Université Euro-Méditerranéenne Fès
Euromed University of Fez

Ecole d'Ingénierie Digitale et d'Intelligence Artificielle
(EIDIA)

Report

Analyzing Network Traffic using ELK

Elasticsearch **Logstash** **Kibana**

Supervised by :
Mr. Airaj Mohammed

Prepared by :
Fatima BOUYARMANE
Wissal BOUTAYEB

Academic year **:** 2024-2025

Summary

I- Introduction

Monitoring and analyzing network traffic is essential for keeping computer systems secure and running smoothly. As networks become more complex, organizations need effective tools to capture and understand the data flowing through them. The ELK stack, which includes Elasticsearch, Logstash, and Kibana, provides a powerful way to do this.

Elasticsearch is a search engine that helps store and analyze large amounts of data quickly. Logstash is used to collect and process data from different sources, while Kibana offers a user-friendly interface for visualizing that data. Together, these tools allow users to see patterns in network traffic, identify potential security issues, and improve overall network performance.

This project focuses on using the ELK stack to analyze network traffic. By setting up this system, organizations can gain better insights into their network activities and make informed decisions to enhance security and efficiency. The following sections will explain how to set up the ELK stack, capture network data, and create visualizations to help understand that data.

II- Objectives

1. **Install and Configure the ELK Stack:** Set up Elasticsearch, Logstash, and Kibana on a Linux system to create a functional environment for data analysis.

2. **Capture Network Traffic:** Use tools like tcpdump to capture network packets and save them in a format suitable for analysis.

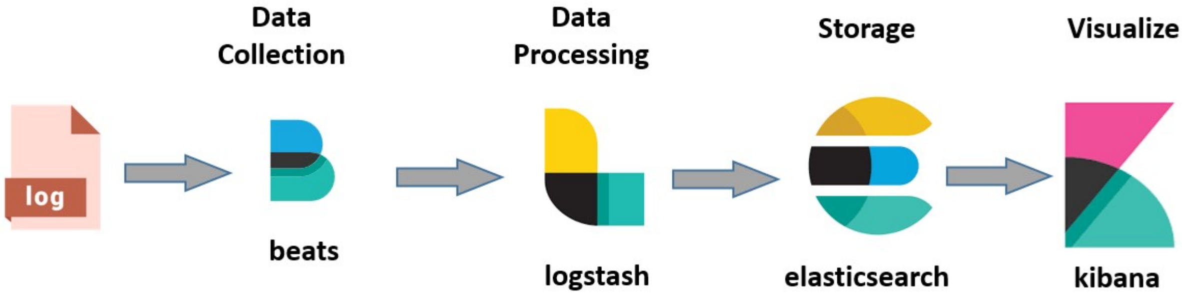
3. **Process and Index Data:** Develop a Python script to convert captured PCAP files into JSON format and index the data into Elasticsearch for efficient searching and querying.

4. **Create Visualizations:** Utilize Kibana to create visual representations of network traffic, including charts and graphs that display protocol distributions, top source IPs, and traffic patterns.

5. **Enhance Monitoring and Reporting**: Implement strategies to monitor the ELK stack's performance and ensure it operates smoothly, allowing for ongoing analysis and reporting of network traffic.

6. **Identify Security Threats:** Analyze the visualized data to detect anomalies and potential security threats within the network traffic.

III- ELK Architecture



This diagram illustrates the **ELK Stack (Elasticsearch, Logstash, Kibana)** workflow, which is a powerful solution for log and data management. Here's an explanation of the components and process:

1. Data Collection (Beats)

- **Input**: Logs or other data sources are the raw input for this system. These can be system logs, application logs, network logs, or any kind of structured/unstructured data.
- **Beats**: Beats are lightweight data shippers that collect logs or other data from different sources and forward them to Logstash or Elasticsearch for further processing.
  Examples of Beats include:
  - **Filebeat** for log files.
  - **Metricbeat** for system metrics.
  - **Packetbeat** for network data.

2. Data Processing (Logstash)

- **Logstash** is a server-side data processing pipeline. It receives data from Beats, processes it, and forwards it to Elasticsearch.
  - It can parse, transform, and enrich data (e.g., remove unnecessary fields, convert formats, or structure unstructured data).
  - Example tasks include filtering, data normalization, and adding metadata.

3. Storage (Elasticsearch)

- **Elasticsearch** is a distributed search and analytics engine where processed data is stored.
    - It organizes and indexes data, making it easily searchable.
    - It provides powerful capabilities for querying and real-time analysis of stored data.

4. Visualization (Kibana)

- **Kibana** is a visualization tool that works with Elasticsearch.
    - It provides dashboards, charts, and graphs to analyze the stored data.
    - Users can interactively search and drill down into data for insights.

IV- Installation

1. Elasticsearch

1.1 System Requirements

Ensure that your system meets the following requirements:

- Java 11 or later (Elasticsearch requires Java to run).
- At least 4 GB of RAM (8 GB or more is recommended for production).

1.2 Install Elasticsearch

```
┌──(kali㉿kali)-[~]
└─$ wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/elasticsearc
h-keyring.gpg

┌──(kali㉿kali)-[~]
└─$ echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg] https://artifacts.elastic.co/packages/8.x/apt stab
le main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.list
deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg] https://artifacts.elastic.co/packages/8.x/apt stable main
```

This step ensures that packages installed from the Elastic repository are authenticated using the GPG key, preventing tampered or unverified software from being installed.

```
┌──(kali㉿kali)-[~]
└─$ wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.17.14-amd64.deb

--2024-11-06 10:27:18--  https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.17.14-amd64.deb
Resolving artifacts.elastic.co (artifacts.elastic.co)... 34.120.127.130, 2600:1901:0:1d7::
Connecting to artifacts.elastic.co (artifacts.elastic.co)|34.120.127.130|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 322663304 (308M) [binary/octet-stream]
Saving to: 'elasticsearch-7.17.14-amd64.deb'

elasticsearch-7.17.14-amd64.de 100%[===================================>] 307.71M  1.88MB/s    in 2m 8s

2024-11-06 10:29:27 (2.40 MB/s) - 'elasticsearch-7.17.14-amd64.deb' saved [322663304/322663304]
```

- The wget command downloads the specified .deb package for Elasticsearch version 7.17.14 for AMD64 architecture from the official Elastic website.

```
┌──(kali㉿kali)-[~]
└─$ sudo dpkg -i elasticsearch-7.17.14-amd64.deb

Selecting previously unselected package elasticsearch.
(Reading database ... 517881 files and directories currently installed.)
Preparing to unpack elasticsearch-7.17.14-amd64.deb ...
Creating elasticsearch group... OK
Creating elasticsearch user... OK
Unpacking elasticsearch (7.17.14) ...
Setting up elasticsearch (7.17.14) ...
### NOT starting on installation, please execute the following statements to configure elasticsearch service to start auto
matically using systemd
 sudo systemctl daemon-reload
 sudo systemctl enable elasticsearch.service
### You can start elasticsearch service by executing
 sudo systemctl start elasticsearch.service
Created elasticsearch keystore in /etc/elasticsearch/elasticsearch.keystore
```

- This command installs the downloaded Elasticsearch package using dpkg. The output indicates that the package is being unpacked and installed.

```
┌──(kali㉿kali)-[~]
└─$ sudo apt --fix-broken install
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 33
```

- This command is used to fix any broken dependencies or installations. The output indicates that there are no packages to upgrade or install, suggesting that the installation was successful.

```
┌──(kali㉿kali)-[~]
└─$ sudo mkdir -p /etc/elasticsearch
sudo mkdir -p /var/lib/elasticsearch
sudo mkdir -p /var/log/elasticsearch
```

We created the necessary directories for Elasticsearch configuration, data storage, and logs. The -p option ensures that no error is thrown if the directory already exists.

