



**UNIVERISTE MOHAMMED PREMIER**  
**École Nationale des Sciences Appliquées**  
**Oujda**

جامعة محمد الأول  
المدرسة الوطنية للعلوم التطبيقية  
وجدة

## **Mémoire de Projet de Fin d'année**

**Spécialité : Sécurité Informatique & Cyber-Sécurité**

**Présenté par :**

**OUARDI Hicham & EL JALYLY Mohammed & HAKKOU Aimane**

**Sujet intitulé :**

**Construction d'un système de détection d'intrusion réseau  
(NIDS) à l'aide de Suricata, Zeek, Filebeat et ELK**

**Membres de jury :**

M. SEFRAOUI Omar

M. MADANI Mohamed Amine

**Encadrés par :**

M. SEFRAOUI Omar

**Année universitaire 2022-2023**

# Remerciements

Au terme de ce projet de fin d'année, nous tenons vivement à remercier notre encadrant Mr SEFRAOUI Omar, pour ses directives précieuses, et pour la qualité de son suivi durant le travail effectué pour ce projet. Que le corps professoral et administratif de l'ENSAO trouve ici nos vifs remerciements.

Nous souhaitons exprimer nos gratitude à nos familles pour leurs soutiens. Ainsi que nos amis et collègues qui nous ont apporté leurs supports moraux tout au long de notre travail. Pour finir, nous remercions Mr MADANI Mohamed Amine qui a accepté d'évaluer notre projet, nous leur présentons toutes nos gratitude et nos profonds respects.

# Résumé

De nos jours, les entreprises et les organisations doivent communiquer avec leurs clients via une connexion Internet, et puisque ces entreprises ont des capitaux qui peuvent être en jeu, la connexion Internet doit être sécurisée et protégée. À travers ce projet, nous allons vous montrer les étapes pour créer un système de détection sécurisé entièrement basé sur des logiciels open source comme *Suricata*, *Zeek*, *Filebeat*, *Elasticsearch*, *Logstash* et *Kibana*.

Les différents outils que nous avons installés auront pour tâche de détecter les tentatives malveillantes d'entrer dans le système implémenté. Pour l'implémentation du système de détection, on va faire deux implémentations, la première consiste à détecter les logs anormaux locaux, et la deuxième, les anomalies externes à l'aide d'une simulation.

# Table des matières

<b>Liste des figures</b> .....	6
<b>Chapitre 1 : Contexte général du projet</b> .....	7
1.1 Introduction générale.....	7
1.2 Présentation d'ENSA OUJDA.....	8
<b>Chapitre 2 : Les systèmes de détection des intrusions</b> .....	9
2.1 Définition .....	9
2.2 Les types des IDS.....	9
2.2.1 NIDS :.....	9
2.2.2 HIDS :.....	10
2.2.3 PIDS :.....	11
2.2.4 APIDS :.....	12
2.2.5 Hybrid Intrusion Detection System :.....	12
2.3 Méthodes de détection d'IDS .....	12
2.4 Différence entre IDS et IPS.....	13
<b>Chapitre 3 : Présentation des outils utilisés</b> .....	15
3.1 Suricata.....	15
3.2 Zeek .....	16
3.3 Filebeat .....	16
3.4. Elasticsearch.....	17
3.5 Logstash.....	18
3.6 Kibana.....	19
<b>Chapitre 4: Application de système de détection réseau</b> .....	21
4.1 Présentation de l'architecture.....	21
4.1.1 Les outils utilisés pour effectuer ce NIDS .....	22

4.1.2	Configuration de l'interface de sniffing .....	22
4.2	Installation et configuration de Suricata .....	24
4.2.1	Installation : .....	24
4.2.2	Configuration .....	24
4.3	Installation et configuration de Zeek .....	26
4.3.1	Installation .....	26
4.3.2	Configuration .....	27
4.4	Installation et configuration de Filebeat .....	30
4.4.1	Installation .....	30
4.4.2	Configuration .....	31
4.5	Installation et configuration d'Elasticsearch .....	32
4.5.1	Installation .....	32
4.5.2	Configuration .....	33
4.6	Installation et configuration de Logstash .....	34
4.6.1	Installation .....	34
4.6.2	Configuration .....	34
4.7	Installation et configuration de Kibana .....	36
4.7.1	Installation .....	36
4.7.2	Configuration .....	36
4.8	Test de fonctionnement .....	37
4.8.1	Implémentation 1 : Logs locaux .....	37
4.8.2	Implémentation 2 : Logs externes .....	39
<b>Conclusion générale .....</b>		<b>42</b>
<b>Webographie .....</b>		<b>44</b>

# Liste des figures

Figure 1 - NIDS  
Figure 2 - HIDS  
Figure 3 - IDS vs IPS  
Figure 4 - Suricata  
Figure 5 - Zeek  
Figure 6 - Le fonctionnement de Filebeat dans notre architecture  
Figure 7 - Le rôle d'Elasticsearch dans notre architecture  
Figure 8 - Logstash  
Figure 9 - Interface de Kibana  
Figure 10 - Le rôle de chaque élément dans ELK  
Figure 11 - Architecture de projet  
Figure 12 - Deux Adaptateurs réseau, Bridged Adapter et Host-only Adapter  
Figure 13 - Fichier de configuration des interfaces  
Figure 14 - Configuration d'interface de sniffing  
Figure 15 - Fichier de configuration de Suricata  
Figure 16 - Suricata service  
Figure 17 - Zeek bin  
Figure 18 - Zeek networks.cfg  
Figure 19 - Zeek node.cfg  
Figure 20 - Zeek zeekctl.cfg  
Figure 21 - Zeek JSON output  
Figure 22 - Zeek cronjob  
Figure 23 - Filebeat configuration  
Figure 24 - Filebeat zeek.yml  
Figure 25 - Elasticsearch service  
Figure 26 - Curl Elasticsearch server  
Figure 27 - Configuration d'Elasticsearch  
Figure 28 - Logstash service  
Figure 29 - Configuration d'index de Logstash  
Figure 30 - Kibana service  
Figure 31 - Kibana configuration  
Figure 32 - Syslog  
Figure 33 - Kibana syslog  
Figure 34 - SSH bruteforcing 1  
Figure 35 - SSH bruteforcing 2  
Figure 36 - Zeek node.cfg  
Figure 37 - Zeek networks.cfg  
Figure 38 - Suricata.yml  
Figure 39 - Suricata.yml sniffing interface  
Figure 40 - Kibana host  
Figure 41 - Elasticsearch output  
Figure 42 - Elasticsearch.yml logs path  
Figure 43 - TCP Replay  
Figure 44 - Suricata events  
Figure 45 - Suricata alertes  
Figure 46 - Network traffic direction  
Figure 47 - Zeek events  
Figure 48 - Top URL domains

# Chapitre 1 : Contexte général du projet

## 1.1 Introduction générale

Les réseaux informatiques sont devenus des ressources vitales et déterministes pour le bon fonctionnement des entreprises. De plus, ces réseaux sont ouverts de fait qu'ils sont pour la plus part raccordés à l'Internet. Cette ouverture qui permet de faciliter la communication, engendre malheureusement des risques importants dans le domaine de la sécurité informatique.

La sécurité informatique consiste d'une manière générale à assurer que les ressources matérielles ou logicielles d'une organisation soient uniquement utilisées dans le cadre du réseau informatique, plusieurs formes d'attaques ont été récemment recensées avec la prolifération de la technologie. De nombreuses entreprises sont actuellement victimes des attaques d'intrusion au niveau de leur système et de base de données. C'est un problème majeur que des intrus aient accès aux données confidentielles. Pour cela, les administrateurs déploient des solutions de sécurité efficace capable de protéger le réseau de l'entreprise. Dans le contexte, les IDS constituent une bonne alternative pour mieux protéger le réseau informatique.

Cette technologie consiste à rechercher une suite de mots ou de paramètres caractérisant une attaque dans un flux de paquets. Un IDS doit être conçu dans une politique globale de sécurité. Son objectif est de détecter toute violation liée à la politique de sécurité, il permet ainsi de signaler les attaques.

## 1.2 Présentation d'ENSA OUJDA

L'École nationale des sciences appliquées d'Oujda (ENSAO) est une grande école d'ingénieurs marocaine localisée à Oujda, généraliste dans le domaine des nouvelles technologies, informatique et télécommunications. Elle fait partie du réseau des ENSA.



### Histoire :

L'ENSAO, première école d'ingénieurs d'état dans la région de l'oriental, a été créée par le décret ministériel no 2-99-1007 du 19 septembre 2001.

L'établissement est localisé sur le campus universitaire de l'UMP. Il s'inscrit dans le cadre de la diversification des formations dispensées au sein de l'Université et de la dynamisation de l'environnement socio-économique et industriel régional et national.

La formation d'ingénieur à l'ENSAO dure 5 ans et se divise en 2 cycles :

- *Cycle préparatoire (deux ans) :*

Durant ce cycle, l'accent est mis sur les sciences fondamentales (maths, physique, chimie, mécanique et informatique) et puis progressivement, le programme d'études s'ouvre à des domaines plus spécialisés en sciences et en technologie. Ce cycle préparatoire généraliste permet aux élève ingénieurs ENSAistes d'acquérir un bagage solide en termes d'aptitudes et de compétences, leur facilitant ainsi de réorienter leur carrière indépendamment de leur spécialisation initiale.

- *Cycle ingénieur (trois ans) :*

À la fin de la deuxième année d'étude, les élèves-ingénieurs choisissent une spécialité parmi les 6 proposées par l'école, l'attribution des spécialités se fait selon le classement et le choix des étudiants.



# Chapitre 2 : Les systèmes de détection des intrusions

Dans ce chapitre on va définir c'est quoi un IDS, ses types, les méthodes de détection et son différence avec IPS.

## 2.1 Définition

Un système appelé système de détection d'intrusion (IDS) observe le trafic réseau à la recherche de transactions malveillantes et envoie des alertes immédiates lorsqu'il est observé. Il s'agit d'un logiciel qui recherche sur un réseau ou un système des activités malveillantes ou des violations de politique. Chaque activité illégale ou violation est souvent enregistrée soit de manière centralisée à l'aide d'un système SIEM, soit notifiée à une administration.

IDS surveille un réseau ou un système à la recherche d'activités malveillantes et protège un réseau informatique contre l'accès non autorisé des utilisateurs, y compris peut-être des initiés. La tâche d'apprentissage du détecteur d'intrusion consiste à construire un modèle prédictif (c'est-à-dire un classifieur) capable de faire la distinction entre les 'mauvaises connexions' (intrusion/attaques) et les 'bonnes connexions (normales)'.

## 2.2 Les types des IDS

### 2.2.1 NIDS :

Système de détection d'intrusion réseau (NIDS) : Ils sont configurés à un point planifié du réseau pour examiner le trafic de tous les périphériques du réseau. Il effectue une observation du trafic passant sur l'ensemble du sous-réseau et fait correspondre le trafic qui est transmis sur les sous-réseaux à la collection d'attaques connues. Une fois qu'une attaque est identifiée ou qu'un comportement anormal est observé, l'alerte peut être envoyée à l'administrateur.

