Projet INF726-Securite

nicolas.binuani@telecom-paris.fr

June 2021

Table des matières

1 Introduction

Le SIEM (Security Information and Event Management) est un outil technique qui va permettre de gérer les logs générés par les différents équipements / applicatifs (de sécurité ou non) qui composent un système d'information.

L'objectif d'un SIEM est de permettre aux équipes sécurité de détecter des attaques grâce à l'exploitation, le filtrage et à la corrélation de ces (millions) de logs provenant de multiples sources d'information (interne ou externe à l'organisation).

Le SIEM est avant tout un système de supervision centralisé de la sécurité. Il se compose de deux solutions qui se complètent : le SIM – Security Incident Management, qui sera focalisé sur l'analyse à-postériori, l'archivage, la conformité et le Reporting et le SEM – Security Event Management qui cherche à collecter et traiter des données en quasi temps réel.

Il semble plus pertinent de parler de démarche SIEM car l'outil n'est qu'une brique de ce système de gestion des évènements de sécurité. Cette démarche doit en effet également prendre en compte les aspects organisationnels, humains et juridiques qui se posent inévitablement lors d'un projet SIEM.

Ainsi dans le contexte du BigData et la multiplication des objets connectés, la gestion de milliards de logs applicatifs est devenu un enjeu aussi bien pour gérer le SI ainsi que dans sa protection.

Le fonctionnement d'un SIEM :

- La collecte des évènements : passive ou active avec la mise en place d'agents directement sur les différents équipements ou à distance.
- La normalisation des évènements : conservation des logs bruts pour valeur juridique puis enregistrement de ces logs dans un format interprétable.
- Le stockage et l'archivage des évènements en fonction de leur nature.
- La corrélation des informations recueillies : mise en place de règles de corrélation permettant d'alerter sur un incident en cours et permettant d'identifier les causes d'un événement a posteriori.
- Le Reporting : génération de tableaux de bord et de rapports pour avoir une visibilité sur la sécurité et la conformité du SIEM.

Ainsi pour mieux gérer la centralisation des logs sécurité, de nouveaux outils adaptés au BigData ont trouvé leur utilité comme la suite ELK pour Elastic - Logstash - Kibana.

Ce projet vise donc à proposer une architecture simple ELK afin de pouvoir lire des fichiers réseaux : pcap (Packet Capture Data).

2 Architecture

2.1 Pyshark - Wireshark

Dans un projet antérieur j'avais utilisé le logiciel Wireshark afin de vérifier que je recevais des paquets de données provenant d'un capteur. Ici, j'ai découvert la librairie python pyshark qui permet l'analyse des paquets à l'aide des wrappers tshark en python. Ainsi voici un petit exemple de code qui permet de visualiser la répartition des différents protocoles provenant d'un fichier réseau.

```
import pyshark
import collections
import matplotlib.pyplot as plt
import numpy as np
cap = pyshark.FileCapture('monfichier.pcap',only_summaries=True)
protocolList=[]
for packet in cap:
        line = str(packet)
        formattedLine= line.split(" ")
        protocolList.append(formattedLine[4])
counter = collections.Counter(protocolList)
plt.style.use('ggplot')
y_pos = np.arange(len(list(counter.keys())))
plt.bar(y_pos,list(counter.values()),align='center',alpha=0.5,color=['b','g','r','c','m'])
plt.xticks(y_pos,list(counter.keys()))
plt.ylabel("Frequency")
plt.xlabel("Protocol Name")
plt.savefig("ProtocolGraph.png")
```

Et voici le résultat que l'on peut obtenir :

