Master Spécialisé

Big Data

COMPTE RENDU DE PROJET

– BGD707 –
SECURITE POUR LE BIG DATA

REALISÉ PAR :

Mathieu SAUVEUR

INTRODUCTION

Dans le cadre du cours « Sécurité pour le Big Data », nous avons pu voir les enjeux et l'importance de la sécurité au niveau des flux de données et des infrastructures de traitement. Cela inclut la mise en œuvre de mesures de protection des données contre les accès non autorisés, les atteintes à la confidentialité, l'intégrité des données et la disponibilité des systèmes.

Afin d'avoir un exemple un peu plus concret d'applications et de méthodes pouvant être employées pour assurer la sécurité des flux, il nous est proposé de déployer et d'utiliser la stack ELK et de l'utiliser afin d'observer des paquets réseau pouvant permettre de détecter des anomalies et potentiellement des attaques au niveau des flux.

Dans ce document, nous évoquerons tout d'abord comment a été réalisée l'installation et la configuration des composants de la stack ELK. Puis dans un second temps, nous verrons comment nous l'avons utilisée dans notre cas d'application. Et enfin, nous voir comment nous avons pu exploiter les données sur Kibana et expliquer le dashboard créé.

PARTIE 1 : INSTALLATION ET CONFIGURATION DE LA STACK ELK/PACKETBEAT

1) CONTEXTE

La stack ELK que nous voulons mettre en place est une suite de 3 logiciels composé de 3 produits open source que sont Elasticsearch, Logstash et Kibana. On utilise généralement cette stack pour la recherche, l'analyse et la visualisation de données en temps réel. Chaque composant de la stack a son rôle à jouer dans le processus de manipulation des données :

- Elasticsearch :

C'est un moteur de recherche et d'analyse distribué basé sur Lucene qui permet de stocker, rechercher et analyser de grandes quantités de données rapidement et en temps réel. Elasticsearch est utilisé comme base de données de recherche qui stocke les données.

- Logstash :

C'est un outil de traitement de données qui ingère des données de diverses sources, les transforme selon les configurations définies et les expédie à un "stockage" comme Elasticsearch. Logstash peut unifier des données de sources disparates et les normaliser dans le format désiré.

- Kibana :

C'est une interface utilisateur web pour visualiser les données stockées dans Elasticsearch. Kibana permet aux utilisateurs de créer des tableaux de bord dynamiques qui peuvent afficher des modifications en temps réel des données indexées par Elasticsearch.

Les différentes applications dans lesquelles la stack ELK peut intervenir sont par exemple :

- La surveillance des logs (log monitoring),
- L'analyse de sécurité,
- L'intelligence opérationnelle.

Dans notre cas de figure, nous serons amenés à réaliser plutôt une analyse de sécurité

2) DEPLOIEMENT DE LA STACK ELK

Il est possible de déployer la stack ELK de plusieurs façons possible. En effet nous pouvons choisir de la déployer :

- Localement sur sa machine en téléchargeant sous forme de paquet logiciel les composants puis en les installant sur notre machine

- Via une virtualisation en isolant un système d'exploitation puis en installant la stack sur ce dernier

- Via Docker en récupérant les images des services dont nous avons besoin pour créer les conteneurs dans lesquels les programmes vont s'exécuter.

  Concernant cette méthode, nous pouvons passer soit par des images de la stack intégrale déjà préconfigurée dans une image unique que nous pouvons retrouver dans des bibliothèques, ou encore récupérer les images individuellement de chaque composant pour créer un conteneur unique par service.

Pour le déploiement d'ELK, j'ai opté pour la dernière méthode citée ci-avant en passant par des conteneurs Docker.

Par ailleurs, il est à noter que la configuration par défaut d'Elasticsearch fait appel à l'extension X-Pack Security qui fournit une couche de sécurité pour protéger les données du cluster. Les principales fonctionnalités sécuritaires rendues possible par cette extension sont, par exemple, la gestion des authentifications, des autorisations, du chiffrement.

Conserver cette sécurité modifie notre configuration car lors des interactions entre les composants d'ELK, il nous faudra spécifier les types et caractères de connexions avec les paires identifiants/mots de passe des sessions ainsi que les certificats d'authentifications nécessaires.

Dans un premier temps, j'ai conservé les sécurités, puis dans un second temps, je les ai désactivées.

CONFIGURATION AVEC LES SECURITES

Je me suis servi d'un fichier docker-compose.yml afin de déployer tous les conteneurs en même temps pour des raisons pratiques. Ce fichier s'organise de la manière suivante :

Grâce au mot-clé « service », je spécifie que tous ce qui directement mentionné au niveau du dessous sont des entités que je veux faire tourner dans un conteneur spécifique.

1) Elasticsearch

Je commence donc d'abord par instancier le premier élément de ma stack qui est Elasticsearch.

```yaml
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:8.12.1
    container_name: elasticsearch_sec
    networks:
      - elastic
    ports:
      - "9200:9200"
    environment:
      - discovery.type=single-node
      - ELASTIC_PASSWORD=elastic
      # - xpack.security.enabled=false
      # - xpack.security.http.ssl.enabled=false
```

Je spécifie en dessous l'image à partir de laquelle je veux créer mon conteneur. En l'occurrence, je récupère ici l'image officielle de la dernière version d'Elasticsearch fournie par *elastic*. Il s'agit de la version 8.12.1, c'est également la version que j'ai utilisé pour tous les composants.

Je spécifie ensuite le nom que je veux attribuer à mon conteneur, ici elasticsearch_sec.