Un exemple de NIDS est de l'installer sur le sous-réseau où se trouvent les pare-feu afin de voir si quelqu'un essaie de casser le pare-feu :

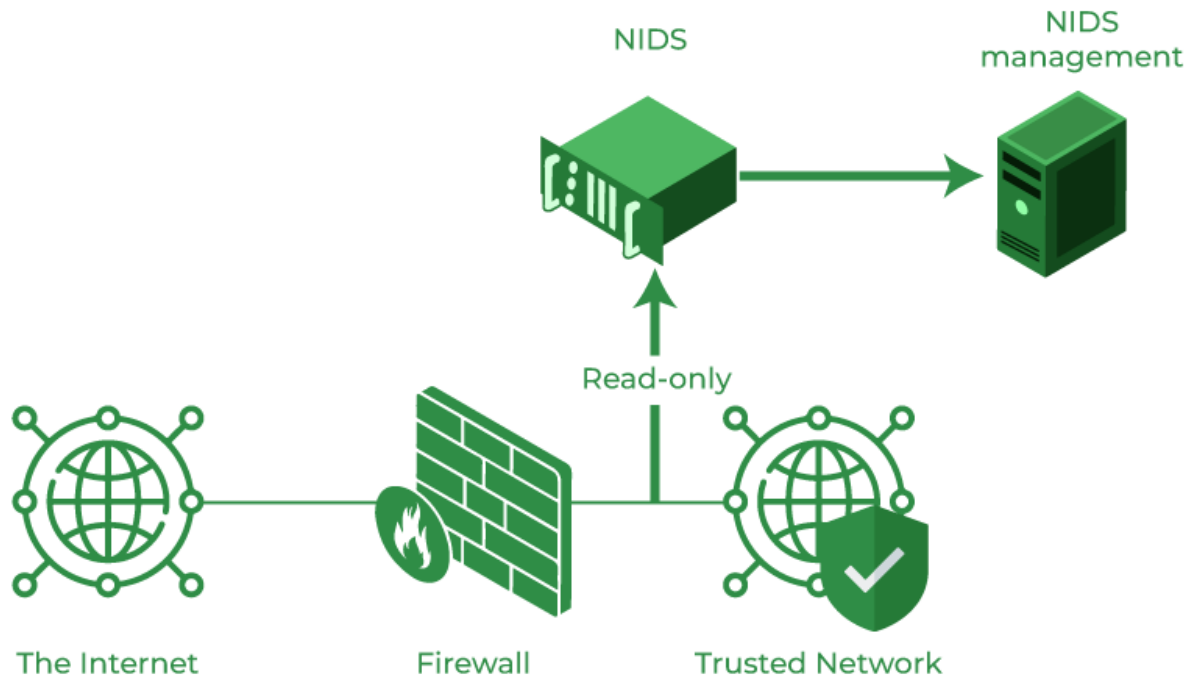


Figure 1 : NIDS

### 2.2.2 HIDS :

Système de détection des intrusions sur l'hôte (HIDS) : Ils s'exécutent sur des hôtes ou des périphériques indépendants sur le réseau. Un HIDS surveille uniquement les paquets entrants et sortants de l'appareil et alerte l'administrateur si une activité suspecte ou malveillante est détectée. Il prend un instantané des fichiers système existants et le compare avec l'instantané précédent. Si les fichiers du système d'analyse ont été modifiés ou supprimés, une alerte est envoyée à l'administrateur pour enquête.

Un exemple d'utilisation de HIDS peut être vu sur des machines critiques, qui ne devraient pas changer leur disposition :

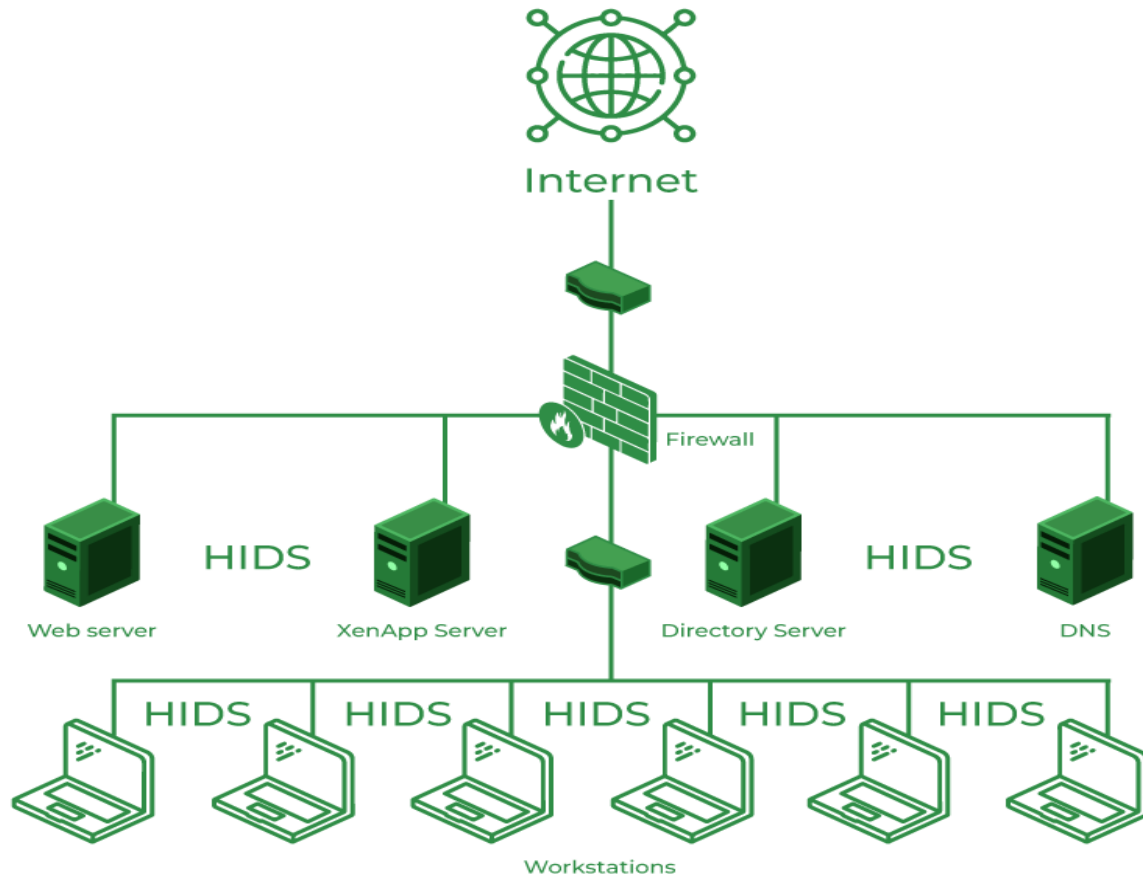


Figure 2 : HIDS

### 2.2.3 PIDS :

Système de détection d'intrusion basé sur protocole (PIDS) : Il comprend un système ou un agent qui résiderait systématiquement à l'avant d'un serveur, contrôlant et interprétant le protocole entre un utilisateur/appareil et le serveur. Il essaie de sécuriser le serveur Web en surveillant régulièrement le flux de protocole HTTPS et en acceptant le protocole HTTP associé. Comme HTTPS n'est pas crypté et avant d'entrer instantanément dans sa couche de présentation Web, ce système devrait résider dans cette interface, entre pour utiliser le HTTPS.

#### 2.2.4 APIDS :

Système de détection d'intrusion basé sur le protocole d'application (APIDS) : C'est un système ou un agent qui réside généralement dans un groupe de serveurs. Il identifie les intrusions en surveillant et en interprétant la communication sur des protocoles spécifiques à l'application. Par exemple, cela surveillerait explicitement le protocole SQL vers le middleware lors de ses transactions avec la base de données sur le serveur Web.

#### 2.2.5 Hybrid Intrusion Detection System :

Système de détection d'intrusion hybride : Il est constitué par la combinaison de deux ou plusieurs approches du système de détection d'intrusion. Dans le système de détection d'intrusion hybride, l'agent hôte ou les données du système sont combinés avec des informations de réseau pour développer une vue complète du système de réseau. Le système de détection d'intrusion hybride est plus efficace par rapport à l'autre système de détection d'intrusion. *Prelude* est un exemple d'IDS hybride.

### 2.3 Méthodes de détection d'IDS

- *Méthode basée sur la signature :*

L'IDS basé sur la signature détecte les attaques sur la base de modèles spécifiques tels que le nombre d'octets ou un nombre de 1 ou le nombre de 0 dans le trafic réseau. Il détecte également sur la base de la séquence d'instructions malveillantes déjà connue qui est utilisée par le logiciel malveillant. Les modèles détectés dans l'IDS sont appelés signatures. L'IDS basé sur les signatures peut facilement détecter les attaques dont le modèle (signature) existe déjà dans le système, mais il est assez difficile de détecter de nouvelles attaques de logiciels malveillants car leur modèle (signature) n'est pas connu.

- *Méthode basée sur les anomalies :*

L'IDS basé sur les anomalies a été introduit pour détecter les attaques de logiciels malveillants inconnus à mesure que de nouveaux logiciels malveillants se développent rapidement. Dans l'IDS basé sur les anomalies, il y a l'utilisation de l'apprentissage automatique pour créer un modèle d'activité de confiance et tout ce qui arrive est comparé à ce modèle et il est déclaré suspect s'il n'est pas trouvé dans le modèle. La méthode basée sur l'apprentissage automatique a une propriété mieux généralisée par rapport à l'IDS basé sur les signatures, car ces modèles peuvent être formés en fonction des applications et des configurations matérielles.

## **2.4 Différence entre IDS et IPS**

Un système de détection d'intrusion (IDS) est défini comme une solution qui surveille les événements du réseau et les analyse pour détecter les incidents de sécurité et les menaces imminentes.

Un système de prévention des intrusions (IPS) est défini comme une solution qui effectue la détection des intrusions, puis va plus loin et empêche toute menace détectée.