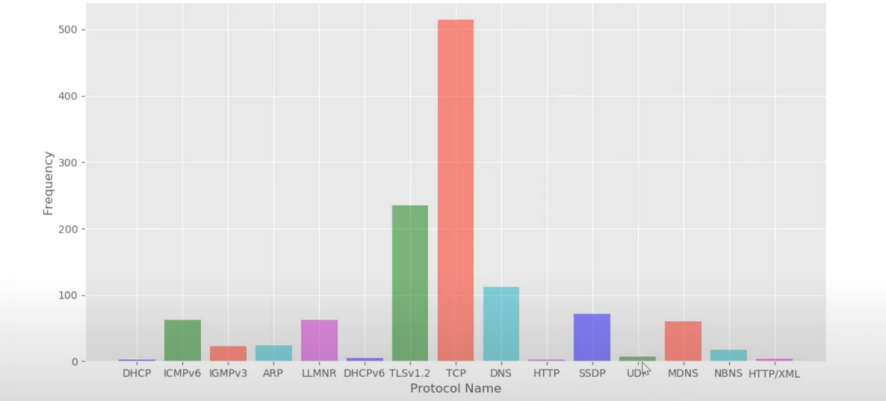

FIGURE 1 – Distribution of protocols in example pcap file

2.2 ELK - Stack

Voici comment se compose une stack ELK (Elastic - Logstash - Kibana) :

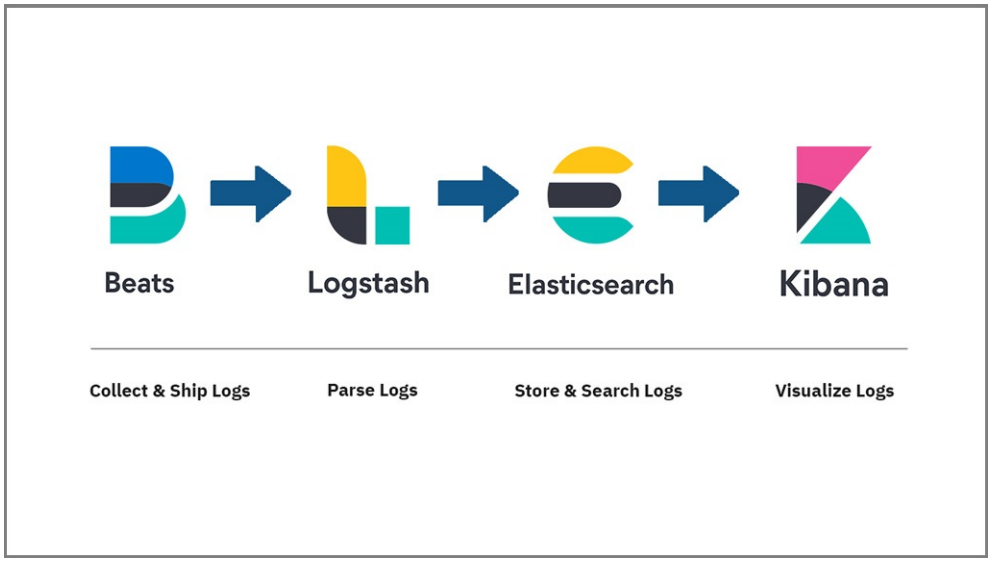

FIGURE 2 – Stack ELK

— Beats – Installe sur les machines clientes, il collecte et envoie les journaux à Logstash.
— Logstash – Traitement des logs envoyés par beats (installés sur les machines clientes).
— Elasticsearch - Stocke les journaux et les événements de Logstash et offre la possibilité de rechercher les journaux en temps réel
— Kibana – Fournit une visualisation des événements et des journaux.

Pour ce projet, j'ai tenté une première approche en voulant tout implémenter sur une VM pour pouvoir la réutiliser pour plus tard. Or j'ai eu de nombreux problèmes avec VirtualBox qui n'arrivait pas à créer de nouvelle VM. Je me suis donc rétracter sur Docker que j'avais utilisé précédemment combiné à ANsible pour déployer des images de base de données sur des clusters AWS.
J'ai créé un fichier de config YAML pour la mise en place de mon architecture que j'ai pu monitorer via Docker Desktop puis pour aller un peu plus loin j'ai aussi tenté de créer un cluster sur Elastic Cloud qui remplacera au final la création de ma VM de départ.

Au début je m'étais basé sur un cluster de 3 elastics avec un pouvoir de réplication de 2. (inspiré de nos cours de NoSQL notamment des configurations cluster de base de données il est toujours conseillé d'en avoir 3, chiffre impair à calculer en fonction du pouvoir de réplication pour définir un quorum score) En

effet comme pour les base de données on se base sur la théorie CAP : Consistency (Cohérence) - Availability (Disponibilité) - Partition tolerance (Tolérance de partition).

- Cohérence : la lecture est garantie de retourner l'écriture la plus récente pour un client donné.
- Disponibilité : le nœud non défaillant renverra une réponse raisonnable dans un délai raisonnable (pas d'erreur ou délai d'attente)
- Tolérance de partition : le système continuera à fonctionner lorsque des partitions réseau se produisent.

