Project Manipulating the Elastic Stack

01/06/2021

Vincent Partimbene

Mastère Spécialisé Big Data

INF726 - Sécurité pour le Big Data

Supervisor

Julien Dreano

Contents

1 Introduction

1.1 Context & Objectives

This project is part of an introductory cybersecurity course, the main objective of this work is to deploy the Elastic Stack in order to monitor and analyze network traffic. This report consists of four parts:

- in chapter 1, the Elastic Stack, its components and how it fits into a pipeline of network packet analysis are briefly presented;
- in chapter 2, the architecture developed to meet the objectives is described, its implementation and the technologies used are detailed;
- in chapter 3, the workflow is detailed, from data acquisition to data analysis, including pre-processing;
- in chapter 4, the dashboard developed to monitor and analyse network traffic is presented.

This project is available on GitHub at `https://github.com/vpartimb/docker_elastic_stack_wireshark`.

1.2 The Elastic Stack

The ELK Stack is an open source suite consisting of three main components: Elasticsearch, Logstash and Kibana [1]. Beats was then added to form the Elastic Stack. Figure 1.1 represents the suite and its main components, each of them have specific functions, from ingesting the data, to storing and visualizing it.

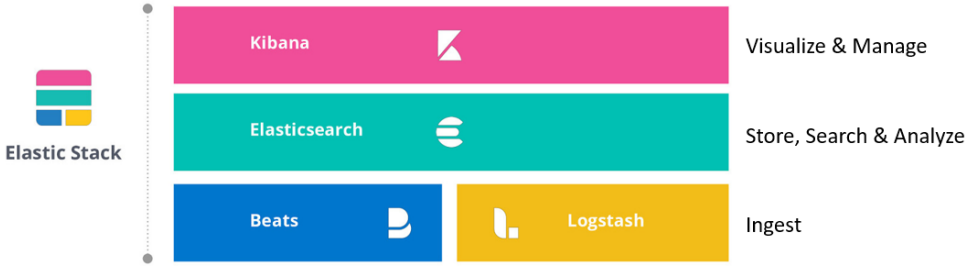

Figure 1.1: The Elastic Stack

Ingest Logstash is a server-side data processing pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to a "stash" like Elasticsearch. Beats is a collection of lightweight agents that allow the transfer of different types of logs: log files (*Filebeat*), network packets (*Packetbeat*), audit files (*Auditbeat*), service availability (*Heartbeat*), Windows logs (*Winlogbeat*), etc.

Store, search and analyze Elasticsearch is the main search and analysis engine of the Elastic Stack, it stores the data and makes them accessible. It can be used for several purposes, but its main functionality is the indexing of data streams. Elasticsearch stores data in JSON format, this data is contained in indexes which are databases. The indexes contain documents in which the data is organized in "fields". It acts as a RESTful API (Application Programming Interface).

Visualize and manage Kibana is a flexible and interactive dashboard that lets users visualize data stored in Elasticsearch with charts and graphs.

The Elastic Stack is a complete and very powerful tool, ideal for the monitoring and analysis of network packets.

1.3 Network Packet Monitoring & Analysis

Network packet capture and analysis is one of the most important capabilities for network administrators and security analysts. The ability to look into every metadata and payload that went through the network provides very useful visibility and helps to monitor systems, debug issues, and detect anomalies and attackers.

pcap (abbr. of *packet capture*) is an API for capturing network traffic [2]. It produces a `pcap` binary file containing, for each packet, many different fields that using proper tools can be parsed out as a `json` file into numbers, text, timestamps, IP addresses, etc. All this data can then be transferred and stored in Elasticsearch and explored, searched, analyzed and visualized in a Kibana dashboard.

The `pcap` file used in this project is the 25 MB file of the capture files from 4SICS Geek Lounge provided by Netresec on their website : `https://download.netresec.com/pcap/4sics-2015/4SICS-GeekLounge-151020.pcap`. The file is 858 MB once in `json` format.