Cette figure ci-dessous répertorie les principales différences et similitudes entre IDS et IPS :

## IDS vs IPS

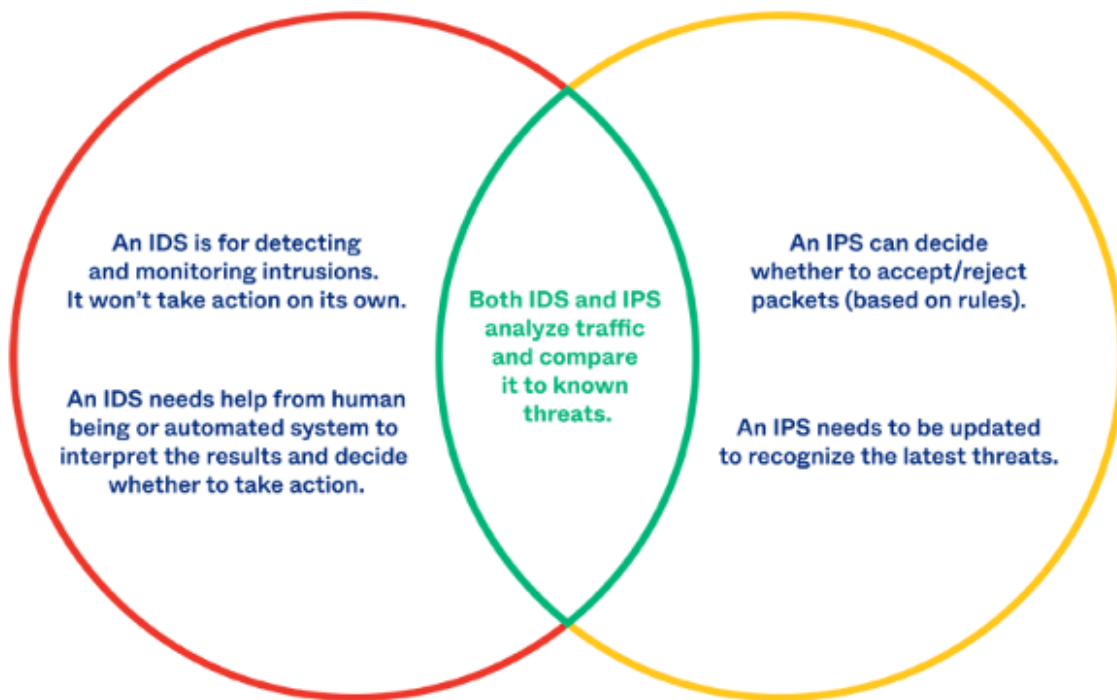


Figure 3 : IDS vs IPS

*Les IDS* : comparent l'activité réseau en cours avec une base de données d'attaques connues afin de détecter divers types de comportements tels que les violations de la politique de sécurité, les malwares et les scanners de port.

*Les IPS* : agissent dans la même zone du réseau qu'un pare-feu, entre le monde extérieur et le réseau interne. Ils rejettent de façon proactive les paquets réseau en fonction d'un profil de sécurité si ces paquets représentent une menace connue.

## Conclusion

Dans ce chapitre nous avons fait un tour des différents types d'IDS ainsi que ses caractéristiques.

# Chapitre 3 : Présentation des outils utilisés

Ce chapitre va contenir les différents outils que nous avons utilisés pour la réalisation de ce projet.

## 3.1 Suricata

Suricata est un moteur de détection open source qui peut agir comme un système de détection d'intrusion (IDS) et un système de prévention d'intrusion (IPS). Il a été développé par l'Open Information Security Foundation (OSIF) et est un outil gratuit utilisé par les entreprises, petites et grandes. Le système utilise un ensemble de règles et un langage de signature pour détecter et prévenir les menaces. Suricata peut fonctionner sous Windows, Mac, Unix et Linux.

La figure ci-dessous présente les deux modes avec lesquels Suricata pourrait fonctionner :

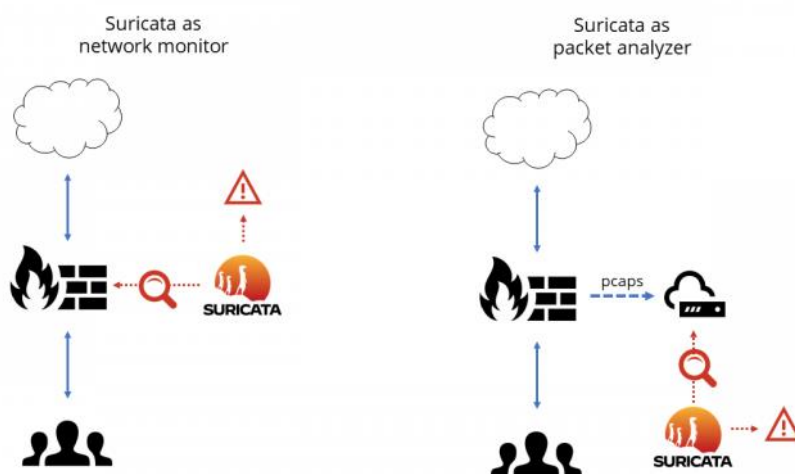


Figure 4 : Suricata

## 3.2 Zeek

Zeek (anciennement connu sous le nom de Bro) est un analyseur de trafic réseau open source. L'outil repose sur un capteur et observe le trafic réseau. Il s'agit d'un logiciel gratuit et open source conçu pour extraire des centaines de champs de données réseau en temps réel. L'outil dispose d'analyseurs préconstruits pour de nombreux protocoles tels que (HTTP, SSL, DNS, FTP, etc.) et permet la création d'analyseurs personnalisés pour les protocoles qui ne sont pas encore pris en charge. Zeek peut détecter des anomalies, mais pas de la même manière qu'un IDS traditionnel (comme Suricata).



Figure 5 : Zeek

## 3.3 Filebeat

Filebeat est un expéditeur de journaux appartenant à la famille Beats - un groupe d'expéditeurs légers installés sur des hôtes pour envoyer différents types de données dans la pile ELK à des fins d'analyse. Filebeat, comme son nom l'indique, fournit des fichiers journaux.

Dans un pipeline de journalisation basé sur ELK, Filebeat joue le rôle d'agent de journalisation, installé sur la machine générant les fichiers journaux, les suivant et transmettant les données soit à Logstash pour un traitement plus avancé, soit directement à Elasticsearch pour l'indexation.



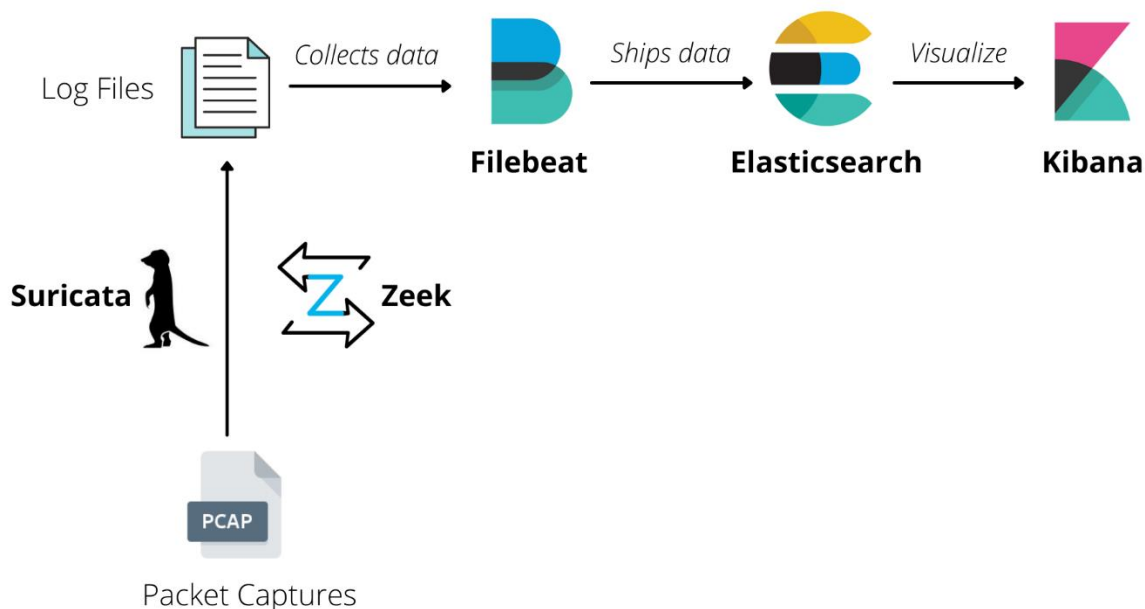


Figure 6 : Le fonctionnement de Filebeat dans notre architecture

### 3.4. Elasticsearch

Elasticsearch est un moteur de recherche et d'analyse distribué gratuit et ouvert pour toutes les données (textuelles, numériques, géo spatiales, structurées et non structurées). Elasticsearch a été conçu à partir d'Apache Lucene et a été lancé en 2010 par Elasticsearch N. V. (maintenant appelé Elastic). Réputé pour ses API REST simples, sa nature distribuée, sa vitesse et sa scalabilité, Elasticsearch est le composant principal de la Suite Elastic, un ensemble d'outils gratuits et ouverts d'ingestion de données, d'enrichissement, de stockage, d'analyse et de visualisation. Couramment appelée la Suite ELK (pour Elasticsearch, Logstash et Kibana), la Suite Elastic comprend désormais une riche collection d'agents de transfert légers, appelés les agents Beats, pour envoyer des données à Elasticsearch.

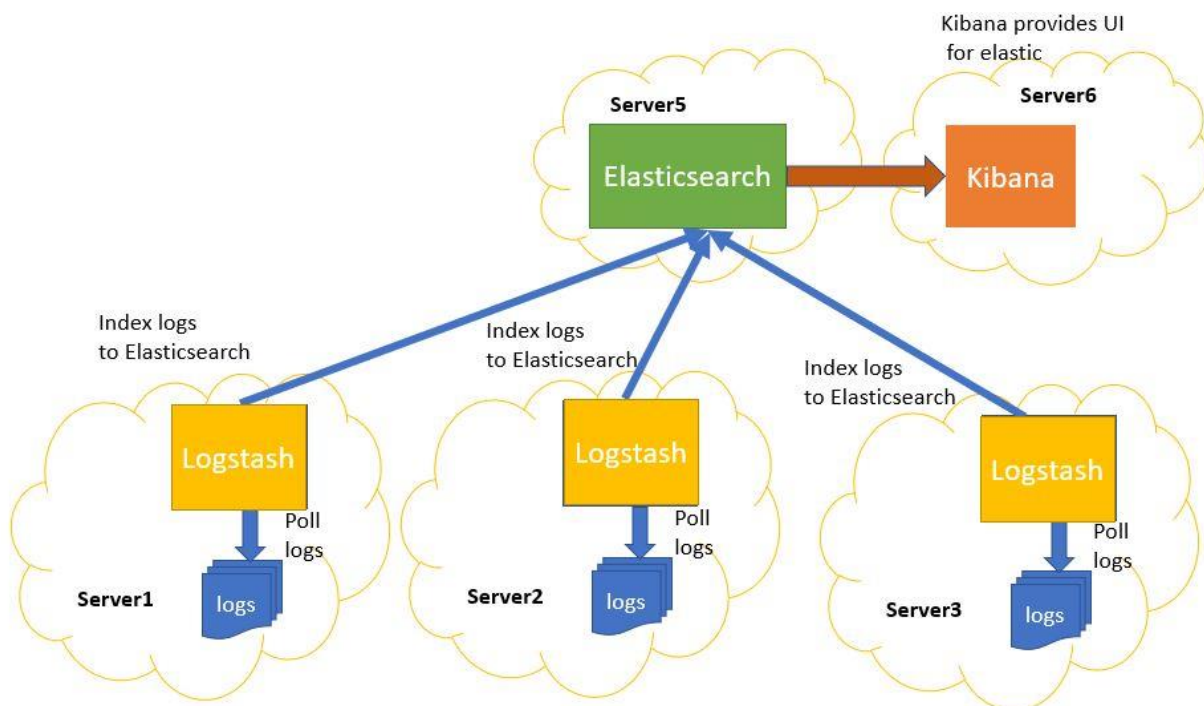


Figure 7 : Le rôle d'Elasticsearch dans notre architecture

### 3.5 Logstash

Logstash est la plate-forme d'analyse de journaux la plus populaire au monde et est responsable de l'agrégation des données provenant de différentes sources, de leur traitement et de leur envoi dans le pipeline, généralement pour être directement indexées dans Elasticsearch.