Sous le mot-clé « networks », je spécifie vouloir exécuter le conteneur dans un réseau interne à docker. C'est-à-dire que les connexions de la machine hôte vers le conteneur ne sont pas possibles sans avoir mappé les ports au préalable.

C'est pourquoi, la configuration ports: - "9200:9200". Comme évoqué, cela permet le mappage de ports entre le conteneur et l'hôte et donc de rendre un port spécifique à l'intérieur du conteneur accessible depuis l'hôte sur lequel Docker est en cours d'exécution.

On spécifie ensuite les variables d'environnement, pour Elasticsearch. L'utilisation de discovery.type=single-node permet de simplifier la configuration et le déploiement d'Elasticsearch, en évitant les étapes de configuration liées à la découverte et à la formation de cluster. Aussi, ELASTIC_PASSWORD=elastic permet de fixer le mot de passe pour passer outre le facteur d'authentification d'Elasticsearch et avoir accès aux différents services.

2) Logstash

Ensuite, on configure l'ouverture du conteneur de Logstash.

```
logstash:
  image: docker.elastic.co/logstash/logstash:8.12.1
  container_name: logstash_sec
  networks:
    - elastic
  volumes:
    - /usr/share/logstash/pipeline/logstash.conf/://c/Users/matsa/Documents/logstash.conf
    - /usr/share/logstash/config/logstash.yml://c/Users/matsa/Documents/logstash.yml
```

De la même manière, on donne l'image de la bonne version, le nom que nous donnons au conteneur et le réseau interne à Docker associé.

Logstash, servant de pont entre Elasticsearch et le service de collecte/envoie des paquets, nécessite une configuration spéciale. Pour le configurer, nous avons besoin de modifier le fichier logstash.conf et logstash.yml. C'est pourquoi j'ai créé une version de ces deux fichiers en concordance avec mon setup en local sur ma machine pour ensuite les transférer dans les fichiers de config du conteneur lors de sa création.

Le fichier logstash.yml sert principalement à configurer les aspects fondamentaux du fonctionnement de Logstash, tels que les paramètres de pipeline et de performance, le logging, le type d'utilisation et la gestion de plugins…

Dans mon fichier **logstash.yml**, je n'ai spécifié que les deux lignes ci-dessous :

```
1  http.host: "0.0.0.0"
2  xpack.monitoring.elasticsearch.hosts: ["https://elasticsearch:9200"]
```

Le paramètre http.host détermine sur quelle adresse IP le service Logstash écoute les connexions HTTP: Spécifier ."0.0.0.0" indique que le service doit écouter sur toutes les interfaces réseau

disponibles de la machine hôte. Cela permet aux clients de se connecter au service à partir de n'importe quelle adresse IP externe ou interne qui atteint la machine hôte, rendant le service accessible de manière plus large.

La ligne xpack.monitoring.elasticsearch.hosts: ["https://elasticsearch:9200"] spécifie l'adresse de l'hôte Elasticsearch.

Le fichier logstash.conf quant à lui spécifie comment Logstash doit recevoir, filtrer, et sortir les données. En effet, il va recevoir des données moi dois s'occuper de les prétraiter pour anvoyer à Elasticsearch un flux propre.

Dans le fichier, on spécifie d'abord la section Input

```
input {
  beats {
    port => 5044
  }
}
```

Input {} définit la source des données entrantes pour Logstash. Dans ce cas, il s'agit de données venant de Beats.

Beats {} spécifie que Logstash attend des données de Beat (dans notre cas Packetbeat), qui sont des agents légers d'envoi de logs et de métriques.

port => 5044 spécifie le port sur lequel Logstash écoute pour les connexions Beats. 5044 est le port par défaut pour la communication entre Beats et Logstash

Ensuite, vient la section pour configurer les paramètres de filtrage avec le mot clé filter {}. Cependant, j'ai décidé de ne pas en mettre car dans notre cas d'application, nous n'en avons pas spécialement besoin.

Pour finir avec le fichier logstash.conf, on configure la section dédiée à la sortie des données.

```
output {
  elasticsearch {
    hosts => "https://elasticsearch:9200"
    index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
    # user et password pour la sécurité si Elasticsearch est sécurisé
    user => "elastic"
    password => "elastic"
    ssl => true
    ssl_certificate_verification => false
    cacert => "/usr/share/logstash/config/elasticsearch-ca.crt"
  }
}
```

Ici, on va bien mentionner que nous voulons envoyer nos données post-filtration à elasticsearch pour qui on va préciser l'adresse.

La ligne index établie le e nom de l'index dans lequel les données seront stockées. Il utilise des champs de métadonnées (beat et version) pour construire dynamiquement le nom de l'index, ainsi qu'une date pour la rotation des indices.

On donne également à Logstash le nécessaire (id et mot de passe) pour passer l'authentification et accéder au host. On lui donne également le certificat de sécurité propre à Elasticsearch pour passer la vérification.

Cependant, pour faciliter la démarche, j'ai pris la décision de désactiver le facteur de vérification par certificat (ssl_certificate_verification = false).