Finalement après plusieurs tentatives j'ai préféré aller au plus simple et créer un Elastic pour un Kibana combiné au traitement du PacketBeat et Logstash.

Voici un extrait du fichier YAML permettant de configurer l'architecture :

```yaml
version: '2.2'
services:
  es01:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.6.2
    container_name: es01
    environment:
      - node.name=es01
      - cluster.name=es-docker-cluster
      - cluster.initial_master_nodes=es01
      - bootstrap.memory_lock=true
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
    ulimits:
      memlock:
        soft: -1
        hard: -1
    volumes:
      - data_es01:/usr/share/elasticsearch/data
    ports:
      - 9200:9200
    networks:
      - elastic
    healthcheck:
      test: ["CMD", "curl","-s" ,"-f", "http://localhost:9200/_cat/health"]
      interval: 30s
      #start_period: 60s

  kibana1:
    image: docker.elastic.co/kibana/kibana:7.6.2
    container_name: kibana1
    ports:
      - 5601:5601
    environment:
      ELASTICSEARCH_URL: http://es01:9200
```

```
    ELASTICSEARCH_HOSTS: http://es01:9200
  networks:
    - elastic
  healthcheck:
    test: ["CMD", "curl", "-s", "-f", "http://localhost:5601/"]
    retries: 10

logstash:
  image: docker.elastic.co/logstash/logstash:7.6.2
  container_name: logstash
  environment:
    - ES_JAVA_OPTS='-Xms1g -Xmx1g'
  volumes:
    - logstashp:/usr/share/logstash/pipeline/
    - logstashc:/usr/share/logstash/config/
  ports:
    - 5000:5000
    - 5044-5044
  networks:
    - elastic
  restart: on-failure

pktb01:
  image: docker.elastic.co/beats/data:7.6.2
  container_name: pktb01
  user: data
  volumes:
    - pktbdata01:/usr/share/packetbeat/data
  cap_add:
    - NET_RAW
    - NET_ADMIN
  network_mode: host
  environment:
    ELASTICSEARCH_HOSTS: localhost:9200
  command: >
    packetbeat
       -e
       -strict.perms=false
       -E setup.kibana.host="localhost:5601"
       -E setup.dashboards.enabled=true
  restart: on-failure
  depends_on:
    es01:
      condition: service_healthy
    kibana1:
      condition: service_healthy
```

```
volumes:
  data_es01:
    driver: local
  pktbdata01:
    driver: local
  logstashp:
    driver: local
  logstashc:
    driver: local

networks:
  elastic:
    driver: bridge
```

Donc cet extrait du fichier de configuation comprend une partie elasticsearch dont l'environnement est inspiré de l'exemple [1] sur une jvm (java virtual machine) car ELK le processus d'indexation se base sur l'API java Lucene. Tout ce premier environnement est configuré sur le port 9200, pour le kibana on est sur le port 5601. Ensuite packetbeat on ajoute bien la command run -I avec le nom du fichier pcap ainsi que les settings pour les connexions avec kibana.

Ensuite pour charger le fichier pcap dans le conteneur docker de packetbeat, j'ai effectué une commande :

```
docker container ls
```

Puis je copie le fichier du local au conteneur :

```
sudo docker cp <Chemin du fichier pcap en local> <conteneur ID>:<Chemin data/>
```

Pour terminer on se connecte au CLI (Command Line Interpreter) du conteneur docker packetbeat puis on lance la commande :

```
./packetbeat -e -c packetbeat.yml -t -I <Chemin pcap dans le conteneur> -d "publish"
```

3 Dashboard

3.1 Prétraitement

Tout d'abord voyons un peu plus en détails le fichier pcap, il correspond au premier jour de capture du trafic réseau du laboratoire ICS (le 20/10/2015). C'est l'entreprise Netresec qui ont travaillé avec l'équipe 4SICS pour cette analyse sur des données industrielles. On a donc 5 racks en réseau et le réseau d'une compagnie suédoise Svenska kraftnät qui gère les systèmes de transmission d'énergie. Maintenant nous allons passer à une partie de pré-traitement afin de mieux le comprendre, pour cela je les d'abord lu sur Wireshark dont voici un extrait.