Logstash peut extraire de presque toutes les sources de données à l'aide de plug-ins d'entrée, appliquer une grande variété de transformations et d'améliorations de données à l'aide de plug-ins de filtrage et expédier les données vers un grand nombre de destinations à l'aide de plug-ins de sortie. Le rôle que joue Logstash dans la pile est donc essentiel - il vous permet de filtrer, masser et façonner vos données afin qu'elles soient plus faciles à utiliser.

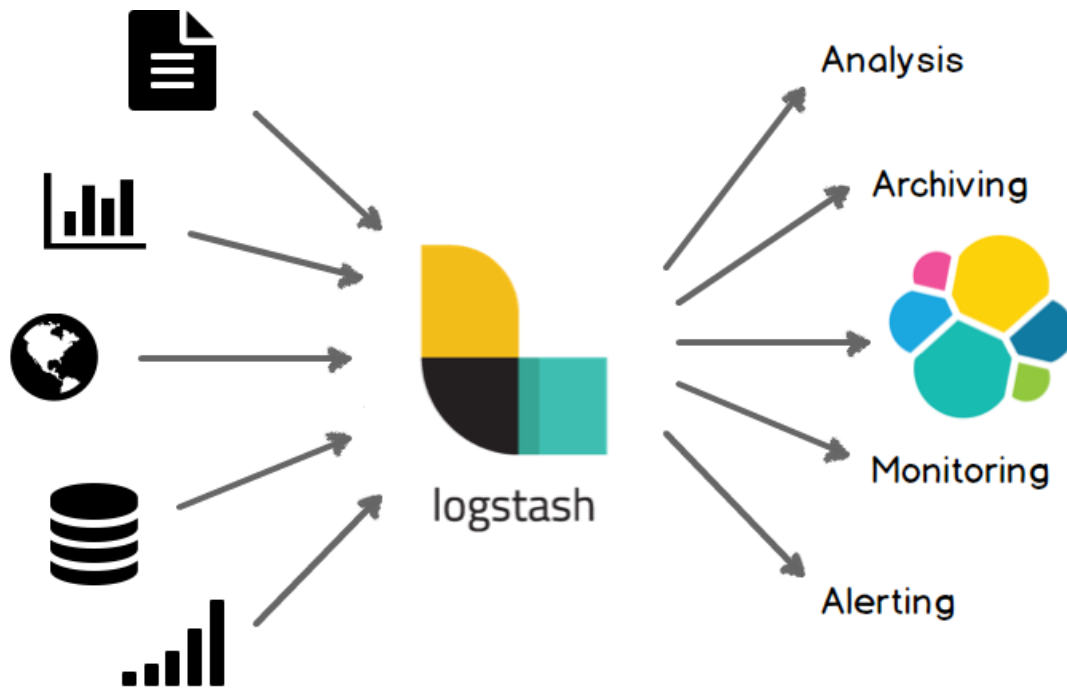


Figure 8 : Logstash

### 3.6 Kibana

Kibana est une application front end gratuite et ouverte qui s'appuie sur la Suite Elastic. Elle permet de rechercher et de visualiser les données indexées dans Elasticsearch. Si Kibana est connue pour être l'outil de représentation graphique de la Suite Elastic (ELK), elle sert aussi d'interface utilisateur pour le monitoring, la gestion et la sécurité des clusters de la Suite Elastic. Sans oublier qu'elle joue aussi le rôle de hub centralisé pour des solutions intégrées, développées sur la Suite Elastic. Créée en 2013 au sein de la communauté Elasticsearch, Kibana s'est développée pour offrir une vue à 360° sur la Suite Elastic, devenant ainsi un véritable portail pour les utilisateurs et les entreprises.

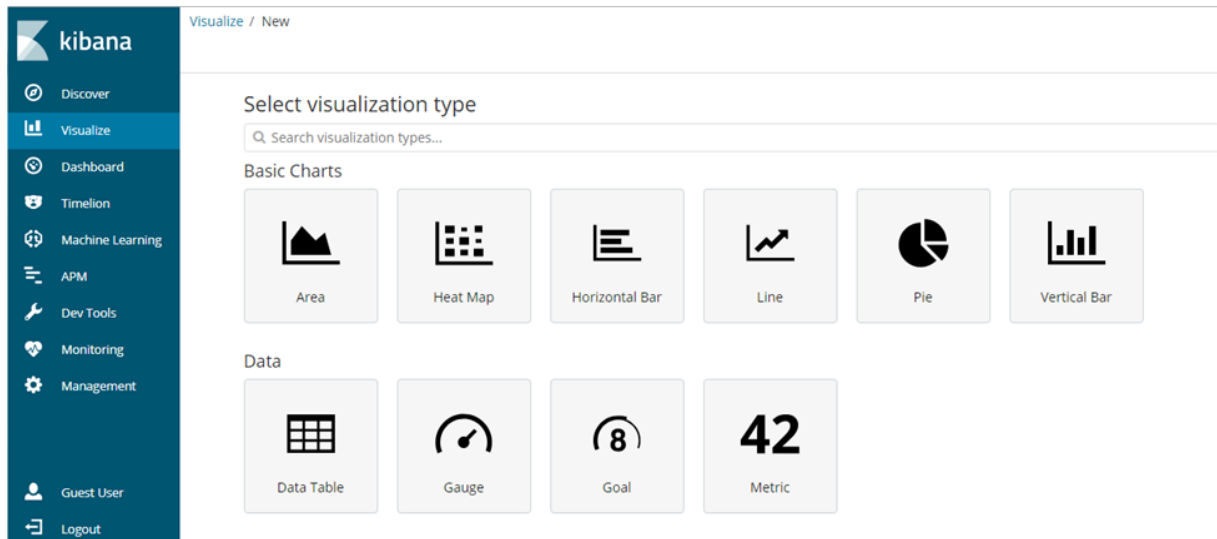


Figure 9 : Interface de Kibana

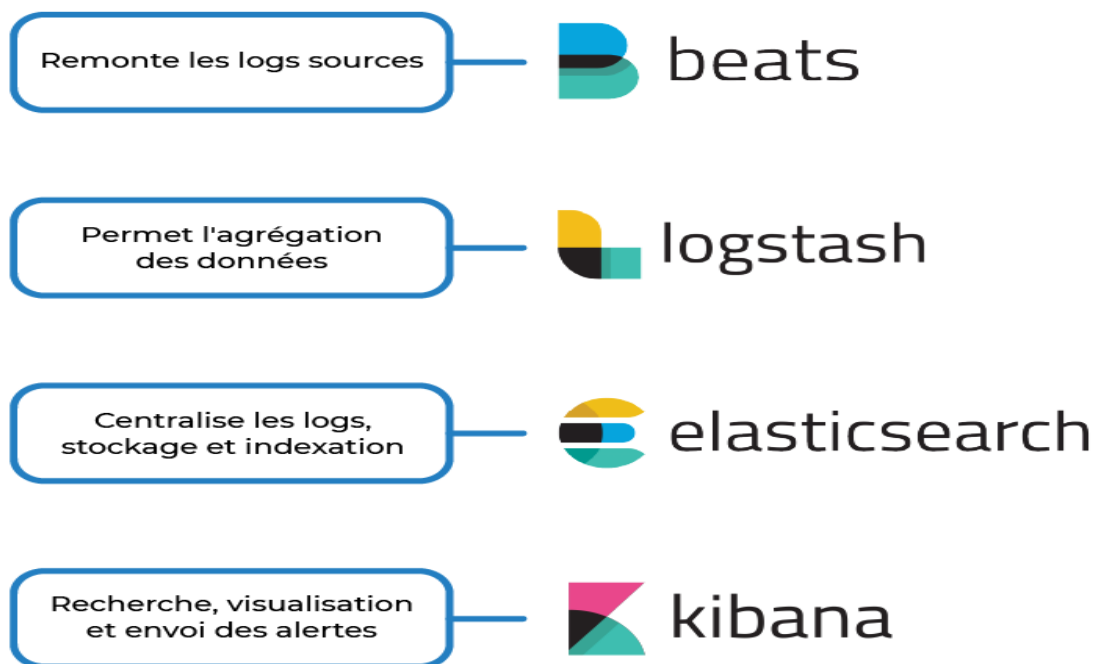


Figure 10 : Le rôle de chaque élément dans ELK

## Conclusion

Dans ce chapitre, nous avons présenté d'une manière détaillée les outils qu'on a utilisés dans notre projet.

# Chapitre 4: Application de système de détection réseau

Dans ce chapitre, nous allons effectuer les installations et les configurations nécessaires pour notre NIDS, ainsi un test de fonctionnement à la fin.

## 4.1 Présentation de l'architecture

Les logs utilisés sont générés à partir d'un fichier pcap, on a utilisé *TCP Replay* c'est un outil très utile pour tester un IDS. Il nous permet de passer une capture de paquet enregistrée via notre interface de sniffing.

Alors Filebeat va envoyer les logs vers Logstash, qui va les collecter et les transformer en des messages JSON, puis il les introduit dans les clusters d'Elasticsearch.