Create the elasticsearch.yml file with cluster and node settings:

```
┌──(kali㉿kali)-[~]
└─$ sudo bash -c 'cat > /etc/elasticsearch/elasticsearch.yml << EOL
cluster.name: kali-cluster
node.name: node-1
path.data: /var/lib/elasticsearch
path.logs: /var/log/elasticsearch
network.host: localhost
http.port: 9200
discovery.type: single-node
EOL'
```

Create the jvm.options file to configure Java memory settings:

```
┌──(kali㉿kali)-[~]
└─$ sudo bash -c 'cat > /etc/elasticsearch/jvm.options << EOL
-Xms512m
-Xmx512m
-XX:+UseG1GC
-XX:G1ReservePercent=25
-XX:InitiatingHeapOccupancyPercent=30
EOL'
```

Change ownership of the configuration, data, and log directories to the elasticsearch user:

```
┌──(kali㉿kali)-[~]
└─$ sudo chown -R elasticsearch:elasticsearch /etc/elasticsearch
sudo chown -R elasticsearch:elasticsearch /var/lib/elasticsearch
sudo chown -R elasticsearch:elasticsearch /var/log/elasticsearch
```

Increase the maximum number of memory map areas:

```
┌──(kali㉿kali)-[~]
└─$ sudo sysctl -w vm.max_map_count=262144
vm.max_map_count = 262144
```

Reload systemd to apply changes and start the elasticsearch service:

```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl daemon-reload

┌──(kali㉿kali)-[~]
└─$ sudo systemctl start elasticsearch
```

The elasticsearch service status is running :

```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl status elasticsearch

● elasticsearch.service - Elasticsearch
     Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; disabled; preset: disabled)
     Active: active (running) since Wed 2024-11-06 10:31:20 EST; 7s ago
 Invocation: 3dba45d226cd493b90c18b4f79dbc921
       Docs: https://www.elastic.co
   Main PID: 107147 (java)
      Tasks: 66 (limit: 4596)
     Memory: 797.6M (peak: 799.9M)
        CPU: 45.536s
     CGroup: /system.slice/elasticsearch.service
             ├─107147 /usr/share/elasticsearch/jdk/bin/java -Xshare:auto -Des.networkaddress.cache.ttl=60 -Des.networkadd>
             └─107352 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-x86_64/bin/controller

Nov 06 10:31:04 kali systemd[1]: Starting elasticsearch.service - Elasticsearch...
Nov 06 10:31:07 kali systemd-entrypoint[107147]: Nov 06, 2024 10:31:07 AM sun.util.locale.provider.LocaleProviderAdapter >
Nov 06 10:31:07 kali systemd-entrypoint[107147]: WARNING: COMPAT locale provider will be removed in a future release
Nov 06 10:31:20 kali systemd[1]: Started elasticsearch.service - Elasticsearch.
```

```
┌──(kali㉿kali)-[~]
└─$ curl http://localhost:9200
{
  "name" : "node-1",
  "cluster_name" : "kali-cluster",
  "cluster_uuid" : "fJqWMnNUQWGtA_J3lXSYgg",
  "version" : {
    "number" : "7.17.14",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "774e3bfa4d52e2834e4d9d8d669d77e4e5c1017f",
    "build_date" : "2023-10-05T22:17:33.780167078Z",
    "build_snapshot" : false,
    "lucene_version" : "8.11.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

The output confirms that Elasticsearch is running correctly on the local machine, providing key details about the node, cluster, and version.

It indicates that your Elasticsearch instance is operational and accessible.

2. Logstash

Installing Logstash using the command:

```
┌──(kali㉿kali)-[~]
└─$ sudo apt-get install logstash
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  logstash
0 upgraded, 1 newly installed, 0 to remove and 281 not upgraded.
Need to get 421 MB of archives.
After this operation, 698 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/8.x/apt stable/main amd64 logstash amd64 1:8.15.4-1 [421 MB]
Fetched 421 MB in 47s (8,962 kB/s)
Selecting previously unselected package logstash.
(Reading database ... 478540 files and directories currently installed.)
Preparing to unpack .../logstash_1%3a8.15.4-1_amd64.deb ...
Unpacking logstash (1:8.15.4-1) ...
Setting up logstash (1:8.15.4-1) ...
```

Starting the elasticsearch service and viewing its status which indicates that it is running

```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl start logstash

┌──(kali㉿kali)-[~]
└─$ sudo systemctl status logstash
● logstash.service - logstash
     Loaded: loaded (/usr/lib/systemd/system/logstash.service; disabled; preset: disabled)
     Active: active (running) since Tue 2024-11-12 11:29:40 EST; 5s ago
 Invocation: e3b891010e2d4bb08b6d189cb3807c1a
   Main PID: 194951 (java)
      Tasks: 22 (limit: 4546)
     Memory: 229.8M (peak: 234.9M)
        CPU: 10.783s
     CGroup: /system.slice/logstash.service
             └─194951 /usr/share/logstash/jdk/bin/java -Xms1g -Xmx1g -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Djruby.compile.invoke

Nov 12 11:29:40 kali systemd[1]: Started logstash.service - logstash.
Nov 12 11:29:40 kali logstash[194951]: Using bundled JDK: /usr/share/logstash/jdk
```

Enabling Logstash service

```
┌──(kali㉿kali)-[~/Project_ELK]
└─$ sudo systemctl enable logstash
Created symlink '/etc/systemd/system/multi-user.target.wants/logstash.service' → '/usr/lib/systemd/system/logstash.service'.

┌──(kali㉿kali)-[~/Project_ELK]
└─$ sudo systemctl status logstash
● logstash.service - logstash
     Loaded: loaded (/usr/lib/systemd/system/logstash.service; enabled; preset: disabled)
     Active: active (running) since Tue 2024-11-12 11:38:15 EST; 415ms ago
 Invocation: 49c417767a2742be9d285b4f6c2c072d
   Main PID: 201108 (java)
      Tasks: 19 (limit: 4546)
     Memory: 38.1M (peak: 38.1M)
        CPU: 588ms
     CGroup: /system.slice/logstash.service
             └─201108 /usr/share/logstash/jdk/bin/java -Xms1g -Xmx1g -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Djruby.compile.invokedyna

Nov 12 11:38:15 kali systemd[1]: logstash.service: Scheduled restart job, restart counter is at 25.
Nov 12 11:38:15 kali systemd[1]: Started logstash.service - logstash.
Nov 12 11:38:15 kali logstash[201108]: Using bundled JDK: /usr/share/logstash/jdk
```

Configuration of logstash network file:

```
┌──(kali㉿kali)-[~/Project_ELK]
└─$ sudo nano /etc/logstash/conf.d/network.conf
```

```
  GNU nano 8.2                                                                          /etc/logstash/conf.d/network.conf
# /etc/logstash/conf.d/network.conf
input {
  file {
    path => "/home/kali/Project_ELK/network_analysis/network_data.json"
    start_position => "beginning"
    sincedb_path => "/dev/null"
    codec => json
    type => "pcap"
  }
}

filter {
  if [type] == "pcap" {
    date {
      match => [ "timestamp", "ISO8601" ]
      target => "@timestamp"
    }

    geoip {
      source => "source_ip"
      target => "source_location"
    }

    # Enrichissement supplémentaire
    mutate {
      add_field => {
        "processed_by" => "logstash"
        "analysis_type" => "enhanced"
      }
    }
  }
}

output {
  elasticsearch {
    hosts => ["localhost:9200"]
    index => "network-analysis-enriched-%{+YYYY.MM.dd}"
  }
  stdout { codec => rubydebug }
}
```

This Logstash configuration file is designed to read network data from a JSON file, process it by parsing timestamps and enriching it with geographical information, and then output the processed data to Elasticsearch while also displaying it in the console for debugging purposes. This setup is useful for analyzing network traffic and gaining insights from the data.

Testing the Logstash configuration file for processing network data.