3) Kibana

Ensuite on édite le docker-compose.yml pour lancer kibana :

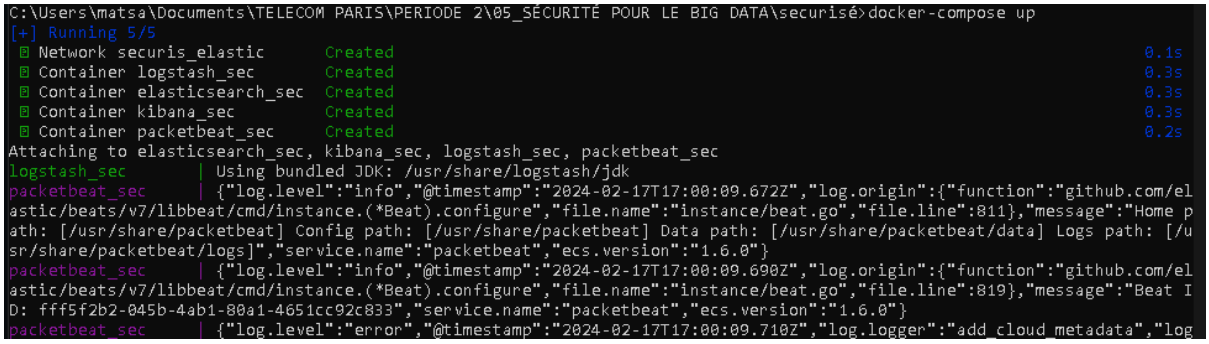
```
kibana:
  image: docker.elastic.co/kibana/kibana:8.12.1
  container_name: kibana_sec
  networks:
    - elastic
  ports:
    - "5601:5601"
  environment:
    - ELASTICSEARCH_URL=https://elasticsearch:9200
    - ELASTICSEARCH_HOSTS=https://elasticsearch:9200
```

La spécificité est qu'on mentionne en variables d'environnement les url d'accès à Elasticsearch.

À la fin du document on spécifie le type du driver du réseau interne docker « elastic » crée

```
networks:
  elastic:
    driver: bridge
```

Une fois le docker-compose établi, il nous suffit de faire la commade « docker-compose up » dans le dossier dans lequel se trouve le fichier yml et les conteneurs se lance. Si les images n'ont pas déjà été récupérées dans docker grâce à une commande « pull », elles sont automatiquement téléchargées.



Une fois les conteneurs lancés, les logs associés s'affichent dans le terminal.

3) CONFIGURATION DE PACKETBEAT

Packetbeat, issu de la collection Beats d'Elastic, est élaboré pour l'observation et l'exploration en direct du trafic réseau. Il fonctionne en saisissant les paquets circulant sur les interfaces réseau de la machine où il est déployé, pour ensuite générer une vue logique des échanges de données, représentée par des séquences de transactions entre divers services. Il est également possible grâce à packetbeats de lire des anciennes captures réseau contenus dans des fichiers pcap.

Il existe d'autres services permettant de faire la même chose comme que nous aurions pu utiliser comme Filebeat ou Metricbeat.

De la même manière, nous pouvons choisir de déployer Packetbeats soit en local soit grâce à docker. J'ai choisi de le déployer par docker en ajoutant une partie lui étant dédiée dans le fichier docker-compose afin qu'il puisse tourner dans le réseau interne docker bien que cela ne soit pas spécialement nécessaire.

```yaml
packetbeat:
  image: docker.elastic.co/beats/packetbeat:8.12.1
  container_name: packetbeat
  networks:
    - elastic
  user: root
  cap_add:
    - NET_RAW
    - NET_ADMIN
    - SYS_PTRACE
  volumes:
    - /c/Users/matsa/Documents/packetbeat.yml:/usr/share/packetbeat/packetbeat.yml
    - /c/Users/matsa/Documents/4SICS-GeekLounge-151022.pcap:/usr/share/packetbeat/4SICS-GeekLounge-151022.pcap
  depends_on:
    - elasticsearch
  environment:
    - strict.perms=false
    - ELASTIC_HOSTS=https://elasticsearch:9200
    - KIBANA_HOSTS=kibana:5601
    - LOGSTASH_HOSTS=logstash:5044
  command: ["/usr/share/packetbeat/packetbeat", "-I", "/usr/share/packetbeat/4SICS-GeekLounge-151022.pcap"]
```

Ici, j'ai bien spécifier les fichiers à volumes à monter dans le conteneur. Les fichiers en question sont le fichier packetbeat.yml ainsi que le fichier pcap que je souhaite lire qui contient les captures réseau que je souhaite envoyer à elasticsearch.

La commande qui finit la section permet de lancer l'exécution de packetbeat avec le fichier pcap.

Le fichier de configuration packetbeat.yml est composé des parties suivantes :

```
packetbeat.interfaces.file: "/usr/share/packetbeat/4SICS-GeekLounge-151021.pcap"
```

On précise le chemin (dans le conteneur) du pcap que nous voulons utiliser.

```
# Set `enabled: false` or comment out all options to disable flows reporting.
packetbeat.flows:
  # Set network flow timeout. Flow is killed if no packet is received before being
  # timed out.
  timeout: 30s

  # Configure reporting period. If set to -1s, only killed flows will be reported
  period: 10s
```

La configuration packetbeat.flows définit le comportement de capture et de rapport des flux réseau dans Packetbeat, en spécifiant un délai d'expiration de 30 secondes pour les flux inactifs et un intervalle de rapport de 10 secondes pour les flux actifs ou terminés.

```
packetbeat.protocols:
  - type: icmp
    # Enable ICMPv4 and ICMPv6 monitoring. The default is true.
    enabled: true

  - type: amqp
    # Configure the ports where to listen for AMQP traffic. You can disable
    # the AMQP protocol by commenting out the list of ports.
    ports: [5672]

  - type: cassandra
    # Configure the ports where to listen for Cassandra traffic. You can disable
    # the Cassandra protocol by commenting out the list of ports.
    ports: [9042]
```

Cette configuration active la surveillance du trafic pour les protocoles spécifiques (ICMP, AMQP, Cassandra, DHCPv4, DNS,…) sur les ports qui ont été définis, permettant la capture et l'analyse détaillée des transactions et des flux de ces protocoles. Dans notre cas de figure nous n'en avons pas forcément besoin car nous voulons envoyer les données du pcap et non écouter et récupérer les paquets transitant sur ces ports.

```
setup.kibana:
  host: "kibana:5601"
```

Cette ligne spécifie l'adresse et le port de Kibana pour la connexion et la configuration automatique des dashboards de Packetbeat dans Kibana.

```
output.logstash:
  # The Logstash hosts
  enable: true
  hosts: ["logstash:5044"]
  index: "packetbeat-%{+YYYY.MM.dd}"
```

Enfin, on configure l'output, l'endroit où nous souhaitons envoyer les données. Ici, nous souhaitons les envoyer à logstash pour un éventuel traitement des données avant de passer sur Elasticsearch.