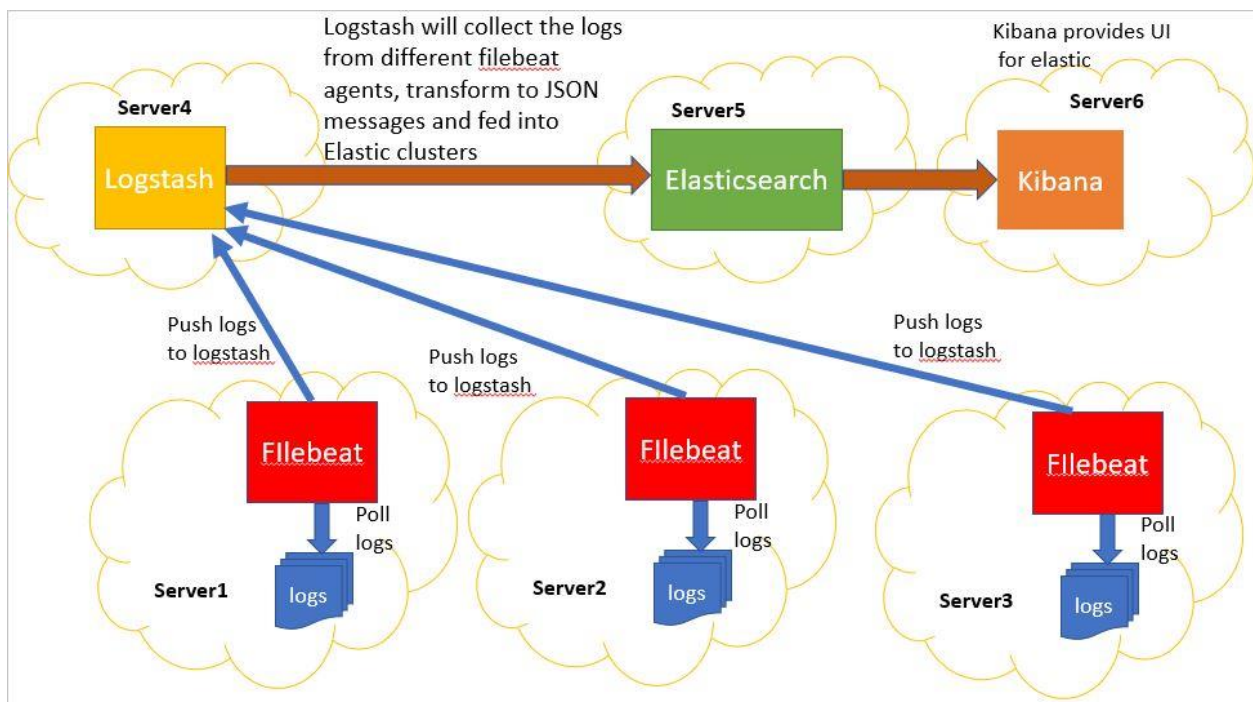


Figure 11 : Architecture de projet

### 4.1.1 Les outils utilisés pour effectuer ce NIDS

- Un PC portable Lenovo ThinkPad P70 avec :
  - 16GB en RAM.
  - Windows 10 64bits.
  - Processeur Intel i7 6ème génération.
- Deux machines virtuelles VM Virtualbox, la première contient Suricata, Zeek, Filebeat. Et la deuxième contient Elasticsearch, Logstash et Kibana.
  - La première machine est Ubuntu Server avec la configuration suivante :
    - 4GB en RAM.
    - 2 CPU
  - La deuxième machine est Kali Linux.

### 4.1.2 Configuration de l'interface de sniffing

Un IDS a toujours au moins deux interfaces. La première s'appelle l'interface de gestion. La deuxième interface est communément connue sous le nom d'interface de sniffing, son travail consiste à recevoir tous les paquets envoyés sur le réseau. Pour ce faire, l'interface doit être mise dans ce qu'on appelle le mode Promiscuous. Habituellement, une interface ne recevra que les paquets qui lui sont adressés, mais lorsqu'elle est configurée en mode Promiscuous, l'interface recevra tous les paquets.

Ubuntu utilise par défaut un système de gestion de réseau appelé NetPlan qui ne prend pas en charge la configuration du mode Promiscuous. Pour contourner ce problème, nous allons installer les anciens outils de gestion de réseau et supprimer NetPlan.

```
Network connections [ Help ]

Configure at least one interface this server can use to talk to other machines,
and which preferably provides sufficient access for updates.

NAME    TYPE    NOTES
[ enp0s3 eth -          ▶ ]
DHCPv4  192.168.1.10/24
08:00:27:36:20:24 / Intel Corporation / 82540EM Gigabit Ethernet Controller
(PRO/1000 MT Desktop Adapter)

[ enp0s8 eth -          ▶ ]
DHCPv4  192.168.77.114/24
08:00:27:94:57:c7 / Intel Corporation / 82545EM Gigabit Ethernet Controller
(Copper) (PRO/1000 MT Single Port Adapter)
```

Figure 12 : Deux Adaptateurs réseau, Bridged Adapter et Host-only Adapter

- *ip addr (On note les noms des interfaces, ma machine virtuelle utilise enp0s3 et enp0s8)*
- *sudo apt install -y ifupdown*
- *sudo nano /etc/network/interfaces*

```
GNU nano 6.2 /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)
# Include files from /etc/network/interfaces.d:
source /etc/network/interfaces.d/*
#loopback
auto lo
iface lo inet loopback
#management interface
allow-hotplug enp0s3
iface enp0s3 inet dhcp
#sniffing interface
allow-hotplug enp0s8
iface enp0s8 inet manual
up ifconfig enp0s8 promisc up
down ifconfig enp0s8 promisc down
```

Figure 13 : Fichier de configuration des interfaces

On arrête les services réseau, supprime NetPlan et on redémarre avec les commandes suivantes :

- *sudo service systemd-networkd stop*
- *sudo apt remove -y netplan*
- *sudo apt install net-tools -y*
- *sudo reboot now*

On remarque d'abord que l'interface `enp0s8` est mise dans le mode Promiscuous. Donc on peut recevoir tous les paquets.

```
sics@ensao:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:7c:91:2c brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.18/24 brd 192.168.1.255 scope global dynamic enp0s3
        valid_lft 86091sec preferred_lft 86091sec
    inet6 fe80::a00:27ff:fe7c:912c/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,PROMISC,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:82:57:9f brd ff:ff:ff:ff:ff:ff
    inet 192.168.77.113/24 metric 100 brd 192.168.77.255 scope global dynamic enp0s8
        valid_lft 548sec preferred_lft 548sec
    inet6 fe80::a00:27ff:fe82:579f/64 scope link
        valid_lft forever preferred_lft forever
```

Figure 14 : Configuration d'interface de sniffing

## 4.2 Installation et configuration de Suricata

### 4.2.1 Installation :

- *sudo apt install software-properties-common*
- *sudo add-apt-repository ppa:oisf/suricata-stable*
- *sudo apt update*
- *sudo apt install -y suricata*
- *sudo suricata-update*

### 4.2.2 Configuration

*sudo nano /etc/suricata/suricata.yaml*

L'ensemble suivant est les principales choses à accomplir :

- On ajoute toutes les plages d'adresses IP à la liste des variables *home-net* qui ne sont pas déjà couvertes.
- Dans la section sur les sorties de journalisation, on assure que *eve-log* a une valeur oui pour activé.



- On remplace eth0 par le nom de notre interface de sniffing (dans mon cas, j'avais enp0s8).
- Nous faisons défiler jusqu'à la validation de la somme de contrôle sous la section flux et définissons la valeur sur non.

```
GNU nano 6.2 /etc/suricata/suricata.yaml
#YAML 1.1
---

# Suricata configuration file. In addition to the comments describing all
# options in this file, full documentation can be found at:
# https://suricata.readthedocs.io/en/latest/configuration/suricata-yaml.html

# This configuration file generated by Suricata 6.0.12.
suricata-version: "6.0"

##
## Step 1: Inform Suricata about your network
##

vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12,192.168.1.0/24]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
```

Figure 15 : Fichier de configuration de Suricata

Puis on crée un service Suricata avec la commande `sudo nano /lib/systemd/system/suricata.service`

Fondamentalement, ce fichier indique au système les commandes et les options à utiliser lors de l'exécution de Suricata. De cette façon, nous pouvons l'exécuter en arrière-plan sans spécifier à chaque fois des éléments tels que les chemins de configuration.

```

GNU nano 6.2 /lib/systemd/system/suricata.service
[Unit]
Description=Suricata Intrusion Detection Service
After=syslog.target network-online.target

[Service]
# Environment file to pick up $OPTIONS. On Fedora/EL this would be
# /etc/sysconfig/suricata, or on Debian/Ubuntu, /etc/default/suricata.
#EnvironmentFile=/etc/sysconfig/suricata
#EnvironmentFile=/etc/default/suricata
ExecStartPre=/bin/rm -f /var/run/suricata.pid
ExecStart=/usr/bin/suricata -c /etc/suricata/suricata.yaml --pidfile /var/run/suricata.pid
ExecReload=/bin/kill -USR2 $MAINPID

[Install]
WantedBy=multi-user.target

```

Figure 16 : Suricata service

## 4.3 Installation et configuration de Zeek

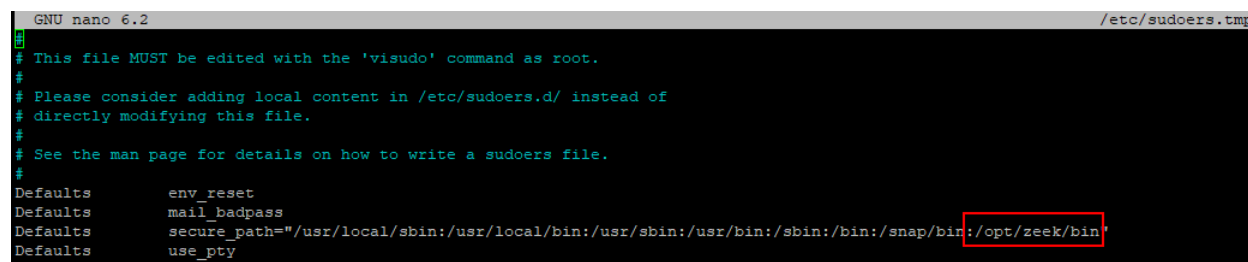
### 4.3.1 Installation

- *sudo apt install git -y*
- *sudo apt install -y cmake make gcc g++ flex bison libpcap-dev libssl-dev python-dev swig zlib1g-dev*
- *sudo git clone --recursive https://github.com/zeek/zeek*
- *cd zeek*
- *./configure*
- *make*
- *sudo make install*

Nous voulons pouvoir exécuter Zeek sans naviguer dans son répertoire. Pour ce faire, nous l'ajouterons à la variable de chemin du système. Fondamentalement, il s'agit de la liste des répertoires approuvés que le système recherche lorsqu'un programme est appelé. Dans la plupart des cas, nous ajouterions le nouveau chemin au fichier */etc/environment* afin qu'il s'applique à tous les utilisateurs.

*sudo visudo*

Puis on ajoute `:/opt/zeek/bin` dans *secure\_path*

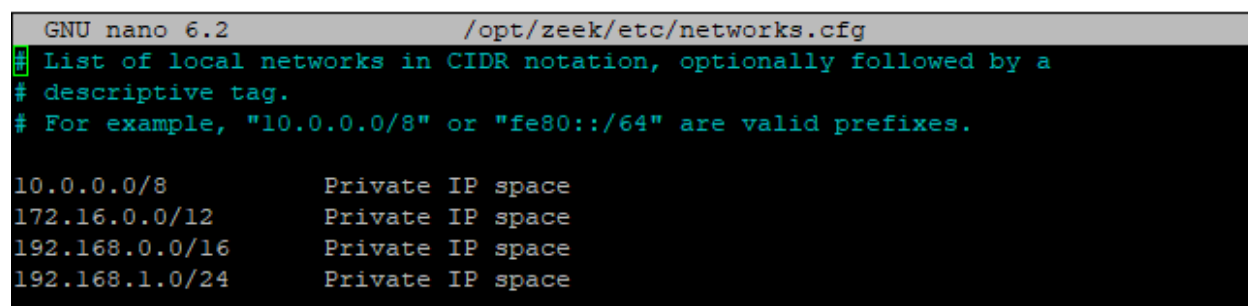


```
GNU nano 6.2 /etc/sudoers.tmp
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
# See the man page for details on how to write a sudoers file.
#
Defaults    env_reset
Defaults    mail_badpass
Defaults    secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin:/opt/zeek/bin"
Defaults    use_pty
```

Figure 17 : Zeek bin

### 4.3.2 Configuration

Zeek a trois fichiers distincts qui contiennent les informations de configuration. Le premier est la configuration des réseaux que Zeek surveillera. Ici, nous pouvons spécifier les plages d'adresses IP que nous souhaitons que Zeek couvre. Par défaut, la plupart des plages d'adresses IP privées sont couvertes. On entre, *sudo nano /opt/zeek/etc/networks.cfg*, pour modifier le fichier.



```
GNU nano 6.2 /opt/zeek/etc/networks.cfg
# List of local networks in CIDR notation, optionally followed by a
# descriptive tag.
# For example, "10.0.0.0/8" or "fe80::/64" are valid prefixes.

10.0.0.0/8      Private IP space
172.16.0.0/12   Private IP space
192.168.0.0/16  Private IP space
192.168.1.0/24  Private IP space
```

Figure 18 : Zeek networks.cfg

On entre ensuite la commande, *sudo nano /opt/zeek/etc/node.cfg*, pour modifier la configuration du nœud Zeek lui-même. Pour nos besoins, on laisse les valeurs par défaut à l'exception du nom de l'interface de détection (dans mon cas, il s'agit d'enp0s8).