```
┌──(venv)─(kali㉿kali)-[~/Project_ELK]
└─$ sudo nano /etc/logstash/conf.d/network.conf

┌──(venv)─(kali㉿kali)-[~/Project_ELK]
└─$ sudo chown -R logstash:logstash /etc/logstash/conf.d

┌──(venv)─(kali㉿kali)-[~/Project_ELK]
└─$ sudo chmod 644 /etc/logstash/conf.d/network.conf

┌──(venv)─(kali㉿kali)-[~/Project_ELK]
└─$ sudo /usr/share/logstash/bin/logstash -t -f /etc/logstash/conf.d/network.conf
Using bundled JDK: /usr/share/logstash/jdk
WARNING: Could not find logstash.yml which is typically located in $LS_HOME/config or /etc/logstash. You can specify the path using --path.settings. Continuing using the defaults
Could not find log4j2 configuration at path /usr/share/logstash/config/log4j2.properties. Using default config which logs errors to the console
[WARN ] 2024-11-12 12:05:09.852 [main] runner - NOTICE: Running Logstash as superuser is not recommended and won't be allowed in the future. Set 'allow_superuser' to 'false' to avoid s
leases.
[INFO ] 2024-11-12 12:05:09.870 [main] runner - Starting Logstash {"logstash.version"=>"8.15.4", "jruby.version"=>"jruby 9.4.9.0 (3.1.4) 2024-11-04 547c6b150e OpenJDK 64-Bit Server VM
-LTS +indy +jit [x86_64-linux]"}
[INFO ] 2024-11-12 12:05:09.874 [main] runner - JVM bootstrap flags: [-Xms1g, -Xmx1g, -Djava.awt.headless=true, -Dfile.encoding=UTF-8, -Djruby.compile.invokedynamic=true, -XX:+HeapDump
.security.egd=file:/dev/urandom, -Dlog4j2.isThreadContextMapInheritable=true, -Dlogstash.jackson.stream-read-constraints.max-string-length=200000000, -Dlogstash.jackson.stream-read-con
=10000, -Djruby.regexp.interruptible=true, -Djdk.io.File.enableADS=true, --add-exports=jdk.compiler/com.sun.tools.javac.api=ALL-UNNAMED, --add-exports=jdk.compiler/com.sun.tools.javac.
ports=jdk.compiler/com.sun.tools.javac.parser=ALL-UNNAMED, --add-exports=jdk.compiler/com.sun.tools.javac.tree=ALL-UNNAMED, --add-exports=jdk.compiler/com.sun.tools.javac.util=ALL-UNNA
/java.security=ALL-UNNAMED, --add-opens=java.base/java.io=ALL-UNNAMED, --add-opens=java.base/java.nio.channels=ALL-UNNAMED, --add-opens=java.base/sun.nio.ch=ALL-UNNAMED, --add-opens=ja
nt=ALL-UNNAMED, -Dio.netty.allocator.maxOrder=11]
[INFO ] 2024-11-12 12:05:09.878 [main] runner - Jackson default value override `logstash.jackson.stream-read-constraints.max-string-length` configured to `200000000`
[INFO ] 2024-11-12 12:05:09.879 [main] runner - Jackson default value override `logstash.jackson.stream-read-constraints.max-number-length` configured to `10000`
[INFO ] 2024-11-12 12:05:09.894 [main] settings - Creating directory {:setting=>"path.queue", :path=>"/usr/share/logstash/data/queue"}
[INFO ] 2024-11-12 12:05:09.902 [main] settings - Creating directory {:setting=>"path.dead_letter_queue", :path=>"/usr/share/logstash/data/dead_letter_queue"}
[WARN ] 2024-11-12 12:05:10.225 [LogStash::Runner] multilocal - Ignoring the 'pipelines.yml' file because modules or command line options are specified
[INFO ] 2024-11-12 12:05:10.840 [LogStash::Runner] Reflections - Reflections took 174 ms to scan 1 urls, producing 138 keys and 481 values
[INFO ] 2024-11-12 12:05:11.067 [LogStash::Runner] json - ECS compatibility is enabled but `target` option was not specified. This may cause fields to be set at the top-level of the ev
to clash with the Elastic Common Schema. It is recommended to set the `target` option to avoid potential schema conflicts (if your data is ECS compliant or non-conflicting, feel free t
[INFO ] 2024-11-12 12:05:11.589 [LogStash::Runner] javapipeline - Pipeline `main` is configured with `pipeline.ecs_compatibility: v8` setting. All plugins in this pipeline will default
v8` unless explicitly configured otherwise.
Configuration OK
[INFO ] 2024-11-12 12:05:11.590 [LogStash::Runner] runner - Using config.test_and_exit mode. Config Validation Result: OK. Exiting Logstash
```

The output provides useful information about the Logstash environment and any potential issues that may need to be addressed.

3. Kibana

- The wget command downloads the specified .deb package for Kibana version 7.17.14 for AMD64 architecture from the official Elastic website.

```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl daemon-reload

┌──(kali㉿kali)-[~]
└─$ wget https://artifacts.elastic.co/downloads/kibana/kibana-7.17.14-amd64.deb

--2024-11-06 10:35:18--  https://artifacts.elastic.co/downloads/kibana/kibana-7.17.14-amd64.deb
Resolving artifacts.elastic.co (artifacts.elastic.co)... 34.120.127.130, 2600:1901:0:1d7::
Connecting to artifacts.elastic.co (artifacts.elastic.co)|34.120.127.130|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 297999604 (284M) [binary/octet-stream]
Saving to: 'kibana-7.17.14-amd64.deb'

kibana-7.17.14-amd64.deb       100%[===================================>] 284.19M  2.04MB/s    in 2m 47s

2024-11-06 10:38:05 (1.71 MB/s) - 'kibana-7.17.14-amd64.deb' saved [297999604/297999604]
```

- This command installs the downloaded Kibana package using dpkg. The output indicates that the package is being unpacked and installed.