À noter que nous n'avons pas forcément besoin de passer par logstash. EN effet, nous pouvons envoyer directement les données à Elasticsearch en définissant l'output ci-après en renseignant toujours les informations d'identification :

```
output.elasticsearch:
#   # Array of hosts to connect to.
  hosts: ["https://localhost:9200"]
  ssl.verification_mode: none
  #ssl.certificate_authorities: ["C:/Program Files/Elastic/Beats/8.12.1/packetbeat/elastic_ca.crt"]

#   # Performance preset - one of "balanced", "throughput", "scale",
#   # "latency", or "custom".
  preset: balanced

#   # Protocol - either `http` (default) or `https`.
  protocol: "https"

#   # Authentication credentials - either API key or username/password.
#   # api_key: "packetbeat_test:TU1JY2xZMEJMMGg0b1hFRF9Vb1Q6Sl8zUFlkV2FSSlNoU3EwdFNzcVkxQQ=="
  username: "elastic"
  password: "elastic"
```

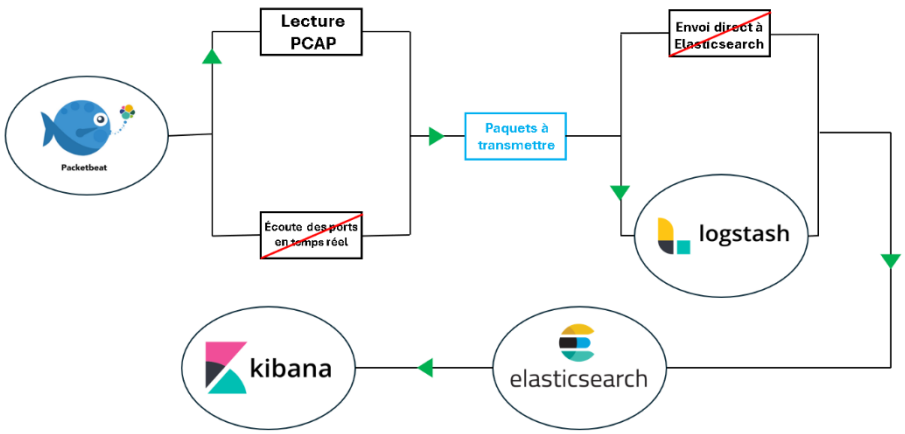
Concernant le pcap utilisé (4SICS-GeekLounge-151021.pcap), je l'ai téléchargé du site : https://www.netresec.com/?page=pcapfiles, qui proposent un certain nombre de captures réseau disponibles. Les capture que j'ai choisies proviennent de la "Geek Lounge" à la 4SICS qui « contient un laboratoire ICS avec des PLCs, des RTUs, des serveurs, des équipements de réseau industriel (commutateurs, pare-feu, etc.)

Les données contenues dans le pcap peuvent être visualisées grâce à des outils comme wireshark (qui permettent également de capturer et d'enregistrer des flux).

No.	Time	Source	Destination	Protocol	Length	Info
1	00:10:34,995270	10.10.10.20	10.10.10.10	S7COMM	153	ROSCTR:[Job] Function:[Read Var]
2	00:10:35,006630	10.10.10.10	10.10.10.20	TCP	60	102 → 49156 [ACK] Seq=1 Ack=100 Win=8192 Len=0
3	00:10:35,006630	10.10.10.10	10.10.10.20	TCP	60	[TCP Dup ACK 2#1] 102 → 49156 [ACK] Seq=1 Ack=100 Win=8192 Len=0
4	00:10:35,007135	10.10.10.10	10.10.10.20	S7COMM	104	ROSCTR:[Ack_Data] Function:[Read Var]
5	00:10:35,007135	10.10.10.10	10.10.10.20	TCP	104	[TCP Retransmission] 102 → 49156 [PSH, ACK] Seq=1 Ack=100 Win=8192 Len=50
6	00:10:35,028775	10.10.10.20	10.10.10.10	TCP	60	49156 → 102 [ACK] Seq=100 Ack=51 Win=8192 Len=0

Figure 1 Contenu de 4SICS-GeekLounge-151021.pcap

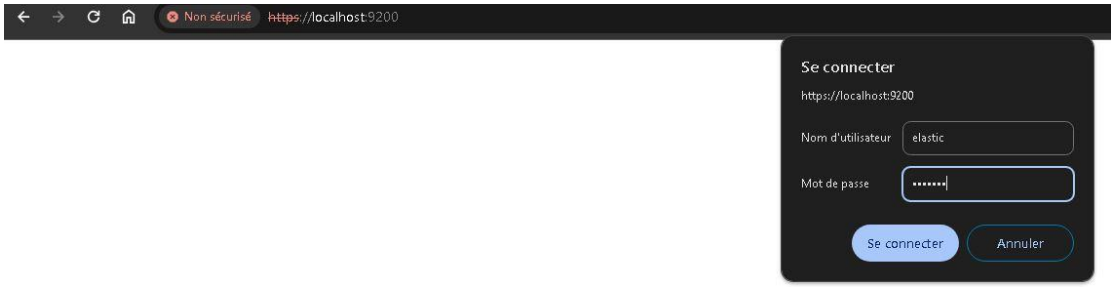
Finalement, une fois l'ensemble stask ELK + packetbeats configurés, nous pouvons représenter le schéma du workflow des traitements comme ci-dessous :