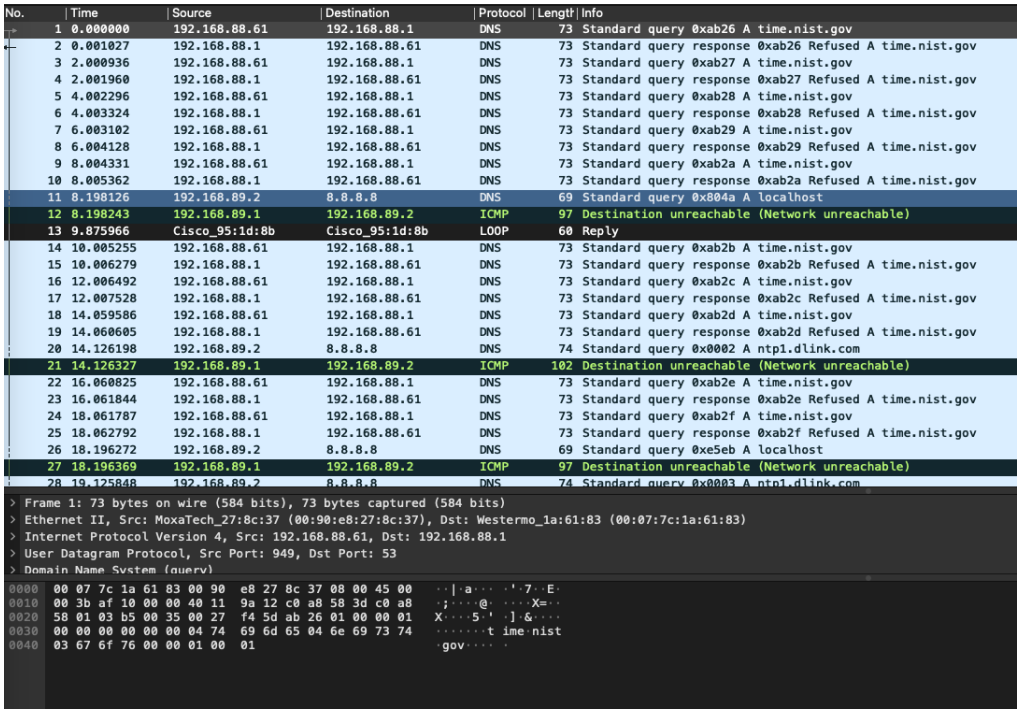

FIGURE 3 – Extrait du fichier pcap sur Wireshark

De ce que l'on voit, il est assez simple, on a 7 colonnes :

— NO qui correspond à l'identifiant du paquet analysé
— Time qui correspond au temps (dont l'échelle pourrait être prétraité pour être reformater)
— Source qui correspond à l'adresse IP de la machine qui envoie le paquet
— Destination qui correspond à l'adresse IP de la machine qui reçoit le paquet

— Protocol qui correspond au type de protocol utilisé pour le transfert du paquet
— Length qui correspond à la taille du paquet transferré sur le réseau
— Info qui correspond aux infos du paquet alors que le rest ecorrespondrait plus au header

Donc on voit bien que ce petit fichier sans trop de features (colonnes) permet de créer une première visualisation simple mais qui nous donne des statistiques intéressantes.

3.2 Dashboard & Analyse

Suite à cette étape, en récupérant le fichier de mon conteneur ELK-Securite, je peux définir comme index "No" puis créer un dashboard interactif dont voici un extrait.