```
┌──(kali㉿kali)-[~]
└─$ sudo dpkg -i kibana-7.17.14-amd64.deb
sudo apt --fix-broken install
Selecting previously unselected package kibana.
(Reading database ... 420023 files and directories currently installed.)
Preparing to unpack kibana-7.17.14-amd64.deb ...
Unpacking kibana (7.17.14) ...
Setting up kibana (7.17.14) ...
Creating kibana group... OK
Creating kibana user... OK
Kibana is currently running with legacy OpenSSL providers enabled! For details and instructions on how to disable see http
s://www.elastic.co/guide/en/kibana/7.17/production.html#openssl-legacy-provider
Created Kibana keystore in /etc/kibana/kibana.keystore
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 34
```

Configuration file of Kibana:

```
┌──(kali㉿kali)-[~]
└─$ sudo apt --fix-broken install

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 34

┌──(kali㉿kali)-[~]
└─$ sudo bash -c 'cat > /etc/kibana/kibana.yml << EOL
server.port: 5601
server.host: "localhost"
elasticsearch.hosts: ["http://localhost:9200"]
EOL'
```

- **server.port**: Sets the port for Kibana to listen on (5601 is the default).
- **server.host:** Configures Kibana to be accessible from localhost.
- **elasticsearch.hosts:** Specifies the Elasticsearch instance that Kibana will connect to, which is running on localhost:9200.

Starting the service, and enabling it to run on system boot.

```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl start kibana

┌──(kali㉿kali)-[~]
└─$ sudo systemctl enable kibana
Synchronizing state of kibana.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable kibana
Created symlink '/etc/systemd/system/multi-user.target.wants/kibana.service' → '/etc/systemd/system/kibana.service'.
```

```
┌──(kali㉿kali)-[/etc/logstash/conf.d]
└─$ curl http://localhost:5601/app/home
<!DOCTYPE html><html lang="en"><head><meta charSet="utf-8"/><meta http-equiv="X-UA-Compatible" content="IE=edg
ewport" content="width=device-width"/><title>Elastic</title><style>

        @font-face {
          font-family: 'Inter';
          font-style: normal;
          font-weight: 100;
          src: url('/ui/fonts/inter/Inter-Thin.woff2') format('woff2'), url('/ui/fonts/inter/Inter-Thin.woff')
        }

        @font-face {
          font-family: 'Inter';
          font-style: italic;
          font-weight: 100;
          src: url('/ui/fonts/inter/Inter-ThinItalic.woff2') format('woff2'), url('/ui/fonts/inter/Inter-ThinI
;
        }
```

The curl command successfully retrieved the HTML content of the Kibana home page, confirming that the Kibana service is running and accessible at http://localhost:5601.

4. Filebeat :

Filebeat is a lightweight data shipper designed to forward and centralize log data. It is part of the Elastic Stack (ELK Stack) and is used to collect logs from various sources and send them to Elasticsearch or Logstash for further processing and analysis.

Downloading Filebeat, which is often used alongside the ELK stack to ship logs to Elasticsearch.

```
┌──(kali㉿kali)-[~]
└─$ wget https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.17.14-amd64.deb
--2024-11-06 10:39:26--  https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.17.14-amd64.deb
Resolving artifacts.elastic.co (artifacts.elastic.co)... 34.120.127.130, 2600:1901:0:1d7::
Connecting to artifacts.elastic.co (artifacts.elastic.co)|34.120.127.130|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 36012058 (34M) [binary/octet-stream]
Saving to: 'filebeat-7.17.14-amd64.deb'

filebeat-7.17.14-amd64.deb     100%[=====================================>]  34.34M  1010KB/s    in 14s

2024-11-06 10:39:41 (2.44 MB/s) - 'filebeat-7.17.14-amd64.deb' saved [36012058/36012058]
```

Unpacking the Filebeat package :

```
└─$ sudo dpkg -i filebeat-7.17.14-amd64.deb
Selecting previously unselected package filebeat.
(Reading database ... 474157 files and directories currently installed.)
Preparing to unpack filebeat-7.17.14-amd64.deb ...
Unpacking filebeat (7.17.14) ...
Setting up filebeat (7.17.14) ...
Processing triggers for kali-menu (2024.3.1) ...
```

Restarting and checking the status of Filebeat:

```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl restart  filebeat

┌──(kali㉿kali)-[~]
└─$ sudo systemctl start  filebeat

┌──(kali㉿kali)-[~]
└─$ sudo systemctl status  filebeat
● filebeat.service - Filebeat sends log files to Logstash or directly to Elasticsearch.
     Loaded: loaded (/usr/lib/systemd/system/filebeat.service; disabled; preset: disabled)
     Active: active (running) since Wed 2024-11-13 13:42:14 EST; 16s ago
 Invocation: 6d02290ea959496a83293e41dbc80ded
       Docs: https://www.elastic.co/beats/filebeat
   Main PID: 27366 (filebeat)
      Tasks: 9 (limit: 2208)
     Memory: 112.9M (peak: 115M)
        CPU: 728ms
     CGroup: /system.slice/filebeat.service
             └─27366 /usr/share/filebeat/bin/filebeat --environment systemd -c /etc/filebeat/filebeat.yml --path.home /usr/shar>

Nov 13 13:42:15 kali filebeat[27366]: 2024-11-13T13:42:15.606-0500       INFO       instance/beat.go:457       filebeat star>
Nov 13 13:42:15 kali filebeat[27366]: 2024-11-13T13:42:15.610-0500       INFO       memlog/store.go:119       Loading data f>
Nov 13 13:42:15 kali filebeat[27366]: 2024-11-13T13:42:15.610-0500       INFO       memlog/store.go:124       Finished loadi>
Nov 13 13:42:15 kali filebeat[27366]: 2024-11-13T13:42:15.611-0500       INFO       [registrar]       registrar/registrar.go>
Nov 13 13:42:15 kali filebeat[27366]: 2024-11-13T13:42:15.611-0500       INFO       [crawler]       beater/crawler.go:71    >
Nov 13 13:42:15 kali filebeat[27366]: 2024-11-13T13:42:15.611-0500       INFO       [crawler]       beater/crawler.go:117   >
Nov 13 13:42:15 kali filebeat[27366]: 2024-11-13T13:42:15.611-0500       INFO       [crawler]       beater/crawler.go:121   >
Nov 13 13:42:15 kali filebeat[27366]: 2024-11-13T13:42:15.612-0500       INFO       [crawler]       beater/crawler.go:106   >
Nov 13 13:42:15 kali filebeat[27366]: 2024-11-13T13:42:15.614-0500       INFO       cfgfile/reload.go:164       Config reloa>
Nov 13 13:42:15 kali filebeat[27366]: 2024-11-13T13:42:15.616-0500       INFO       cfgfile/reload.go:224       Loading of c>
lines 1-22/22 (END)
```

This output indicates that Filebeat is running successfully.

V- Environment Setup

Creating a Python virtual environment

To install Python and the venv module, we used the command:

sudo apt install python3 python3-venv

To create the environment:

```
┌──(kali㉿kali)-[~/Project_ELK/network_analysis]
└─$ python3 -m venv venv
```

To activate the created environment, we used:

```
┌──(kali㉿kali)-[~/Project_ELK/network_analysis]
└─$ source venv/bin/activate
```

After activating the environment, we can install python packages.