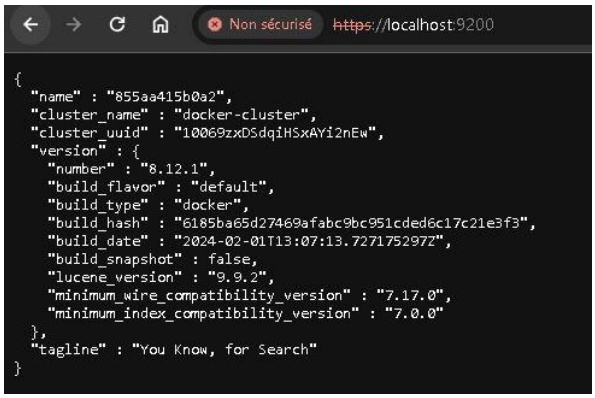
PARTIE 2 : EXPLOITATION DES DONNEES REÇUES PAR ELASTICSEARCH

1) VERIFICATION DU BON DEMARRAGE DES SERVICES

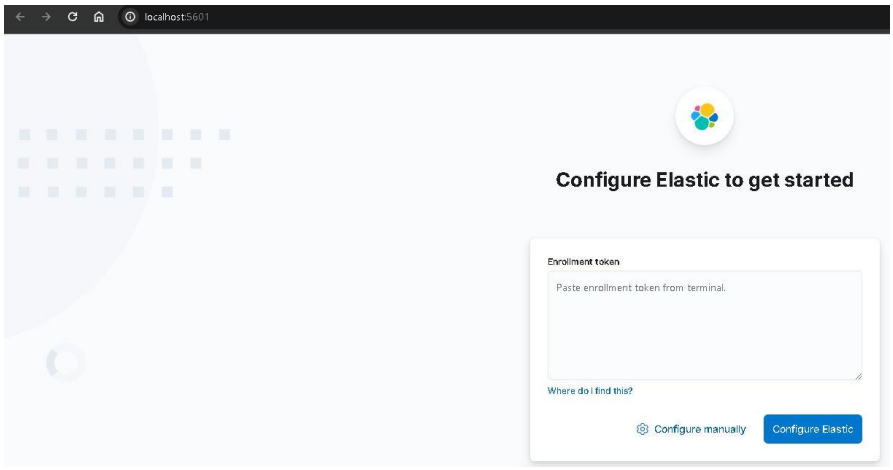
On peut vérifier d'abord si Elasticsearch a bien démarrer en vérifiant si les informations du cluster sont bien disponibles à l'adresse https://localhost:9200.



Après avoir taper ses identifiants, les informations sont affichées.



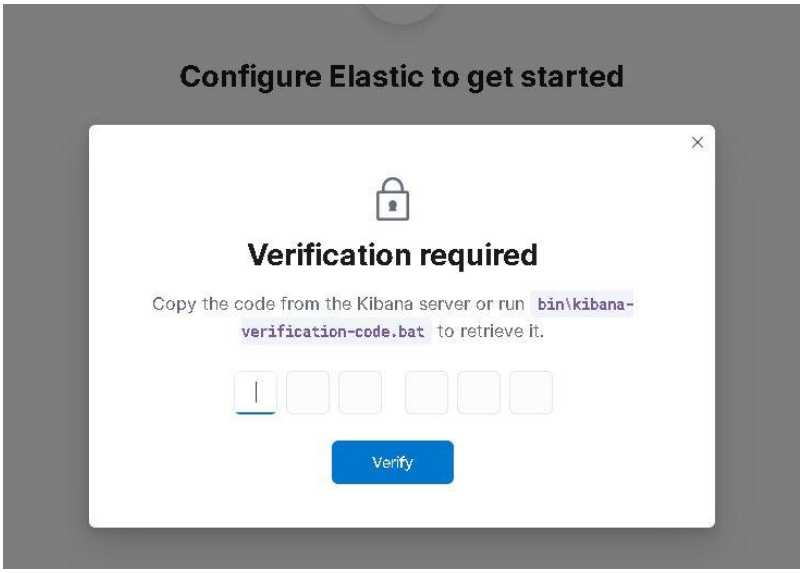
Ensuite, nous pouvons lancer Kibana à l'adresse http://localhost:5601



Il nous faut maintenant générer un enrollement token pour Kibana à partir du conteneur d'Elasticsearch avec la commande **docker exec -it elasticsearch_sec bin/elasticsearch-create-enrollment-token -s kibana.**