Notez que c'est dans ce fichier qu'un cluster de nœuds Zeek peut être configuré pour fonctionner ensemble.

```

GNU nano 6.2 /opt/zeek/etc/node.cfg
# Example ZeekControl node configuration.
#
# This example has a standalone node ready to go except for possibly changing
# the sniffing interface.

# This is a complete standalone configuration. Most likely you will
# only need to change the interface.
[zeek]
type=standalone
host=localhost
interface=enp0s8

## Below is an example clustered configuration. If you use this,
## remove the [zeek] node above.

#[logger-1]
#type=logger
#host=localhost

```

Figure 19 : Zeek node.cfg

Nous entrerons ensuite la commande, *sudo nano /opt/zeek/etc/zeekctl.cfg*, pour configurer le reste des paramètres de gestion Zeek. On apporte les modifications suivantes au fur et à mesure que nous parcourons le fichier de configuration :

- On assure que tous les paramètres de messagerie sont définis sur 0, car nous n'utiliserons pas ces fonctionnalités.
- On modifie les valeurs de rotation du journal sur un paramètre bas. Étant donné que nos nœuds disposent d'une petite quantité d'espace, on règle *LogExpireInterval* sur 48h.
- Idem avec l'intervalle de journalisation des statistiques, on le définit sur une petite valeur, comme 1 ou 2 jours.
- Enfin, on définit le chemin du journal pour Zeek sur */var/log/zeek/logs*.

```

GNU nano 6.2 /opt/zeek/etc/zeekctl.cfg
## Global ZeekControl configuration file.

#####
# Mail Options

# Recipient address for all emails sent out by Zeek and ZeekControl.
MailTo = root@localhost

# Mail connection summary reports each log rotation interval. A value of 1
# means mail connection summaries, and a value of 0 means do not mail
# connection summaries. This option has no effect if the trace-summary
# script is not available.
MailConnectionSummary = 0

# Lower threshold (in percentage of disk space) for space available on the
# disk that holds SpoolDir. If less space is available, "zeekctl cron" starts
# sending out warning emails. A value of 0 disables this feature.
MinDiskSpace = 5

# Send mail when "zeekctl cron" notices the availability of a host in the
# cluster to have changed. A value of 1 means send mail when a host status
# changes, and a value of 0 means do not send mail.
MailHostUpDown = 0

#####
# Logging Options

# Rotation interval in seconds for log files on manager (or standalone) node.
# A value of 0 disables log rotation.
LogRotationInterval = 3600

```

Figure 20 : Zeek zeekctl.cfg

Nous devons d'abord changer le format de sortie des journaux en JSON.  
*sudo nano /opt/zeek/share/zeek/site/local.zeek*

```

GNU nano 6.2 /opt/zeek/share/zeek/site/local.zeek
# Extend email alerting to include hostnames
@load policy/frameworks/notice/extend-email/hostnames

# Uncomment the following line to enable detection of the heartbleed attack. En
# this might impact performance a bit.
# @load policy/protocols/ssl/heartbleed

# Uncomment the following line to enable logging of connection VLANs. Enabling
# this adds two VLAN fields to the conn.log file.
# @load policy/protocols/conn/vlan-logging

# Uncomment the following line to enable logging of link-layer addresses. Enabl
# this adds the link-layer address for each connection endpoint to the conn.log
# @load policy/protocols/conn/mac-logging

# Uncomment this to source zkg's package state
# @load packages
# Output to JSON
@load policy/tuning/json-logs.zeek

```

Figure 21 : Zeek JSON output

ZeekCtl nécessite un ajout au crontab pour configurer ses activités de maintenance régulières telles que la rotation des journaux.

*Sudo crontab -e*

```
GNU nano 6.2 /tmp/crontab.4z36Fw/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
*/5 * * * * /opt/zeek/bin/zeekctl cron
```

Figure 22 : Zeek Cronjob

## 4.4 Installation et configuration de Filebeat

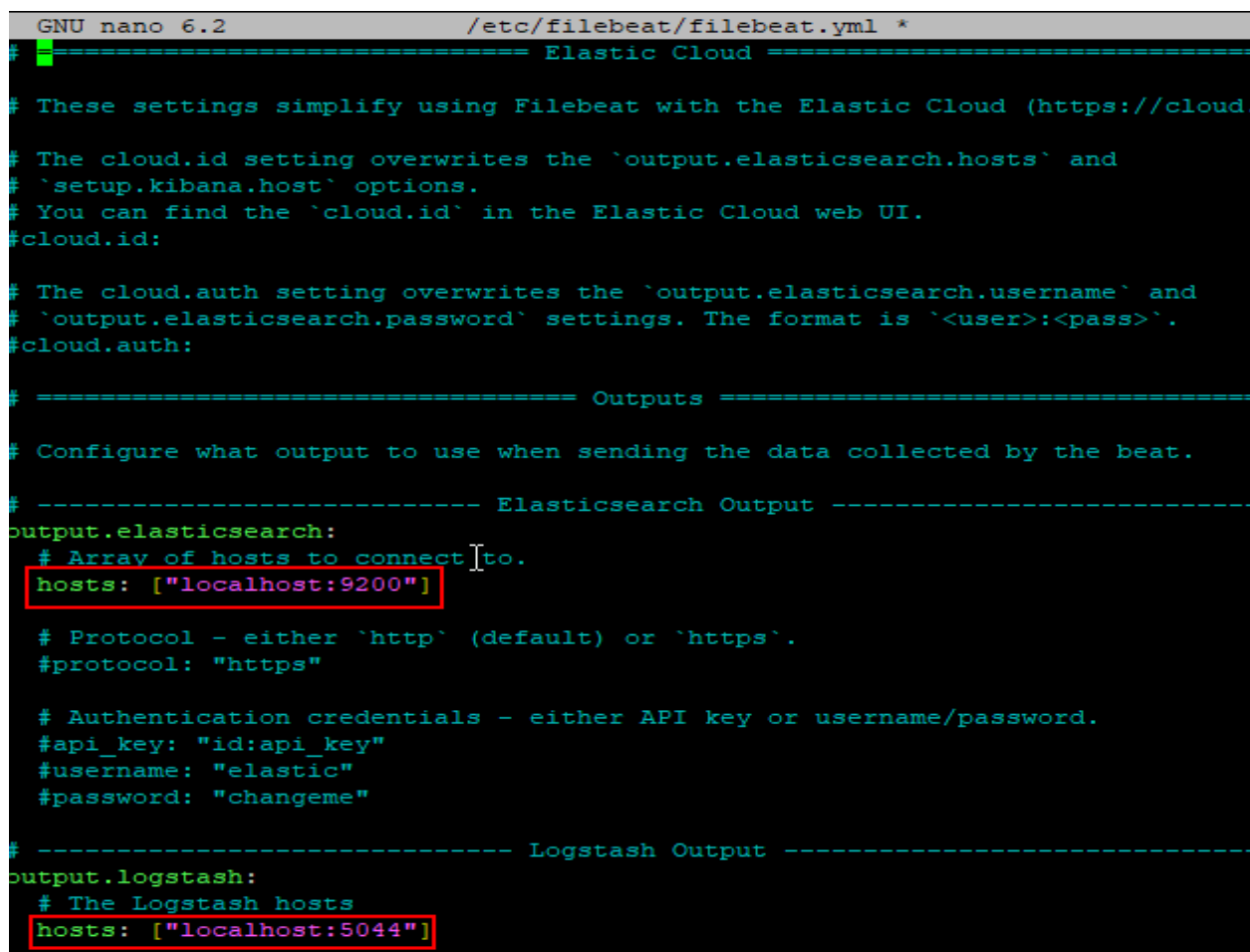
### 4.4.1 Installation

- *wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -*
- *sudo apt-get install apt-transport-https*
- *echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-7.x.list*
- *sudo apt update && sudo apt install Filebeat*

## 4.4.2 Configuration

*sudo nano /etc/filebeat/filebeat.yml*

- Décommenter et remplacer la valeur de l'hôte Kibana par `${kibana_host}`
- L'hôte et le port utilisés est `localhost:5601`
- Décommenter et remplacer la valeur de l'hôte Elasticsearch par `${es_host}`
- L'hôte et le port utilisés est `localhost:9200`