```
┌──(venv)─(kali㉿kali)-[~/Project_ELK/network_analysis]
└─$ pip install elasticsearch pyshark
Collecting elasticsearch
  Using cached elasticsearch-8.15.1-py3-none-any.whl.metadata (8.7 kB)
Collecting pyshark
  Using cached pyshark-0.6-py3-none-any.whl.metadata (806 bytes)
Collecting elastic-transport<9,⩾8.13 (from elasticsearch)
  Using cached elastic_transport-8.15.1-py3-none-any.whl.metadata (3.7 kB)
Collecting lxml (from pyshark)
  Using cached lxml-5.3.0-cp312-cp312-manylinux_2_28_x86_64.whl.metadata (3.8 kB)
Collecting termcolor (from pyshark)
  Using cached termcolor-2.5.0-py3-none-any.whl.metadata (6.1 kB)
Collecting packaging (from pyshark)
  Using cached packaging-24.2-py3-none-any.whl.metadata (3.2 kB)
Collecting appdirs (from pyshark)
  Using cached appdirs-1.4.4-py2.py3-none-any.whl.metadata (9.0 kB)
Collecting urllib3<3,⩾1.26.2 (from elastic-transport<9,⩾8.13→elasticsearch)
  Using cached urllib3-2.2.3-py3-none-any.whl.metadata (6.5 kB)
Collecting certifi (from elastic-transport<9,⩾8.13→elasticsearch)
  Using cached certifi-2024.8.30-py3-none-any.whl.metadata (2.2 kB)
Using cached elasticsearch-8.15.1-py3-none-any.whl (524 kB)
Using cached pyshark-0.6-py3-none-any.whl (41 kB)
Using cached elastic_transport-8.15.1-py3-none-any.whl (64 kB)
Using cached appdirs-1.4.4-py2.py3-none-any.whl (9.6 kB)
Using cached lxml-5.3.0-cp312-cp312-manylinux_2_28_x86_64.whl (4.9 MB)
Using cached packaging-24.2-py3-none-any.whl (65 kB)
Using cached termcolor-2.5.0-py3-none-any.whl (7.8 kB)
Using cached urllib3-2.2.3-py3-none-any.whl (126 kB)
Using cached certifi-2024.8.30-py3-none-any.whl (167 kB)
Installing collected packages: appdirs, urllib3, termcolor, packaging, lxml, certifi, pyshark, elastic-transport, elasticsearch
Successfully installed appdirs-1.4.4 certifi-2024.8.30 elastic-transport-8.15.1 elasticsearch-8.15.1 lxml-5.3.0 packaging-24.2 pyshark-0.6 termcolor-2.5.0 urllib3-2.2.3
```

We installed pyshark package of python. This package is a wrapper for the Wireshark packet capture library, enabling us to analyze network packets in Python.

VI- Data Capture and Analysis

To capture and analyze network traffic, we created a python script to help us to do so:

nano analyze_pcap.py :

```
GNU nano 8.2                                                    analyze_pcap.py *
from elasticsearch import Elasticsearch
import pyshark
from datetime import datetime, timezone

def analyze_pcap(pcap_file):
    es = Elasticsearch(['http://localhost:9200'])
    capture = pyshark.FileCapture(pcap_file)

    for packet in capture:
        try:
            current_time = datetime.now(timezone.utc)
            packet_data = {
                'timestamp': current_time.isoformat(),
                'protocol': str(packet.highest_layer),
                'length': int(packet.length)
            }

            if hasattr(packet, 'ip'):
                packet_data.update({
                    'source_ip': str(packet.ip.src),
                    'dest_ip': str(packet.ip.dst)
                })

            if hasattr(packet, 'tcp'):
                packet_data.update({
                    'source_port': int(packet.tcp.srcport),
                    'dest_port': int(packet.tcp.dstport)
                })

            es.index(index='network-analysis', document=packet_data)
            print(f"Paquet indexé: {packet.highest_layer}")

        except Exception as e:
            print(f"Erreur: {e}")
            continue

    capture.close()

if __name__ == "__main__":
    analyze_pcap('capture.pcap')
```

This script analyzes a PCAP file (**capture.pcap**), extracts relevant information from each packet (such as timestamps, protocols, IP addresses, and ports), and indexes this data into Elasticsearch for further analysis. It uses the **Pyshark** library for packet capture and the Elasticsearch client for data indexing. If any errors occur during processing, they are caught and printed to the console.

Capturing network traffic:

We used the tcpdump command to capture network packets on the eth0 interface and save them to a file named capture.pcap. The capture was configured to run for 60 seconds, and since we set the option to keep only one file, it would overwrite the previous capture after each period.

```
┌──(kali㉿kali)-[~/Project_ELK]
└─$ sudo tcpdump -i eth0 -w capture.pcap -G 60 -W 1
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C29 packets captured
29 packets received by filter
0 packets dropped by kernel
```

During the capture, we successfully recorded 29 packets without any loss, as indicated by the output. This data can now be analyzed using tools like Wireshark or processed with our Python script to extract insights.

While running the tcpdump command we executed commands such as ping to test connectivity and nmap to scan for open ports on the target system. This simultaneous activity allowed us to generate network traffic for analysis while capturing the data with tcpdump.

```
┌──(kali㉿kali)-[~]
└─$ nmap localhost
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-12 11:51 EST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000070s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
9200/tcp open  wap-wsp

Nmap done: 1 IP address (1 host up) scanned in 0.10 seconds
```

```
┌──(kali㉿kali)-[~]
└─$ ping google.com
PING google.com (142.250.200.142) 56(84) bytes of data.
64 bytes from mad41s14-in-f14.1e100.net (142.250.200.142): icmp_seq=1 ttl=128 time=24.4 ms
64 bytes from mad41s14-in-f14.1e100.net (142.250.200.142): icmp_seq=2 ttl=128 time=38.3 ms
64 bytes from mad41s14-in-f14.1e100.net (142.250.200.142): icmp_seq=3 ttl=128 time=24.9 ms
64 bytes from mad41s14-in-f14.1e100.net (142.250.200.142): icmp_seq=4 ttl=128 time=32.0 ms
64 bytes from mad41s14-in-f14.1e100.net (142.250.200.142): icmp_seq=5 ttl=128 time=28.9 ms
64 bytes from mad41s14-in-f14.1e100.net (142.250.200.142): icmp_seq=6 ttl=128 time=24.8 ms
64 bytes from mad41s14-in-f14.1e100.net (142.250.200.142): icmp_seq=7 ttl=128 time=29.3 ms
64 bytes from mad41s14-in-f14.1e100.net (142.250.200.142): icmp_seq=8 ttl=128 time=32.9 ms
64 bytes from mad41s14-in-f14.1e100.net (142.250.200.142): icmp_seq=9 ttl=128 time=28.9 ms
64 bytes from mad41s14-in-f14.1e100.net (142.250.200.142): icmp_seq=10 ttl=128 time=31.4 ms
64 bytes from mad41s14-in-f14.1e100.net (142.250.200.142): icmp_seq=11 ttl=128 time=30.1 ms
64 bytes from mad41s14-in-f14.1e100.net (142.250.200.142): icmp_seq=12 ttl=128 time=25.6 ms
64 bytes from mad41s14-in-f14.1e100.net (142.250.200.142): icmp_seq=13 ttl=128 time=27.9 ms
64 bytes from mad41s14-in-f14.1e100.net (142.250.200.142): icmp_seq=14 ttl=128 time=30.4 ms
64 bytes from mad41s14-in-f14.1e100.net (142.250.200.142): icmp_seq=15 ttl=128 time=167 ms
64 bytes from mad41s14-in-f14.1e100.net (142.250.200.142): icmp_seq=16 ttl=128 time=27.9 ms
64 bytes from mad41s14-in-f14.1e100.net (142.250.200.142): icmp_seq=17 ttl=128 time=30.5 ms
^C
--- google.com ping statistics ---
17 packets transmitted, 17 received, 0% packet loss, time 16021ms
rtt min/avg/max/mdev = 24.398/37.371/167.224/32.634 ms
```

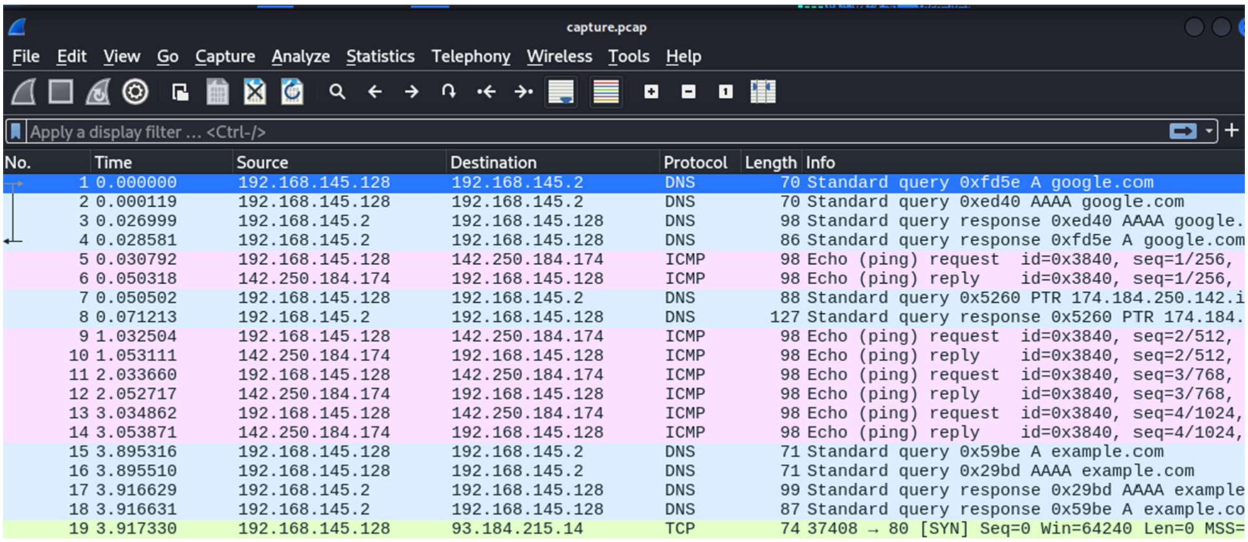
After capturing the network traffic, we analyzed the PCAP file using the python script which provided us with a report about the number of packets, the protocols, the IP addreses and ports.