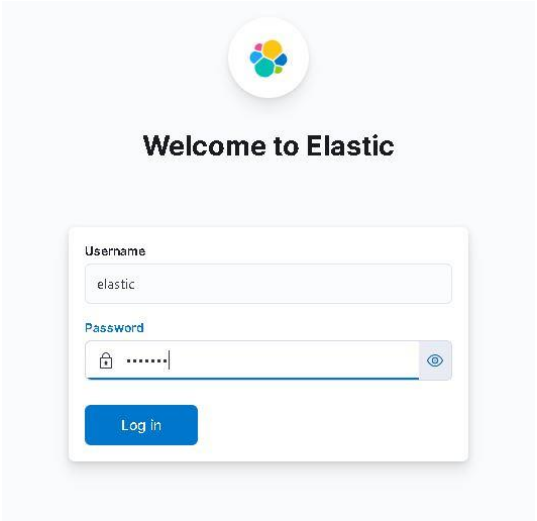
```
C:\Users\matsa>docker exec -it elasticsearch_sec bin/elasticsearch-create-enrollment-token -s kibana
WARNING: Owner of file [/usr/share/elasticsearch/config/users] used to be [root], but now is [elasticsearch]
WARNING: Owner of file [/usr/share/elasticsearch/config/users_roles] used to be [root], but now is [elasticsearch]
eyJ2ZXIiOiI4LjEyLjEiLCJhZHIiOlsiMTcyLjI0LjAuMjo5MjAwIl0sImZnciI6ImQyMTQ4ZGUyNmVlNzgwNzRmMWZiMGFhY2UwYjYyZjM2MWJjNTAwMzc3
NDQ1NDZlZGE4NmJjNDNhM2U3Yjc3MDgiLCJrZXkiOiJvc2Y3dG8wQlZVbXFLdFRNYnlwYTphd1R2Q1kwcVQzcUxTUVJ0N05VUkl3In0=
```

Une fois rentré dans kibana, il nous faut ensuite générer un code de vérification dans le conteneur de kibana grâce à la commande : **docker exec -it kibana_sec bin/kibana-verification-code**

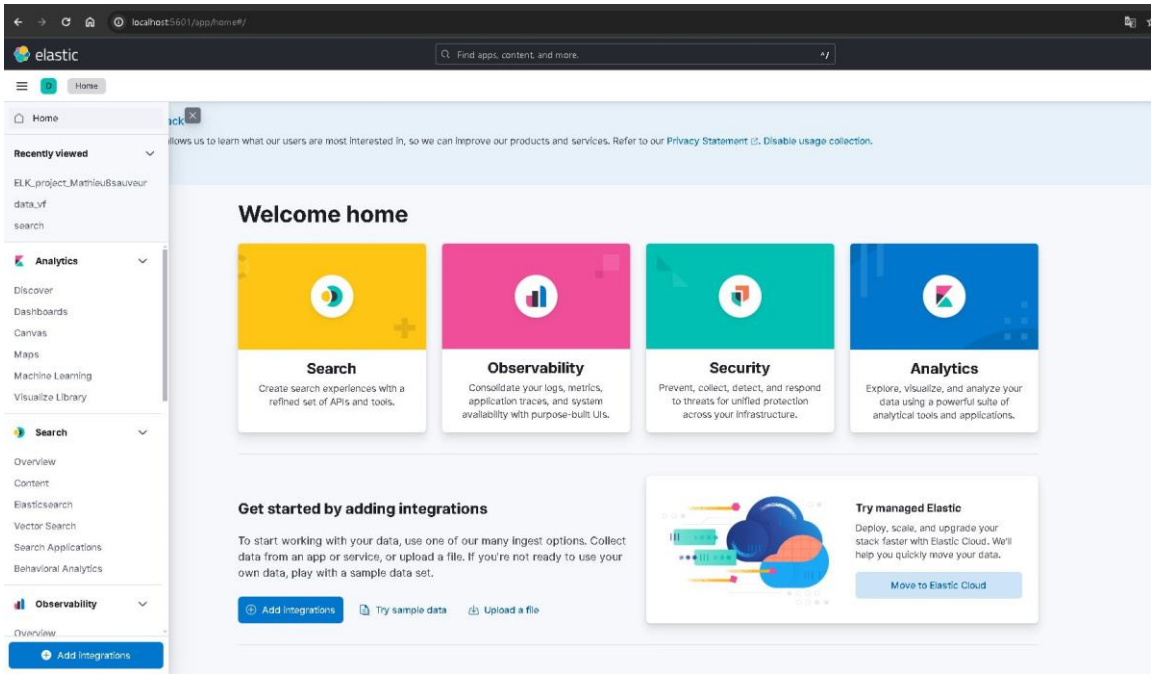



```
C:\Users\matsa>docker exec -it kibana_sec bin/kibana-verification-code
Kibana is currently running with legacy OpenSSL providers enabled! For details and instructions on how to disable see ht
tps://www.elastic.co/guide/en/kibana/8.12/production.html#openssl-legacy-provider
Your verification code is:  236 585
```

Une fois le code de vérification rentré dans Kibana, il y a une fenêtre d'authentification qui s'ouvre dans laquelle on rentre identifiant et mot de passe Elasticsearch.



Dès lors qu'on s'identifie, nous avons accès à toutes les fonctionnalités de kibana. Notre stack a donc bien été configurée.

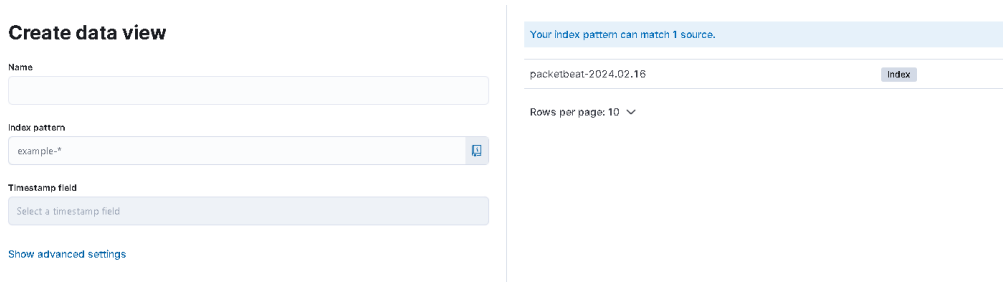


Cependant, on ne sait pas si on récupère bien les données de packetbeat passant par logstash. Pour en être sûr, on suit la méthode de la partie suivante.