1.4 Wireshark

Although *Packetbeat*, an agent primarily designed to capture live traffic, can be configured to read packets from a `pcap` capture file, Wireshark was preferred for this project.

Wireshark is the most widely used packet analyzer and its TShark network protocol analyzer can read packets from `pcap` capture files and output them as `json` files for the Elasticsearch Bulk API format [3].

2 Architecture

In order to build an architecture capable of monitoring and analyzing network packets, two options are possible: create a virtual machine and install the necessary components or use containers. The latter alternative is preferred in this project and we will be using the Docker software [4] and build a container for Wireshark and each component of the Elastic Stack.

The advantages of this approach are:

- its flexibility: each container runs a single application, containers can then be connected to each other through well-defined channels and additional containers can easily be added to the architecture;
- its limited resource usage: containers are lightweight and if no process is started in the container, then it stops;
- its ability to be used on a large number of platforms (Microsoft Windows, Linux, macOS and other Unix-like operating systems).

2.1 Overview

This architecture is heavily inspired by a blog post from Christoph Wurm on the Elastic website [5] and a GitHub repository explaining how to run the Elastic Stack with Docker [6].

Figure 2.1 represents an overview of the architecture developed for this project.

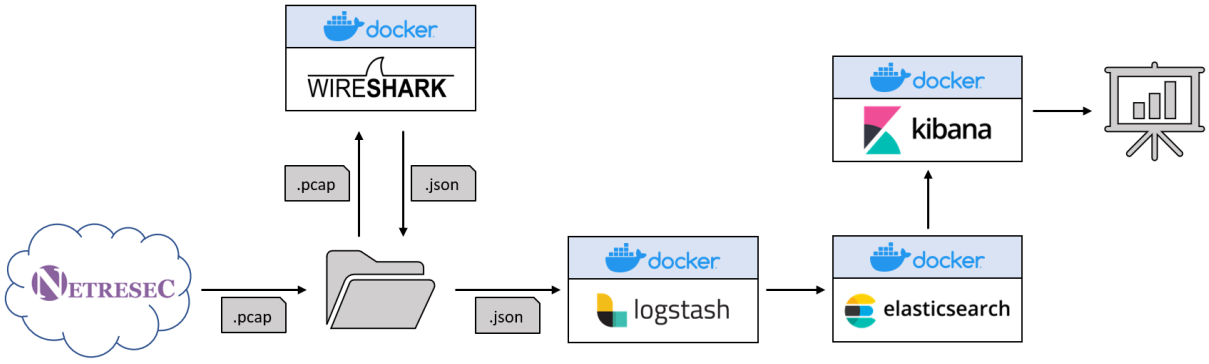

Figure 2.1: Network packet analysis pipeline with Wireshark and the Elastic Stack

This pipeline consists of 5 steps:

1. download `pcap` files to a local volume;
2. have the Wireshark container read `pcap` and output them as `json` files;
3. have the Logstash container read the `json` files, pre-process them if needed and send them to Elasticsearch;
4. have the Elasticsearch container receive the `json` files, index and store them;
5. use the Kibana container to visualize and analyze the data stored in Elasticsearch.

2.2 Docker Compose

Docker Compose is a tool for defining and running multi-container Docker applications [7]. It uses YAML (Yet Another Markup Language) files to configure the application's services and performs the creation and start-up process of all the containers with a single command.

Figure 2.2 represents the file structure of the Docker Compose folder. Each container has its own folder in which there is at least a `Dockerfile` text document that contains all the commands a user could call on the command line to assemble an image [8]. For the Elastic Stack's containers, it also contains a `yml` configuration file which specifies: server and host names, security credentials, etc. The Logstash folder also has a `conf` file that specifies the data input source, the pre-processing functions and the output to Elasticsearch. The Wireshark folder contains an additional `bash` script that converts `pcap` files to `json` files using the TShark utility.