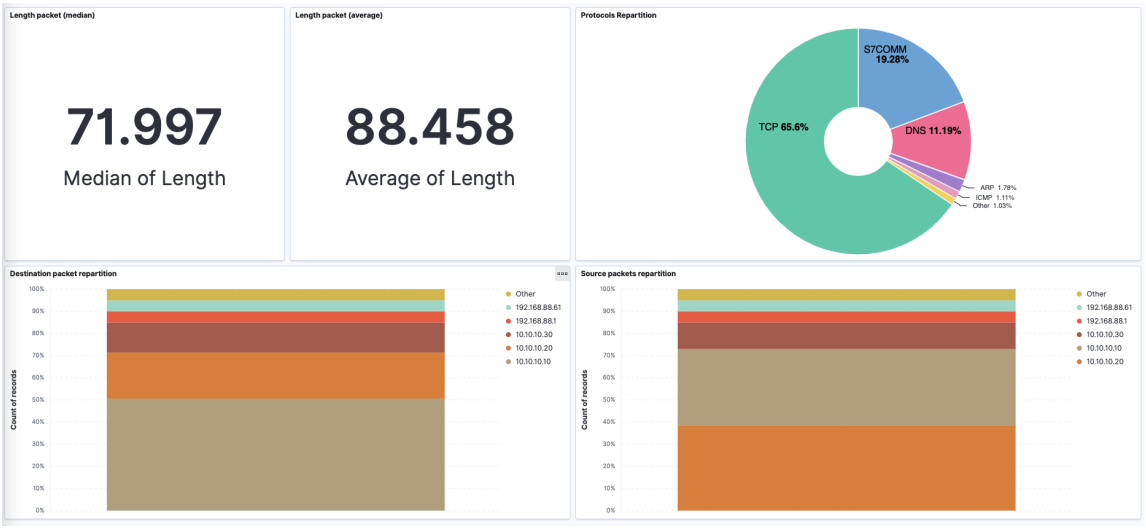

FIGURE 4 – Extrait du dashboard Kibana

Afin de le visualiser en direct lorsque celui-ci est déployé sur un cluster Elatsic Cloud, ce qui peut être très intéressant lorsqu'on veut déployer une solution de monitoring en stream processing.
On peut aussi suivre ce lien :

Lien pour visualiser le dashboard Kibana :https://observability-deployment-bfccd3.kb.us-west1.gcp.cloud.es.io:9243/app/dashboards#/view/ed0cd490-d344-11eb-805a-cd4926de5481?embed=true&_g=(filters%3A!()%2CrefreshInterval%3A(pause%3A!t%2Cvalue%3A0)%2Ctime%3A(from%3Anow-30m%2Cto%3Anow))

Pour la partie analyse du dashboard, on remarque que la plupart des échanges se font en TCP. Ensuite on remarque sur le réseau de l'entreprise suédoise c'est-à-dire les machines dont l'IP commence par 10.10.10.XX on a un objet inconnu le 20 qui récupère de nombreux paquets du Siemens SIMATIC S7 (PLC) ce qui peut laisser supposer la présence d'une clé USB ou autre non identifié dans le réseau qui récupère des données de production. On peut aussi ajouter que la taille des paquets n'est pas très importante non plus et donc qu'il faudrait "sniffer" le réseau assez longtemps pour avoir des données exploitables. D'ailleurs sur l'extrait Wireshark, on voit bien que les infos sont souvent courtes.

4 Conclusion

De part ma petite expérience précédente comme développeur dans une équipe Middleware, j'ai pu être sensibiliser à la sécurité des SI ainsi que le déploiement des systèmes applicatifs.

J'avais ainsi donc pu m'initier aux outils de CI (Continous Integration), mais je n'avais pas eu l'occasion de participer à un projet de centralisation de logs sur ELK. Ainsi ce projet m'a permis de réutiliser Docker que l'on pourrait déployer via Ansible pour déployer la stack ELK. J'ai donc pu tester cette stack très utilisé dans l'écosystème BigData IT.

De plus, j'ai pu aussi tester la librairie pyshark pour créer des scripts python qui combine les wrappers du logiciel Wireshark afin de parser des fichiers log ou réseaux. J'ai aussi essayé de réutiliser le résultat de ma stack sur docker en la déplyant sur Elastic Cloud ce qui est souvent très utilisé en entreprise (même si le CLoud pour des données de Sécurité doit être assez rare)

Pour conclure ce projet permet d'avoir une approche théorique et opérationnelle d'une architecture de traitement de logs et donc une sensibilisation à la sécurité des systèmes d'information.

5 Bibliographie

1 Running the Elastic Stack on Docker - https ://www.elastic.co/guide/en/elastic-stack-get-started/current/get-started-docker.htmlrc G. Bellemare and Joelle Pineau (2018))

2 Run PacketBeat on Docker - https ://www.elastic.co/guide/en/beats/packetbeat/current/running-on-docker.html

3 Docker for beginners - https ://towardsdatascience.com/docker-for-absolute-beginners-what-is-docker-and-how-to-use-it-examples-3d3b11efd830