2) VISUALISATION DE LA BONNE CAPTURE DES DONENES

On se place dans l'onglet Analytics de Kibana dans le menu bandeau à gauche, et plus spécialement dans la partie Discover.

Si Kibana nous propose directement de créer une Data View, c'est qu'il y a des données qui lui sont parvenues et que nous pouvons visualiser.



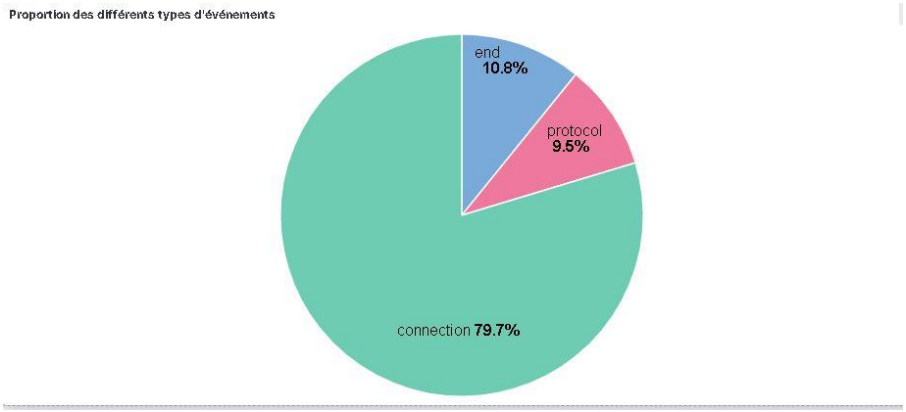
Il nous suffit juste de donner un nom à notre data view et sélectionner l'index pattern des données que nous voulons voir. Dans l'exemple ci-dessus, il n'y en a qu'un de disponible, c'est l'index spécifié dans mes configurations au début du rapport.

Depuis cette interface, on peut visualiser les données réçues par Elasticsearch (date, quantité) sous forme de hits. Nous pouvons ajuster l'intervalle de temps des données que nous souhaitons observer/analyser. En bas du screenshot, on peut avoir un aperçu des paquets réseaux.

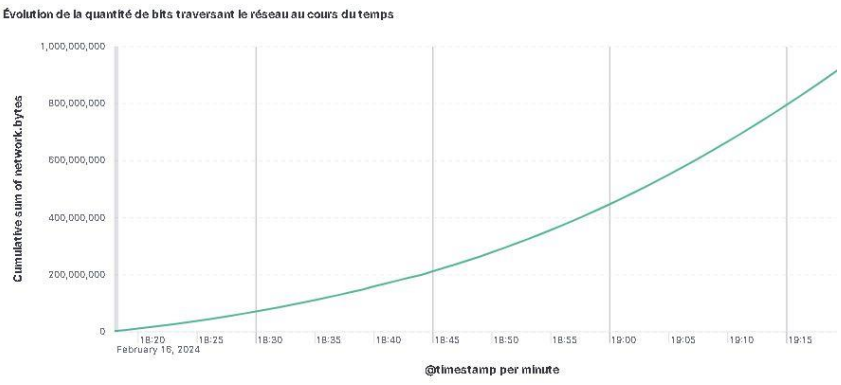
Maintenant qu'on a les données, nous pouvons réaliser des dashboards afin de les exploiter en temps réel et ainsi observer par exemple des anomalies, ou tout simplement s'informer sur la nature des paquets prélevés.

3) PRESENTATION DU DASHBOARD

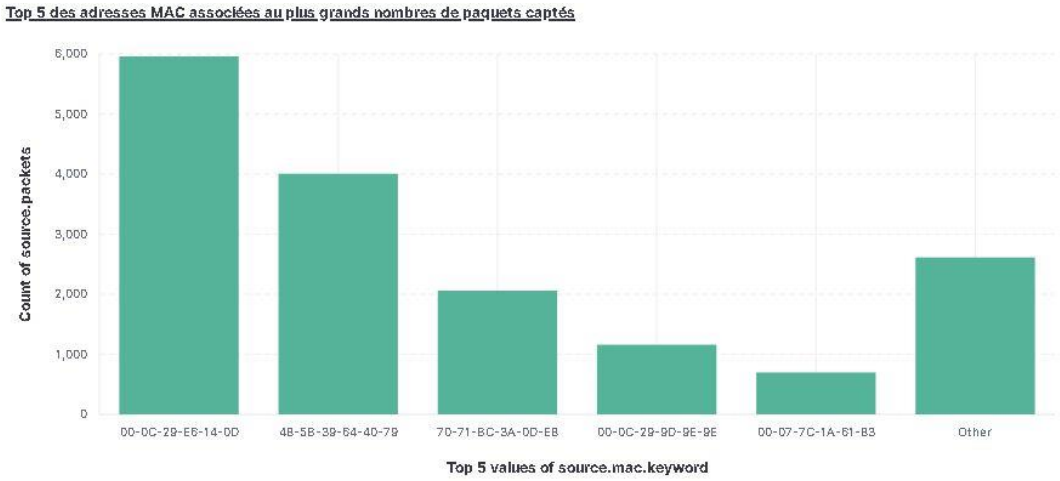
- Diagramme camembert présentant la proportion de la présence des différents types d'événements