The reader is referred to the appendices for detailed listings of the various files.

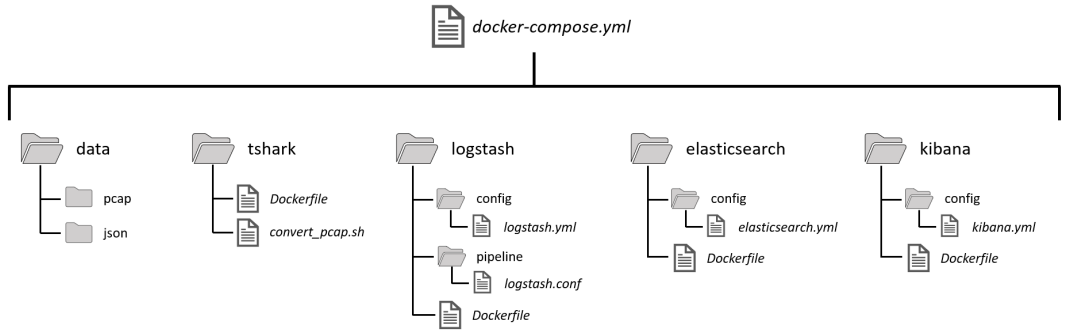

Figure 2.2: Docker Compose file structure

The `docker-compose.yml` file (see Appendix A) orchestrates the implementation of the architecture and connects the different containers to each other and to the data folder.

Note that for reliability and security considerations, a cluster of several nodes (at least 3) of Elasticsearch would have been required.

2.3 Setup & Requirements

Host requirements:

- Docker Engine version 20.10.6 or newer;
- Docker Compose version 1.29.1 or newer;
- at least 3 GB of RAM;
- Elastic Stack version used in this project: 7.13.0.

Setup:

- copy `pcap` files in the data folder;
- launch the stack using the following command:

```
$ docker-compose up
```

- wait for files to be converted and connect to Kibana at `http://localhost:5601/`
- once you're done, stop the stack using the following command:

```
$ docker-compose down -v
```

3 Workflow

Figure 3.1 represents the network packet processing workflow with extracts from Wireshark and Logstash pre-processing scripts.

The only pre-processing required was to match the timestamp with the Unix timestamp and to extract the innermost protocol.