```
┌──(venv)─(kali㉿kali)-[~/Project_ELK/network_analysis]
└─$ python analyze_pcap.py
/home/kali/Project_ELK/network_analysis/analyze_pcap.py:31: ElasticsearchWarning: Elasticsearch built-in security features
ee https://www.elastic.co/guide/en/elasticsearch/reference/7.17/security-minimal-setup.html to enable security.
  es.index(index='network-analysis', document=packet_data)
Paquet indexé: DNS
Paquet indexé: DNS
Paquet indexé: DNS
Paquet indexé: DNS
Paquet indexé: ICMP
Paquet indexé: ICMP
Paquet indexé: DNS
```

```
Paquet indexé: TCP
Paquet indexé: DATA-TEXT-LINES
Paquet indexé: TCP
Paquet indexé: TCP
Paquet indexé: TCP
Paquet indexé: TCP
Paquet indexé: TCP
Paquet indexé: ARP
Paquet indexé: ARP
Paquet indexé: ARP
Paquet indexé: ARP
Paquet indexé: ARP
Paquet indexé: ARP
Paquet indexé: ARP
Paquet indexé: ARP

=== Rapport d'Analyse ===

Total des paquets: 52

Top 5 Protocoles:
- ICMP: 16
- ARP: 15
- DNS: 10
- TCP: 9
- HTTP: 1

Top 5 IPs Sources:
- 192.168.204.128: 19
- 142.250.184.174: 8
- 192.168.204.2: 5
- 93.184.215.14: 5