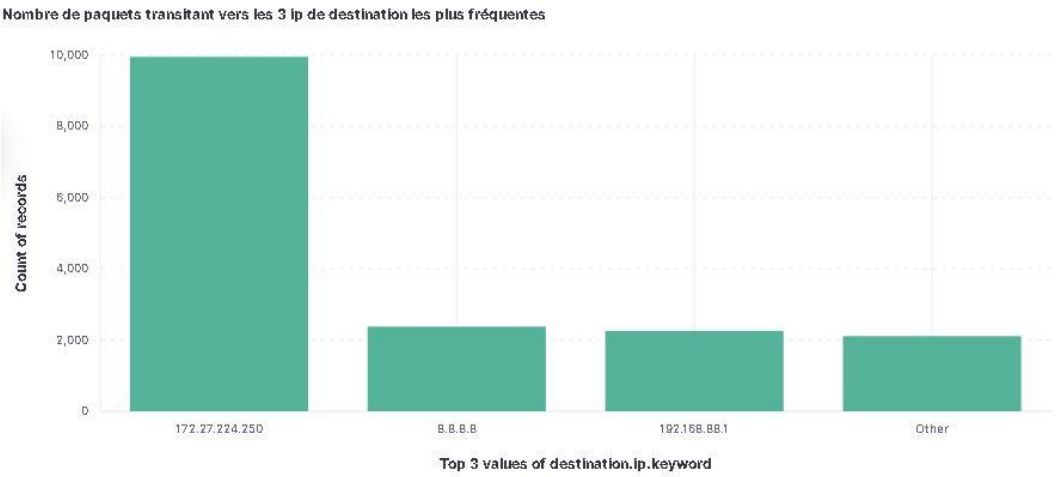
- Courbe représentant l évolution de la somme cumulative de bits traversant le réseau au cours du temps



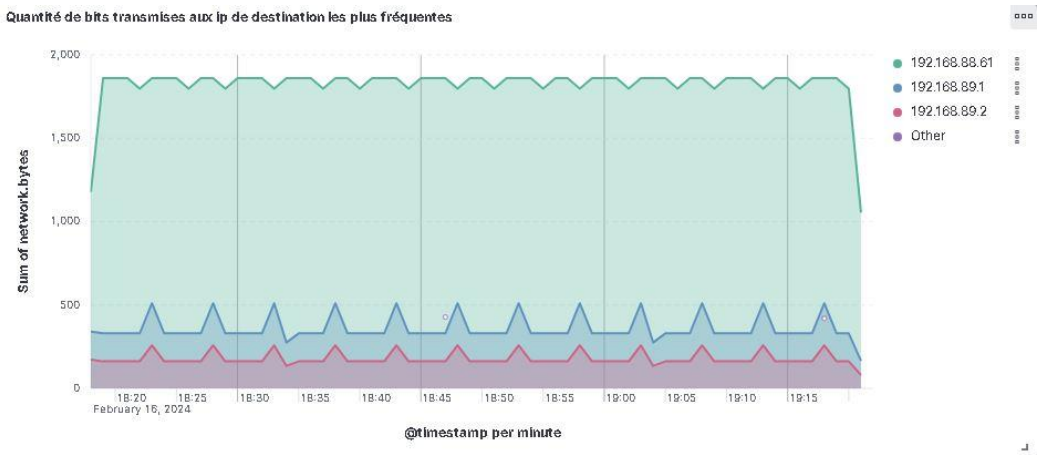
- Histogramme du top 5 des adresses MAC associées au plus grands nombres de paquets captés



- Histogramme du nombre de paquets transitant vers les 3 IPs de destination les plus fréquentes



- Courbe temporelle montrant la quantité de bits transmises aux IP de destination les plus fréquentes

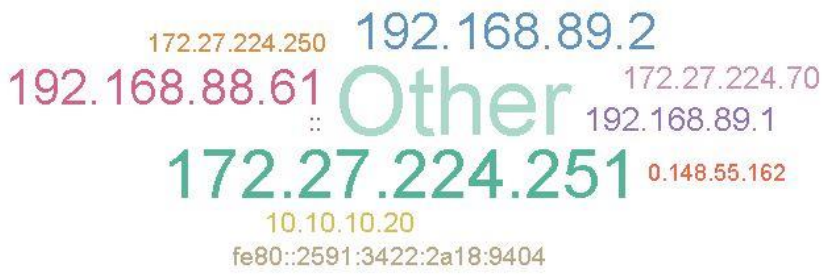


- Un aperçu et résumé des caractéristiques des derniers hits reçus par Elasticsearch (selon le filtrage temporel appliqué)

Résumé derniers hits reçus

@timestamp: Descen...	Last host.name.keyw...	Last source.ip.keywo...	Last source.port	Last destination.ip.ke...	Last destination.port	Last dns.question.na...	Last dns.question.ty...	Last dns.response_c...	Last network.commu...
Feb 18, 2024 @ 19:21:...	77c68c40a405	192.168.88.61	949	192.168.88.1	53	time.nist.gov	A	REFUSED	1:P7jlxReUPBkrfEsrEJys...
Feb 18, 2024 @ 19:21:...	77c68c40a405	192.168.88.61	949	192.168.88.1	53	time.nist.gov	A	REFUSED	1:P7jlxReUPBkrfEsrEJys...
Feb 18, 2024 @ 19:21:...	48f8b9c5a2bf	-	-	-	-	-	-	-	-
Feb 18, 2024 @ 19:21:...	77c68c40a405	-	-	-	-	-	-	-	-
Feb 18, 2024 @ 19:21:...	77c68c40a405	192.168.88.61	949	192.168.88.1	53	time.nist.gov	A	REFUSED	1:P7jlxReUPBkrfEsrEJys...

- Un diagramme nuage représentant le top 10 des IPs source les plus récurrentes dans les paquets



Top 10 values of source.ip.keyword - Count of records

- Un diagramme barre donnant le nombre de requêtes dns pour les 3 domaines de destination les plus fréquemment utilisés.