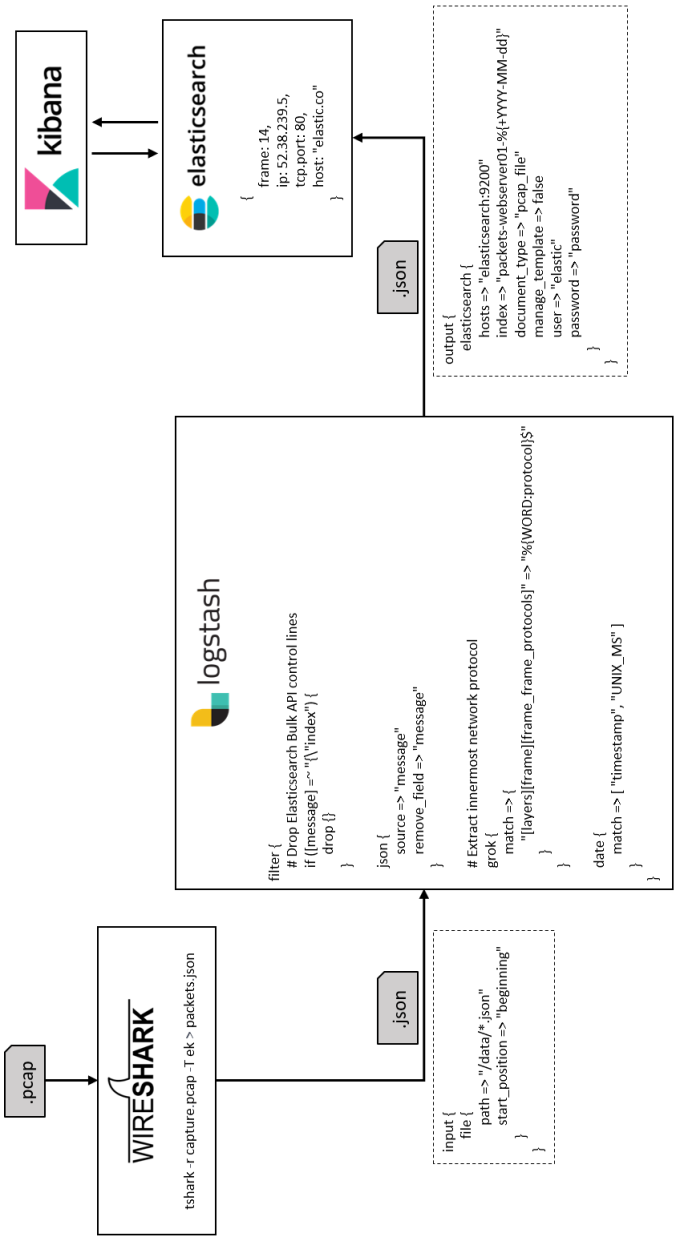

Figure 3.1: Network packet processing workflow

4 Kibana Dashboard

Figure 4.1 represents an example of a Kibana dashboard a user could make with the data used in this project.

IPv4 addresses This chart shows that more than 50 % of the traffic comes or goes to two IPv4 addresses.

Source TCP ports to destination TCP ports This chart shows the proportions of TCP ports and ports relations: source ports in the inner circle to destination ports in the outer circle.

Protocols This chart shows that most of the traffic consists of TCP protocols.

Traffic record This table shows the full record of the traffic each 5 minutes with source and destination IPs and frame lengths.

Source MAC addresses This chart shows that most of the incoming packets come from only two machines.

Destination MAC addresses This chart shows that almost half of the outgoing packets go to a single machine.