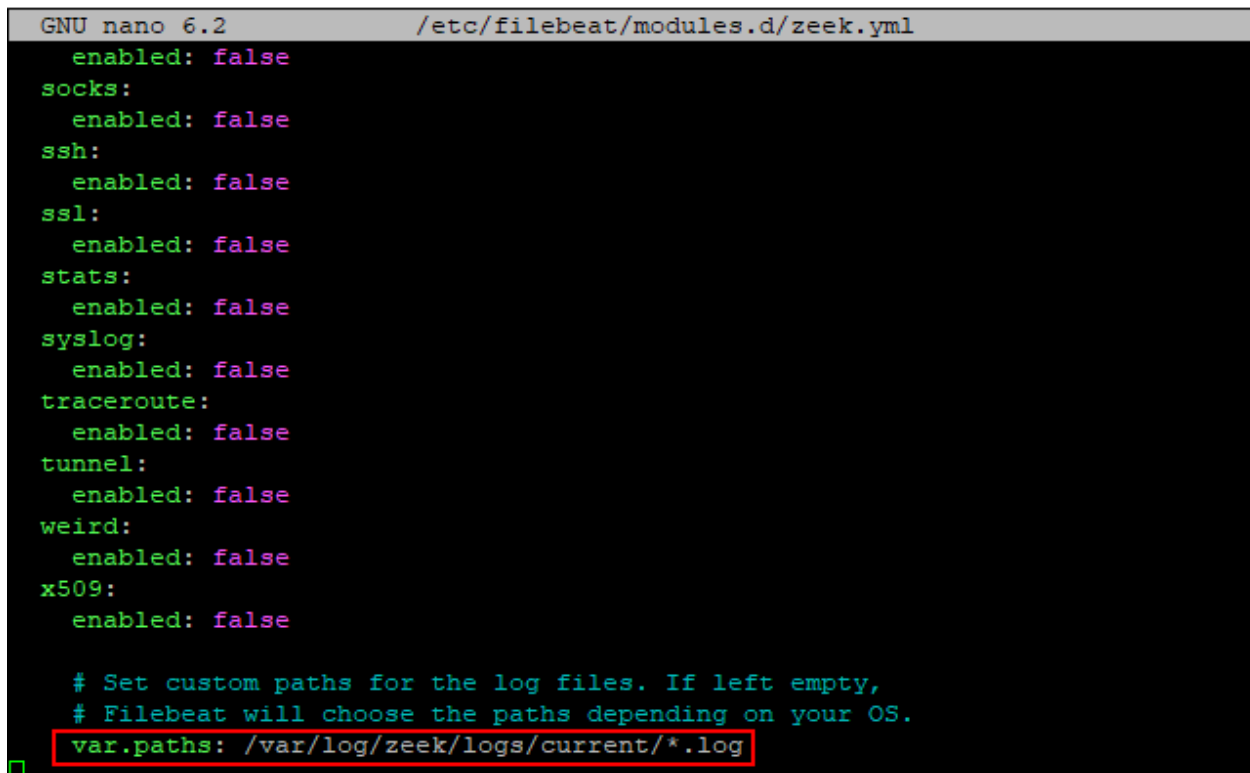
```
GNU nano 6.2 /etc/filebeat/filebeat.yml *
# ===== Elastic Cloud =====
# These settings simplify using Filebeat with the Elastic Cloud (https://cloud.
# The cloud.id setting overwrites the `output.elasticsearch.hosts` and
# `setup.kibana.host` options.
# You can find the `cloud.id` in the Elastic Cloud web UI.
#cloud.id:
# The cloud.auth setting overwrites the `output.elasticsearch.username` and
# `output.elasticsearch.password` settings. The format is `<user>:<pass>`.
#cloud.auth:
# ===== Outputs =====
# Configure what output to use when sending the data collected by the beat.
# ----- Elasticsearch Output -----
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["localhost:9200"]
  # Protocol - either `http` (default) or `https`.
  #protocol: "https"
  # Authentication credentials - either API key or username/password.
  #api_key: "id:api_key"
  #username: "elastic"
  #password: "changeme"
# ----- Logstash Output -----
output.logstash:
  # The Logstash hosts
  hosts: ["localhost:5044"]
```

Figure 23 : Filebeat configuration

D'abord on active et configure les modules. Nous utiliserons les modules intégrés pour traiter les journaux. Les modules ont des pipelines d'ingestion prédéfinis qui traduiront les journaux en *Elastic Common Schema*.

Elastic Common Schema (ECS) est une spécification open source, développée avec le soutien de la communauté d'utilisateurs Elastic. ECS définit un ensemble commun de champs à utiliser lors du stockage des données d'événement dans Elasticsearch, tels que les journaux et les métriques.

```
sudo filebeat modules enable suricata zeek  
sudo nano /etc/filebeat/modules.d/zeek.yml
```



```
GNU nano 6.2 /etc/filebeat/modules.d/zeek.yml  
  enabled: false  
socks:  
  enabled: false  
ssh:  
  enabled: false  
ssl:  
  enabled: false  
stats:  
  enabled: false  
syslog:  
  enabled: false  
traceroute:  
  enabled: false  
tunnel:  
  enabled: false  
weird:  
  enabled: false  
x509:  
  enabled: false  
  
# Set custom paths for the log files. If left empty,  
# Filebeat will choose the paths depending on your OS.  
var.paths: /var/log/zeek/logs/current/*.log
```

Figure 24 : Filebeat zeek.yml

## 4.5 Installation et configuration d'Elasticsearch

### 4.5.1 Installation

- *wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-keyring.gpg*
- *sudo apt-get install apt-transport-https*



- `echo "[signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg] https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.list`
- `sudo apt-get update && sudo apt-get install Elasticsearch`

## 4.5.2 Configuration

On test si Elasticsearch est installé correctement :

```
(root@kali)~# systemctl status elasticsearch.service
● elasticsearch.service - Elasticsearch
   Loaded: loaded (/lib/systemd/system/elasticsearch.service; enabled; vendor preset: disabled)
   Active: active (running) since Thu 2023-06-08 06:30:28 EDT; 3min 55s ago
     Docs: https://www.elastic.co
   Main PID: 787 (java)
    Tasks: 79 (limit: 3499)
   Memory: 1.5G
      CPU: 1min 12.422s
   CGroup: /system.slice/elasticsearch.service
           └─ 787 /usr/share/elasticsearch/jdk/bin/java -Xms4m -Xmx64m -XX:+UseSerialGC ->
              1148 /usr/share/elasticsearch/jdk/bin/java -Des.networkaddress.cache.ttl=60 ->
              1171 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-x86_64/bin/co>
```

Figure 25 : Elasticsearch service

```
(root@kali)~# curl 192.168.1.11:9200
{"name": "node-1",
 "cluster_name": "pfa",
 "cluster_uuid": "tt5hioiVQxOQIKYxO4W4aw",
 "version": {
   "number": "7.17.10",
   "build_flavor": "default",
   "build_type": "deb",
   "build_hash": "fec6d68e3150eda0c307ab9a9d7557f5d5fd71349",
   "build_date": "2023-04-23T05:33:18.138275597Z",
   "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"}
```

Figure 26 : Curl Elasticsearch server

On règle quelques configurations pour adapter Elasticsearch à nos besoins, et Notre Elasticsearch host est : <http://192.168.1.11:9200>

```
# ----- Cluster -----
# Management
# Use a descriptive name for your cluster:
#
cluster.name: pfa
# Time field: '@timestamp' Default
# ----- Node -----
# Ingest Pipelines
# Use a descriptive name for the node:
#
node.name: node-1
# Fields (11) Scripted fields (0) Field filters (0)
# ----- Network -----
# By default Elasticsearch is only accessible on localhost. Set a different
# address here to expose this node on the network:
#
network.host: 192.168.1.11
# By default Elasticsearch listens for HTTP traffic on the first free port it
# finds starting at 9200. Set a specific HTTP port here:
#
http.port: 9200
```

Figure 27 : Configuration d'Elasticsearch

## 4.6 Installation et configuration de Logstash

### 4.6.1 Installation

- `wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/elastic-keyring.gpg`
- `sudo apt-get install apt-transport-https`
- `echo "deb [signed-by=/usr/share/keyrings/elastic-keyring.gpg] https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-8.x.list`
- `sudo apt-get update && sudo apt-get install Logstash`

### 4.6.2 Configuration

On vérifie si Logstash est installé correctement :

```

(kali㉿kali)-[~]
└─$ systemctl status logstash.service
● logstash.service - logstash
   Loaded: loaded (/etc/systemd/system/logstash.service; disabled; vendor prese>
   Active: active (running) since Thu 2023-06-08 18:08:14 EDT; 3min 45s ago
     Main PID: 6115 (java)
        Tasks: 37 (limit: 4632)
   Memory: 788.4M
      CPU: 2min 34.885s
   CGroup: /system.slice/logstash.service
           └─6115 /usr/share/logstash/jdk/bin/java -Xms1g -Xmx1g -XX:+UseConcMa>

Jun 08 18:10:07 kali logstash[6115]: [2023-06-08T18:10:07,748][INFO ][logstash.ou>
Jun 08 18:10:20 kali logstash[6115]: [2023-06-08T18:10:20,946][INFO ][logstash.fi>
Jun 08 18:11:04 kali logstash[6115]: [2023-06-08T18:11:04,284][INFO ][logstash.fi>
Jun 08 18:11:04 kali logstash[6115]: [2023-06-08T18:11:04,296][INFO ][logstash.fi>
Jun 08 18:11:05 kali logstash[6115]: [2023-06-08T18:11:05,799][INFO ][logstash.ja>
Jun 08 18:11:10 kali logstash[6115]: [2023-06-08T18:11:10,685][INFO ][logstash.ja>
Jun 08 18:11:10 kali logstash[6115]: [2023-06-08T18:11:10,932][INFO ][logstash.ja>
Jun 08 18:11:11 kali logstash[6115]: [2023-06-08T18:11:11,001][INFO ][filewatch.o>
Jun 08 18:11:11 kali logstash[6115]: [2023-06-08T18:11:11,385][INFO ][logstash.ag>
Jun 08 18:11:12 kali logstash[6115]: [2023-06-08T18:11:12,117][WARN ][filewatch.t>

```

Figure 28 : Logstash service

Puis on indique la méthode avec laquelle on veut voir nos logs sur Kibana :

```

input {
  file {
    path => "/Home/kali/Downloads/apache-daily-access.log"
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}

filter {
  grok { match => { "message" => "%{COMBINEDAPACHELOG}" }}
  date { match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ] }
  geoip { source => "clientip" }
}

output { elasticsearch { hosts => ["192.168.1.11:9200"] } }

```

Figure 29 : Configuration d'index de Logstash

## 4.7 Installation et configuration de Kibana

### 4.7.1 Installation

- `wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-keyring.gpg`
- `sudo apt-get install apt-transport-https`
- `echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg] https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.list`
- `sudo apt-get update && sudo apt-get install kibana`

### 4.7.2 Configuration

On vérifie si Kibana est installée correctement :

```
(root@kali)~# systemctl status kibana.service
● kibana.service - Kibana
   Loaded: loaded (/lib/systemd/system/kibana.service; enabled; vendor preset: disabled)
   Active: active (running) since Thu 2023-06-08 06:45:39 EDT; 2s ago
     Docs: https://www.elastic.co
   Main PID: 2440 (node)
    Tasks: 7 (limit: 3499)
   Memory: 108.8M
      CPU: 2.602s
   CGroup: /system.slice/kibana.service
           └─2440 /usr/share/kibana/bin/../../node/bin/node /usr/share/kibana/bin/../../src/cli>

Jun 08 06:45:39 kali systemd[1]: Started Kibana.
```

Figure 30 : Kibana service

Notre Kibana host est <http://192.168.1.11:5601> :

```
# Kibana is served by a back end server. This setting specifies the port to use.
server.port: 5601

# Specifies the address to which the Kibana server will bind. IP addresses and host names are both valid values.
# The default is 'localhost', which usually means remote machines will not be able to connect.
# To allow connections from remote users, set this parameter to a non-loopback address.
server.host: 192.168.1.11

# Enables you to specify a path to mount Kibana at if you are running behind a proxy.
# Use the `server.rewriteBasePath` setting to tell Kibana if it should remove the basePath
# from requests it receives, and to prevent a deprecation warning at startup.
# This setting cannot end in a slash.
#server.basePath: ""

# Specifies whether Kibana should rewrite requests that are prefixed with
# `server.basePath` or require that they are rewritten by your reverse proxy.
# This setting was effectively always `false` before Kibana 6.3 and will
# default to `true` starting in Kibana 7.0.
#server.rewriteBasePath: false

# Specifies the public URL at which Kibana is available for end users. If
# `server.basePath` is configured this URL should end with the same basePath.
# server.publicBaseUrl: "http://192.168.1.11:5601"

# The maximum payload size in bytes for incoming server requests.
#server.maxPayload: 1048576

# The Kibana server's name. This is used for display purposes.
#server.name: "your-hostname"

# The URLs of the Elasticsearch instances to use for all your queries.
elasticsearch.hosts: ["http://192.168.1.11:9200"]
```

Figure 31 : Kibana configuration

## 4.8 Test de fonctionnement

### 4.8.1 Implémentation 1 : Logs locaux

Dans ce cas, nous avons travaillé sur la machine Kali Linux, où nous avons lié les logs du Syslog avec ELK + Suricata.

