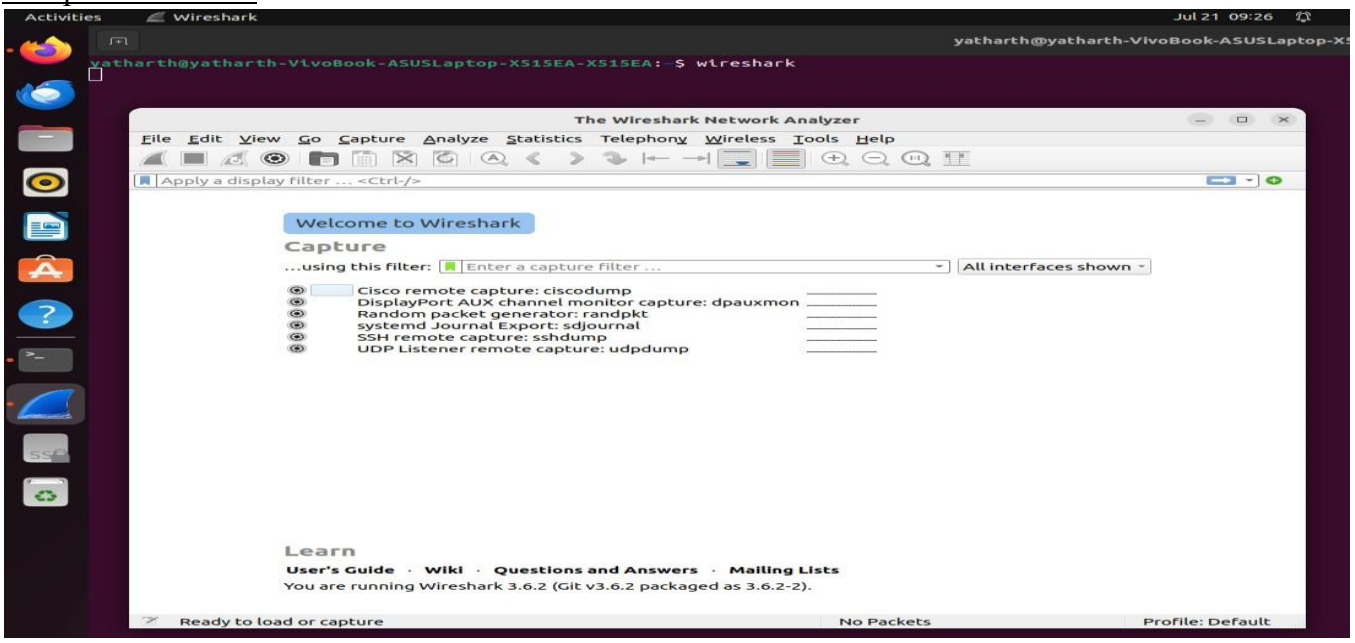




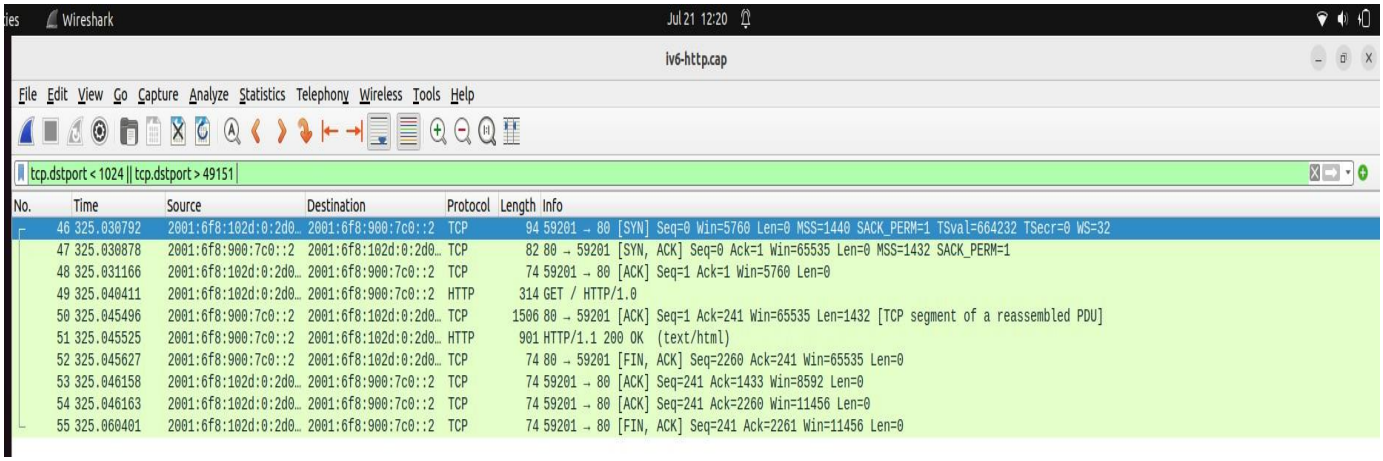
First we created Index Pattern and then we created Dashboards and extract some insights based on it

Step 3: Deep inspection of packet file (pcap) created by Arkime through Wireshark:

a. Open Wireshark:



b. Started Captured file Analysis created by Arkime during Live Capture



4. Conclusion & Recommendations:

4.1 Conclusion based on Analysis from Wireshark:

The captured IPv6 TCP traffic reveals several notable points:

1. **Three-Way Handshake Observation**:
   - The SYN, SYN-ACK, and ACK packets confirm a complete TCP handshake.
   - The connection is established on **port 80**, indicating **unencrypted HTTP communication**.
   - HTTP GET requests are visible, suggesting active web data retrieval.
2. **Application Layer Details**:
   - In Screenshot 2, the GET and 200 OK messages are clearly visible, including text/html data, which hints at potentially **exposed web content**.
3. **Lack of Encryption**:
   - As this traffic is on HTTP (port 80), it's **unencrypted** — a major red flag, especially if sensitive data is being exchanged.
4. **Potential Threat Vectors Identified**:
   - **Data Interception Risk**: The visibility of HTTP GET requests means credentials or cookies could be intercepted by attackers.
   - **Unsecured Protocol**: Usage of port 80 (instead of HTTPS) introduces **MITM (Man-in-the-Middle)** risks.
   - **IPv6 Targeting**: Traffic is using IPv6; many systems are less rigorously monitored in IPv6 environments, making this a possible **blind spot for attackers**.
   - The presence of multiple ACK/FIN packets suggests **clean termination**, but packet sequencing could be analyzed further to detect **session hijacking** attempts.

4.2 Conclusion based on Analysis from Kibana Dashboard:

a. Dashboard: Top values of User Agent

- Most traffic is from a single Firefox user-agent; CLI tool curl also detected — indicating potential scripted or automated access.
- Lack of diversity in user agents suggests possible internal or bot-based activity.

b. Dashboard: Top values of Protocol

- High DNS and UDP traffic volume could indicate tunneling, reconnaissance, or DDoS attempts.
- Low TLS usage implies data is often unencrypted and vulnerable.

c. Dashboard: Top Values of http.uri

- Around **165 connection attempts** were made to connectivitycheck.ubuntu.com, suggesting routine OS-level connectivity checks (Ubuntu-based machines).
- example.com was accessed **only once**, likely a placeholder or test.

- Approximately **10 hits** were made to detectportal.firefox.com/success, indicating Firefox browser's captive portal detection in action.

d. Dashboard: Top Values of destination.ip

- **192.168.89.139** received the highest traffic (~1250 packets), indicating it might be the **local host** or a heavily used internal resource.
- Significant IPv6 traffic seen:
  - 2405:200:800::1 – ~1100 records.
  - 2401:4900:5a3b:90b5::22 – ~1000 records.
  - 2401:4900:a67f:bda3::42 – ~500 records.
- External IP 34.107.243.93 logged ~750 packets.
- A broad category of **'other' IPs** make up ~4500 records, indicating distributed communication patterns.

e. Insights: Destination Port Distribution

- **Port 53 (DNS)** tops with **4500+ records**, suggesting high DNS usage or potential tunneling.
- **Port 443 (HTTPS)** has ~1750 records — typical for secure web traffic.
- **Port 0** (~750 records) is abnormal; often linked to **malformed packets** or **misconfigured systems**.
- **Port 80 (HTTP)** and **53153** follow with moderate traffic; others collectively contribute ~100.

4.3. Recommendations based on Analysis from Wireshark:

a. Enforce HTTPS: Redirect all HTTP traffic to HTTPS using SSL/TLS to ensure secure communication.

b. Use DPI Tools: Regularly inspect traffic with Wireshark or Arkime to detect anomalies.

c. Enable IDS/IPS: Deploy Snort or Suricata with Logstash to flag suspicious activity.

d. Restrict Open Ports: Close unused ports to minimize exploitation risks.

e. Monitor IPv6 Traffic: Apply strict rules and monitoring for IPv6, not just IPv4.

f. Protect Endpoints: Harden devices exchanging unencrypted traffic and monitor them actively.

4.4 Recommendations based on Analysis from Kibana Dashboard:

a. Block or monitor CLI-based user-agents like curl, and whitelist only trusted ones.

b. Increase encrypted communication (TLS), and closely inspect abnormal DNS/UDP traffic for tunneling or data leakage.

c. Introduce IP categorization (internal, external, cloud provider, unknown) and map IPs to organizations or geolocations. This can help detect unauthorized external communications or uncover suspicious destinations.

d. Add a **filter for system-generated URIs** (like Ubuntu connectivity check or Firefox captive portal) to segregate human-initiated browsing from background traffic. This will improve visibility into actual user browsing behavior.

e. Investigate traffic on **Port 53 and Port 0** — validate DNS activity and examine Port 0 usage for misconfigurations or reconnaissance attempts.

5. Links & References:

- **Full Screenshot Archive**: Explore 100+ unedited screenshots covering setup, error handling, skipped visualizations, and complete configurations in my GitHub repository: github.com/YatharthKumarSaxena/Cybersecurity-Screenshots *(My own detailed work log.)*
- **Moloch Official Documentation**: https://github.com/aol/moloch
- **Elastic Stack Overview**: https://www.elastic.co/what-is/elk-stack
- **Wireshark Overview**: https://www.wireshark.org/learn

🔸 *Note: All analysis presented here is a trimmed view. For deeper evidence of learning, execution, and problems addressed — please refer to the GitHub link above.*