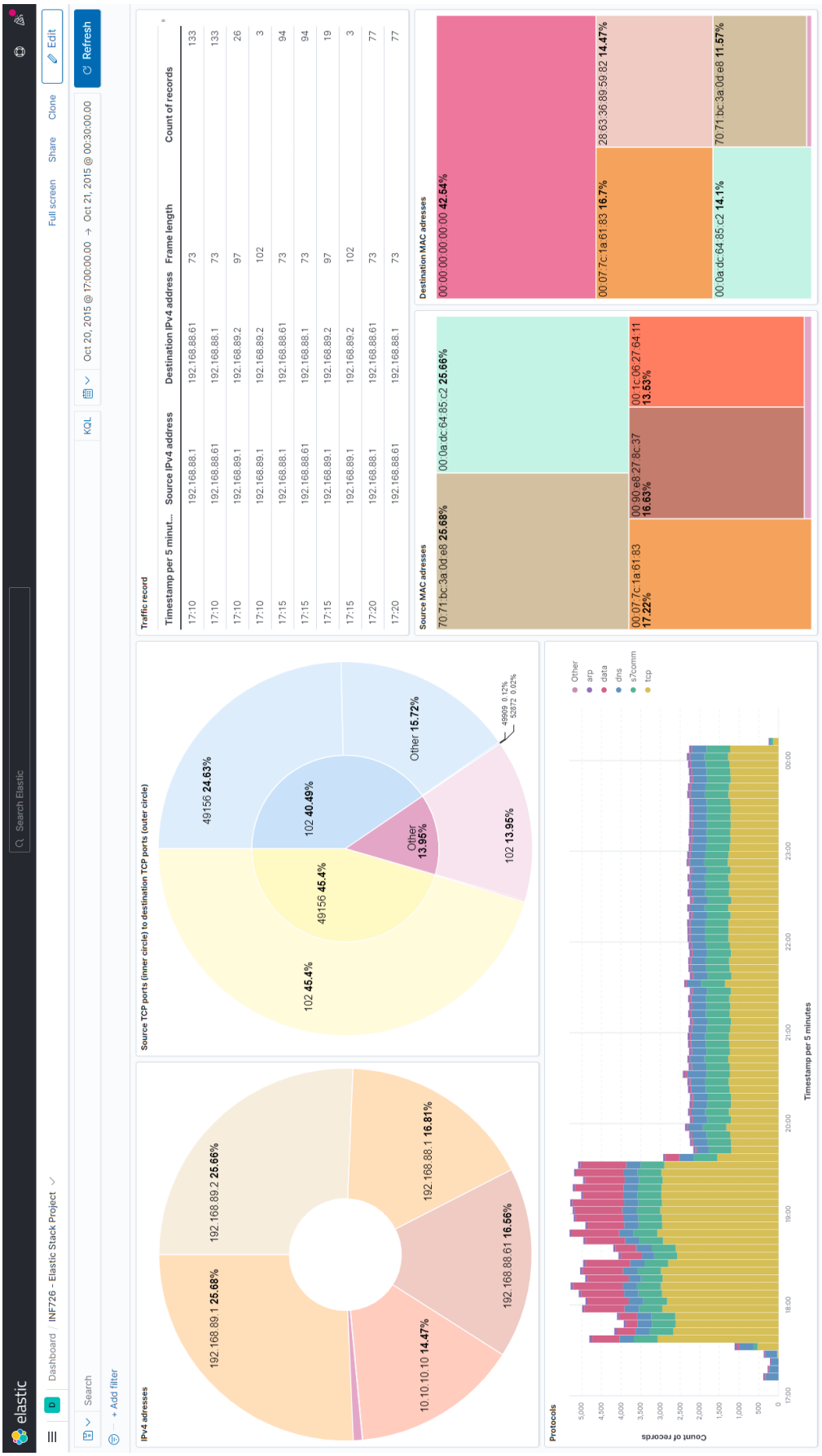

Figure 4.1: Kibana dashboard example

A Listing: `docker-compose.yml`

```yaml
version: '3.8'
services:
    tshark:
        container_name: wireshark
        image: toendeavour/tshark
        volumes:
            - type: bind
              source: ./data
              target: /data
            - ./tshark/convert_pcap.sh:/convert_pcap.sh
        entrypoint: ['/bin/sh', '/convert_pcap.sh']

    logstash:
        container_name: logstash
        build:
            context: logstash/
        volumes:
            - type: bind
              source: ./logstash/config/logstash.yml
              target: /usr/share/logstash/config/logstash.yml
            - type: bind
              source: ./data/json
              target: /data
            - type: bind
              source: ./logstash/pipeline
              target: /usr/share/logstash/pipeline
        ports:
            - "5044:5044"
            - "5000:5000/tcp"
            - "5000:5000/udp"
            - "9600:9600"
        networks:
            - elk
        depends_on:
            elasticsearch: { condition: service_healthy }

    elasticsearch:
        container_name: elasticsearch
        build:
            context: elasticsearch/
        volumes:
```

```
            - type: bind
              source: ./elasticsearch/config/elasticsearch.yml
              target: /usr/share/elasticsearch/config/elasticsearch.yml
              read_only: true
            - type: volume
              source: ./elasticsearch
              target: /usr/share/elasticsearch/data
        ports:
            - "9200:9200"
            - "9300:9300"
        environment:
            ELASTIC_PASSWORD: password
            discovery.type: single-node
        networks:
            - elk
        healthcheck:
            test: ["CMD","curl","-s","-f","http://localhost:9200/_cat/health"]

    kibana:
        container_name: kibana
        build:
            context: kibana/
        volumes:
            - type: bind
              source: ./kibana/config/kibana.yml
              target: /usr/share/kibana/config/kibana.yml
        ports:
            - "5601:5601"
        networks:
            - elk
        depends_on:
            elasticsearch: { condition: service_healthy }

networks:
    elk:
        driver: bridge

volumes:
    elasticsearch:
    data:
```