Top 5 Ports:
- Port 59504: 6
- Port 80: 5

Rapport sauvegardé dans 'network_analysis_report.json'
```

Here are more information about the PCAP file:

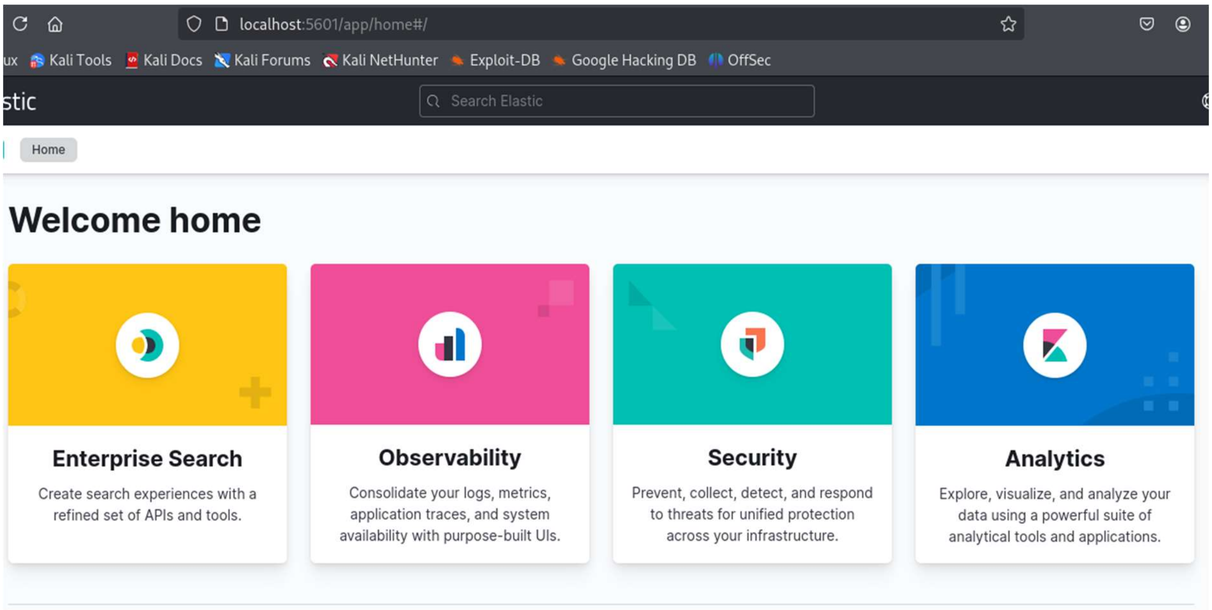
```
┌──(kali㉿kali)-[~/network_analysis]
└─$ capinfos capture.pcap
File name:           capture.pcap
File type:           Wireshark/tcpdump/... - pcap
File encapsulation:  Ethernet
File timestamp precision:  microseconds (6)
Packet size limit:   file hdr: 262144 bytes
Number of packets:   45
File size:           6,055 bytes
Data size:           5,311 bytes
Capture duration:    19.320220 seconds
First packet time:   2024-11-06 10:55:06.577778
Last packet time:    2024-11-06 10:55:25.897998
Data byte rate:      274 bytes/s
Data bit rate:       2,199 bits/s
Average packet size: 118.02 bytes
Average packet rate: 2 packets/s
SHA256:              33eba25be883c61bc5fd4d0d967c0e5a6e546853ffc07d29c004f4dc66ce497
SHA1:                a1a2e3e1e4ba1f87dbc72e6e8cc48b76dee99146
Strict time order:   True
Number of interfaces in file: 1
Interface #0 info:
                     Encapsulation = Ethernet (1 - ether)
                     Capture length = 262144
                     Time precision = microseconds (6)
                     Time ticks per second = 1000000
                     Number of stat entries = 0
                     Number of packets = 45
```



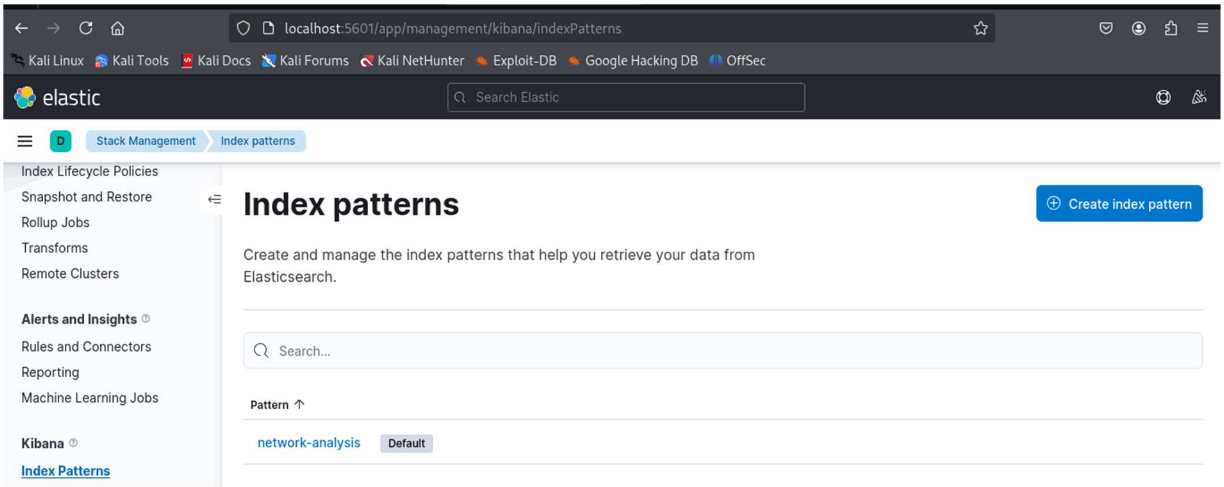
VII- Kibana Setup

1. Creating an index pattern in Kibana

To access Kibana use the URL http://localhost:5601

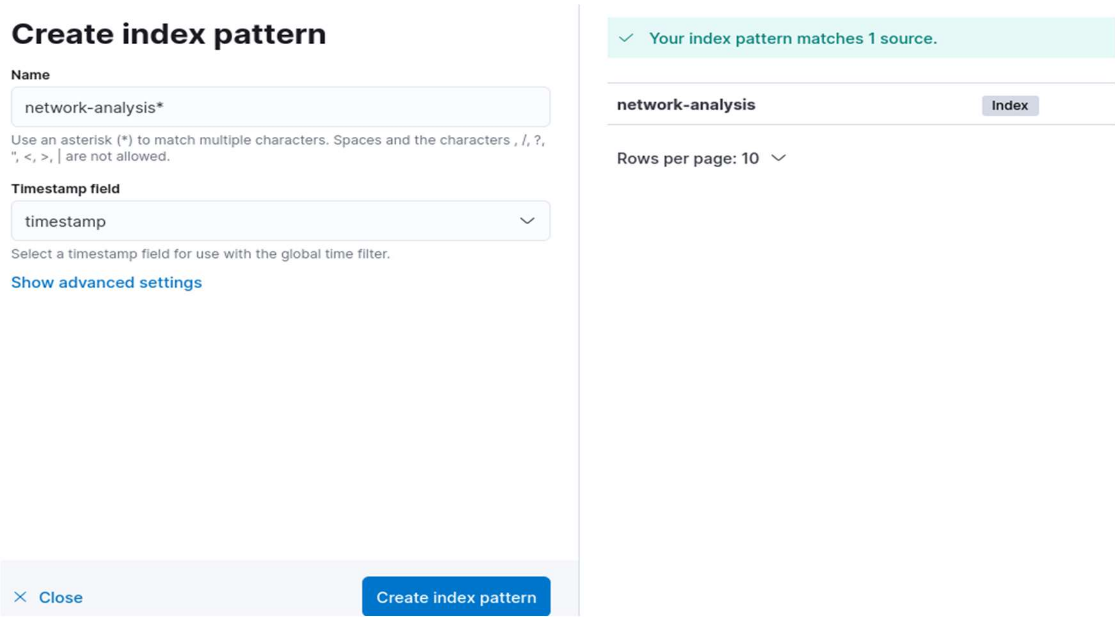


Go to: Menu → Stack Management → Index Patterns



Create index pattern:

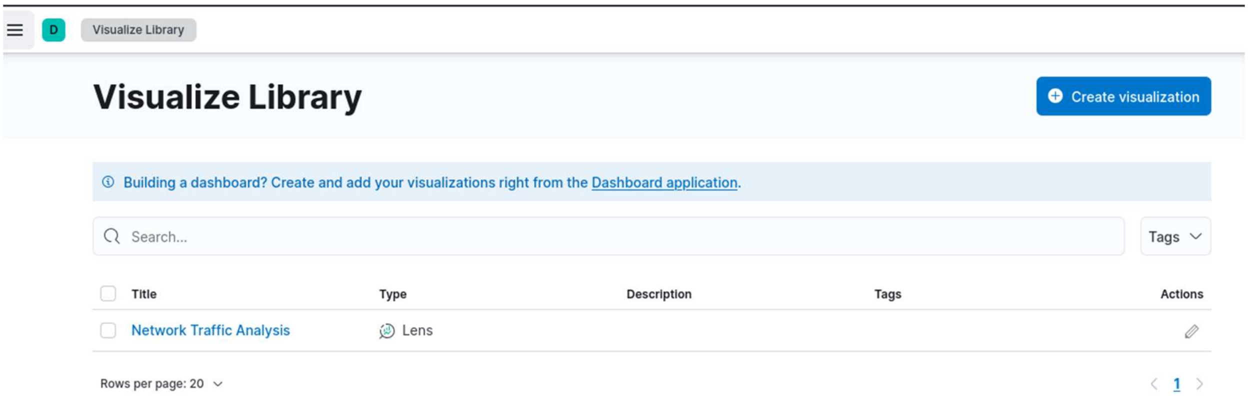
- **Pattern name:** network-analysis
- **Time field:** timestamp
- Create index pattern



2. Developing visualizations and compiling them into a dashboard

Menu → Visualize

- Create visualization

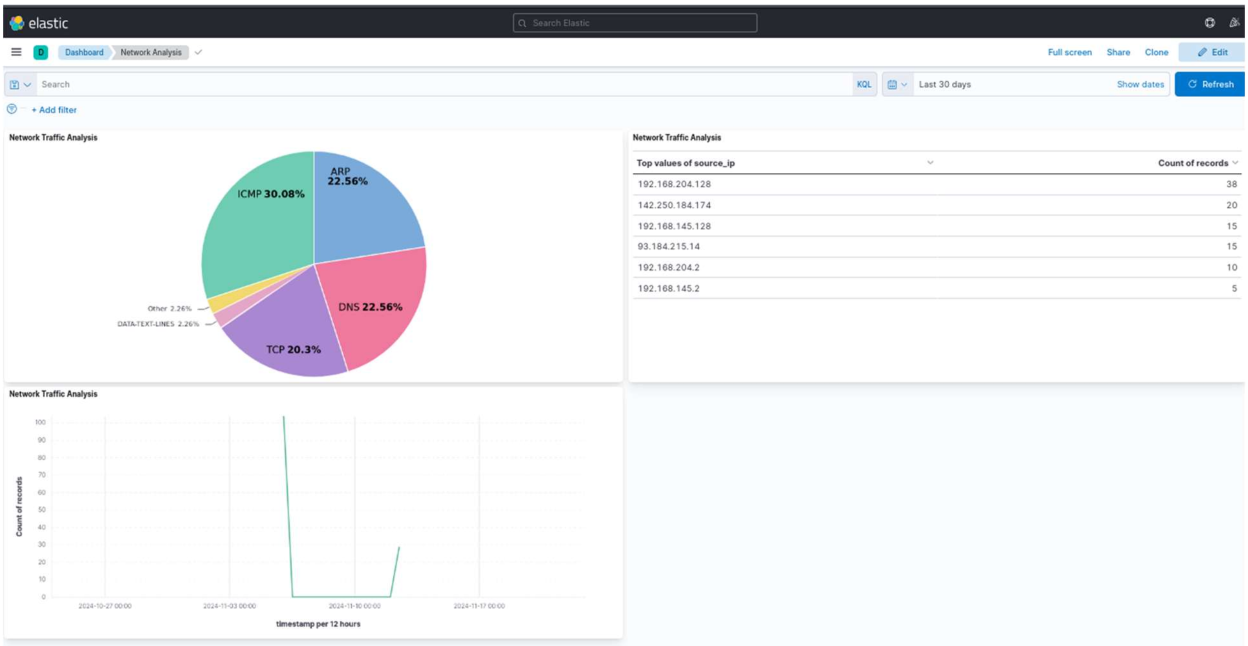


Create the visualizations:

- Distribution des Protocoles (Pie)
- Top IPs Sources (Data Table)
- Timeline du Trafic (Line)

3. Creation of the Dashboard:

- Menu → Dashboard
- Create dashboard
- Add → Add the created visualizations
- Save



VIII- Verification and Maintenance

1.Verifing data indexing

To check the indices in Elasticsearch, we used :

```
┌──(venv)-(kali㉿kali)-[~/network_analysis]
└─$ curl -X GET "localhost:9200/_cat/indices?v"
health status index                            uuid                   pri rep docs.count docs.deleted store.size pri.store
.size
green  open   .geoip_databases                 3EbMFPdsStKwcjLOdsZdnw   1   0         36            0     34.9mb           3
4.9mb
green  open   .apm-custom-link                 Zx6qezhrQomFPX5_eBDJqg   1   0          0            0       227b
 227b
yellow open   network-analysis                 s8TpsRvhRBSY6IAo5f879Q   1   1         97            0     22.2kb           2
2.2kb
green  open   .apm-agent-configuration         EF_l471pSOi8EhmfvmR73g   1   0          0            0       227b
 227b
green  open   .kibana_task_manager_7.17.14_001 V9IFevcMTx-KV4joG7S5rA   1   0         17         3658    610.4kb          61
0.4kb
green  open   .kibana_7.17.14_001              zjt_1jEoTXazVOPP4N56EQ   1   0         96          344      9.6mb
9.6mb
green  open   .async-search                    geIgB9_KRxKEcFtVOTmpnQ   1   0          2            2     12.9kb           1
2.9kb
green  open   .tasks                           X8wzCUfqTNyB6mtMZ_22Mw   1   0          4            0     21.4kb           2
1.4kb
```

To confirm that the expected data is present in the network-analysis index by querying it:

```
curl -X GET "localhost:9200/network-analysis/_search?pretty"
{
  "took" : 3,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 97,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "network-analysis",
        "_type" : "_doc",
        "_id" : "BjM1ApMBObgthoZfurbw",
        "_score" : 1.0,
        "_source" : {
          "timestamp" : "2024-11-06T11:01:35.820380",
          "protocol" : "DNS",
          "length" : 70,
          "source_ip" : "192.168.204.128",
          "dest_ip" : "192.168.204.2"
        }
      },
      {
```

2. Monitoring Logs for Troubleshooting

To monitor the logs to avoid errors, we use the command:

sudo tail -f /var/log/elasticsearch/elasticsearch.log

sudo tail -f /var/log/filebeat/filebeat.log

sudo tail -f /var/log/kibana/kibana.log

sudo tail -f /var/log/logstash/logstash-plain.log

IX- Challenges faced during the implementation of the project

Overview of Challenges

During the implementation of the ELK stack for our project, we encountered multiple technical challenges that hindered system stability and affected data processing and visualization. Below are the key challenges faced and how they were resolved:

Challenge 1: ELK Version Compatibility Issues

We encountered compatibility issues due to mismatched versions of Elasticsearch, Logstash, and Kibana. This caused connectivity errors and failure of data pipelines.

Impact:

- Kibana was unable to connect to Elasticsearch.
- Logstash pipelines failed to send data to Elasticsearch.

Cause:

- Inconsistent versions of ELK stack components.
- Breaking changes introduced in newer versions.

Resolution:

- Identified version compatibility using the Elastic Stack Compatibility Matrix.
- Upgraded all components to version 8.6.0 to ensure consistency.
- Adjusted configuration files to reflect updated syntax and options.

Challenge 2: Connection Issues on Port 9200

Elasticsearch service intermittently failed to establish connections on port 9200. This blocked communication with Logstash, Kibana, and external clients.

Impact:

Data ingestion and visualization were delayed.

Manual intervention was required to restore connectivity.

Cause:

Incorrect binding in the **elasticsearch.yml** configuration file.

Firewall restrictions blocking port 9200.

Elasticsearch service failures due to insufficient memory allocation.

Challenge 3: Frequent Service Restarts

The Elasticsearch service would restart repeatedly, causing downtime and loss of connection.

Impact:

Logs were not indexed on time.

Kibana dashboards showed incomplete data.

Cause:

Insufficient system resources (RAM, CPU).

Lack of proper monitoring tools to detect service issues early.

Resolution:

Allocated additional resources to the virtual machine running Elasticsearch.

Deployed monitoring tools to track resource usage.

Configured Elasticsearch to handle resource constraints more effectively.

X- Conclusion

This project successfully demonstrates the integration of various tools within the Elastic Stack to analyze network traffic captured in PCAP files. By leveraging tcpdump for packet capture, Filebeat for log shipping, Logstash for data processing, and Elasticsearch for storage and search capabilities, we have created a robust pipeline for real-time network analysis.

Throughout the project, we ensured that services were properly configured and running, allowing for seamless data indexing and retrieval. The use of Pyshark in our Python script enabled us to extract meaningful insights from the captured packets, which were then indexed into Elasticsearch for easy access and analysis.

Monitoring logs and verifying service statuses were crucial steps in maintaining the integrity of the system, allowing us to troubleshoot and resolve issues effectively. The project not only highlights the power of the Elastic Stack in handling network data but also emphasizes the importance of security measures, as indicated by the warnings regarding Elasticsearch's built-in security features.

Overall, this project serves as a comprehensive framework for network traffic analysis, providing valuable insights into network behavior and potential security threats. It lays the groundwork for further enhancements, such as implementing security features, expanding data visualization capabilities in Kibana, and integrating additional data sources for a more holistic view of network activity.