B Wireshark Container Listings

B.1 Listing: Dockerfile

```
# https://hub.docker.com/r/toendeavour/tshark:latest
FROM toendeavour/tshark:latest
```

B.2 Listing: convert_pcap.sh

```
#!/bin/bash

# Transform all .pcap files to json file using TShark
for filename in data/pcap/*.pcap; do
    name="$(echo $filename | cut -d'/' -f3 | cut -d'.' -f1)"
    /usr/bin/tshark -r $filename -T ek > /data/json/$name.json
done
```

C Logstash Container Listings

C.1 Listing: Dockerfile

```
# https://www.docker.elastic.co/
FROM docker.elastic.co/logstash/logstash:7.13.0
```

C.2 Listing: logstash.yml

```
## Default Logstash configuration from Logstash base image.
#
http.host: "0.0.0.0"
xpack.monitoring.elasticsearch.hosts: [ "http://elasticsearch:9200" ]

## X-Pack security credentials
#
xpack.monitoring.enabled: true
xpack.monitoring.elasticsearch.username: elastic
xpack.monitoring.elasticsearch.password: password
```

C.3 Listing: logstash.conf

```
input {
    file {
        path => "/data/*.json"
        start_position => "beginning"
    }
}

filter {
    # Drop Elasticsearch Bulk API control lines
    if ([message] =~ "{\"index") {
        drop {}
    }

    json {
        source => "message"
        remove_field => "message"
    }

    # Extract innermost network protocol
```

```
    grok {
        match => {
            "[layers][frame][frame_frame_protocols]" => "%{WORD:protocol}$"
        }
    }

    date {
        match => [ "timestamp", "UNIX_MS" ]
    }
}

output {
    elasticsearch {
        hosts => "elasticsearch:9200"
        index => "packets-webserver01-%{+YYYY-MM-dd}"
        document_type => "pcap_file"
        manage_template => false
        user => "elastic"
        password => "password"
    }
}
```

D Elasticsearch Container Listings

D.1 Listing: `Dockerfile`

```
# https://www.docker.elastic.co/
FROM docker.elastic.co/elasticsearch/elasticsearch:7.13.0
```

D.2 Listing: `elasticsearch.yml`

```
## Default Elasticsearch configuration from Elasticsearch base image.
#
cluster.name: "docker-cluster"
network.host: 0.0.0.0
```

E Kibana Container Listings

E.1 Listing: `Dockerfile`

```
# https://www.docker.elastic.co/
FROM docker.elastic.co/kibana/kibana:7.13.0
```

E.2 Listing: `kibana.yml`

```
## Default Kibana configuration from Kibana base image.
#
server.name: kibana
server.host: 0.0.0.0
elasticsearch.hosts: [ "http://elasticsearch:9200" ]
monitoring.ui.container.elasticsearch.enabled: true
```

References

[1] *Free and Open Search: The Creators of Elasticsearch, ELK & Kibana | Elastic*. URL: https://www.elastic.co/what-is/elk-stack.

[2] *pcap*. URL: https://fr.wikipedia.org/wiki/Pcap.

[3] *Wireshark. Go Deep*. URL: https://www.wireshark.org/.

[4] *Empowering App Development for Developers | Docker*. URL: https://www.docker.com/.

[5] Christoph Wurm. *Analyzing Network Packets with Wireshark, Elasticsearch, and Kibana*. URL: https://www.elastic.co/blog/analyzing-network-packets-with-wireshark-elasticsearch-and-kibana.

[6] *Elastic stack (ELK) on Docker*. URL: https://github.com/deviantony/docker-elk.

[7] *Overview of Docker Compose | Docker Documentation*. URL: https://docs.docker.com/compose/.

[8] *Dockerfile reference | Docker Documentation*. URL: https://docs.docker.com/engine/reference/builder/.