Voici le contenu de syslog :

```
(kali@kali)~$ cat /var/log/syslog | tail
Jun  9 13:41:10 kali metricbeat[1893]: 2023-06-09T13:41:10.594-0400#011INFO#011[monitoring]#011log/log.go:184#011Non-zero metrics in the last 30s#011{"monitoring":
  "tricks": {"beat":{"cgroup":{"memory":{"mem":{"usage":{"bytes":-2965504}}},"cpu":{"system":{"ticks":58840,"time":{"ms":695},"total":{"ticks":84560,"time":{"ms":10
    "alue":84560},"user":{"ticks":25720,"time":{"ms":323}}},"handles":{"limit":{"hard":524288,"soft":524288},"open":9},"info":{"ephemeral_id":"1e0fd7c7-2227-48b0-9b8d-
    16853d","uptime":{"ms":1662435},"version":"7.17.10"},"memstats":{"gc_next":22282184,"memory_alloc":18215784,"memory_total":3526018864,"rss":48971776},"runtime":{"
    "ines":60},"libbeat":{"config":{"module":{"running":3},"output":{"events":{"acked":48,"active":0,"batches":6,"total":48},"read":{"bytes":4536},"write":{"bytes":1
    "pipeline":{"clients":10,"events":{"active":0,"published":48,"total":48},"queue":{"acked":48},"metricbeat":{"system":{"cpu":{"events":3,"success":3},"filesystem
    "events":3,"success":3},"fsstat":{"events":1,"success":1},"load":{"events":3,"success":3},"memory":{"events":3,"success":3},"network":{"events":9,"success":9},"pro
    {"events":20,"success":20},"process_summary":{"events":3,"success":3},"socket_summary":{"events":3,"success":3},"system":{"load":{"1":0.55,"15":1.52,"5":0.75,"n
    "1":0.275,"15":0.76,"5":0.375}}}}}}
Jun  9 13:41:40 kali metricbeat[1893]: 2023-06-09T13:41:40.591-0400#011INFO#011[monitoring]#011log/log.go:184#011Non-zero metrics in the last 30s#011{"monitoring":
  "tricks": {"beat":{"cgroup":{"memory":{"mem":{"usage":{"bytes":-397312}}},"cpu":{"system":{"ticks":59870,"time":{"ms":1028},"total":{"ticks":85990,"time":{"ms":14
    "alue":85990},"user":{"ticks":26120,"time":{"ms":402}}},"handles":{"limit":{"hard":524288,"soft":524288},"open":9},"info":{"ephemeral_id":"1e0fd7c7-2227-48b0-9b8d-
    16853d","uptime":{"ms":1692432},"version":"7.17.10"},"memstats":{"gc_next":19328920,"memory_alloc":14165856,"memory_total":3589780176,"rss":44974080},"runtime":{"
    "ines":60},"libbeat":{"config":{"module":{"running":3},"output":{"events":{"acked":49,"active":0,"batches":6,"total":49},"read":{"bytes":4549},"write":{"bytes":1
    "pipeline":{"clients":10,"events":{"active":0,"published":49,"total":49},"queue":{"acked":49},"metricbeat":{"system":{"cpu":{"events":3,"success":3},"load":{"
    "3,"success":3},"memory":{"events":3,"success":3},"network":{"events":9,"success":9},"process":{"events":25,"success":25},"process_summary":{"events":3,"success
    "ocket_summary":{"events":3,"success":3},"system":{"load":{"1":0.75,"15":1.5,"5":0.78,"norm":{"1":0.375,"15":0.75,"5":0.39}}}}}}
Jun  9 13:42:10 kali metricbeat[1893]: 2023-06-09T13:42:10.590-0400#011INFO#011[monitoring]#011log/log.go:184#011Non-zero metrics in the last 30s#011{"monitoring":
  "tricks": {"beat":{"cgroup":{"memory":{"mem":{"usage":{"bytes":-4409600}}},"cpu":{"system":{"ticks":60560,"time":{"ms":763},"total":{"ticks":87070,"time":{"ms":107
    "alue":87070},"user":{"ticks":27040,"time":{"ms":440}}},"handles":{"limit":{"hard":524288,"soft":524288},"open":9},"info":{"ephemeral_id":"1e0fd7c7-2227-48b0-9b8d-
    16853d","uptime":{"ms":1702432},"version":"7.17.10"},"memstats":{"gc_next":19328920,"memory_alloc":14165856,"memory_total":3589780176,"rss":44974080},"runtime":{"
    "ines":60},"libbeat":{"config":{"module":{"running":3},"output":{"events":{"acked":49,"active":0,"batches":6,"total":49},"read":{"bytes":4549},"write":{"bytes":1
    "pipeline":{"clients":10,"events":{"active":0,"published":49,"total":49},"queue":{"acked":49},"metricbeat":{"system":{"cpu":{"events":3,"success":3},"load":{"
    "3,"success":3},"memory":{"events":3,"success":3},"network":{"events":9,"success":9},"process":{"events":25,"success":25},"process_summary":{"events":3,"success
    "ocket_summary":{"events":3,"success":3},"system":{"load":{"1":0.75,"15":1.5,"5":0.78,"norm":{"1":0.375,"15":0.75,"5":0.39}}}}}}
```

Figure 32 : Syslog



Et voilà on peut les visualiser sur notre interface Kibana :

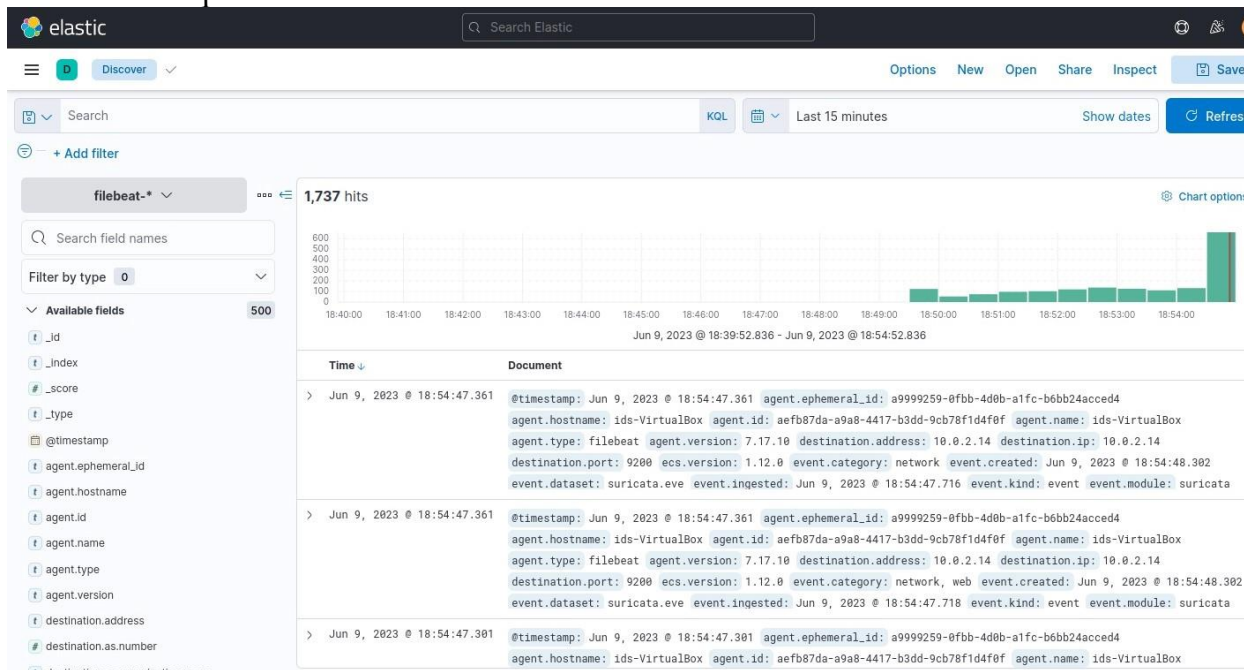


Figure 33 : Kibana Syslog

Ainsi en peut voir les alertes générées par Suricata et basées sur syslog, on a simulé une attaque de SSH bruteforcing :



Figure 34 : SSH bruteforcing 1

SSH login attempts [Filebeat System] ECS

8 documents

Time	system.auth.ssh.event	system.auth.ssh.method	user.name	source.ip	source.geo.country_iso_code
Jun 11, 2023 @ 14:27:43.000	error:	maximum authentication attempts exceeded	kali	192.168.1.5	-
Jun 11, 2023 @ 14:27:41.000	Failed	password	kali	192.168.1.5	-
Jun 11, 2023 @ 14:27:38.000	Failed	password	kali	192.168.1.5	-

Rows per page: 50

Figure 35 : SSH bruteforcing 2

## 4.8.2 Implémentation 2 : Logs externes

D'abord, on configure l'interface de sniffing sur Zeek (qui est enp0s3 dans cette simulation) dans le fichier 'node.cfg' :

```
# Example ZeekControl node configuration.
#
# This example has a standalone node ready to go except for possibly changing
# the sniffing interface.

# This is a complete standalone configuration. Most likely you will
# only need to change the interface.
[zeek]
type=standalone
host=localhost
interface=enp0s3
```

Figure 36 : Zeek node.cfg

Et on ajoute notre réseau IP dans le fichier 'networks.cfg' :

```
1 # List of local networks in CIDR notation, optionally followed by a
2 # descriptive tag.
3 # For example, "10.0.0.0/8" or "fe80::/64" are valid prefixes.
4
5 10.0.3.0/24      Private IP space
```

Figure 37 : Zeek network.cfg

Puis on spécifie la liste des adresses IP réseaux ainsi notre interface de sniffing enp0s3 :

```
vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[192.168.121.0/24,10.0.0.0/8,172.16.0.0/12]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"
```

Figure 38 : Suricata.yml

```
# Linux high speed capture support
af-packet:
  - interface: enp0s3
    # Number of receive threads. "auto" uses the number of cores
    #threads: auto
    # Default clusterid. AF_PACKET will load balance packets based on flow.
```

Figure 39 : Suricata.yml sniffing interface

Et sur ‘filebeat.yml’ on définit le host du Kibana, l’output d’Elasticsearch et nos identifiants d’authentification :

```
# ===== Kibana =====  
  
# Starting with Beats version 6.0.0, the dashboards are loaded via the Kibana API.  
# This requires a Kibana endpoint configuration.  
setup.kibana:  
  
# Kibana Host  
# Scheme and port can be left out and will be set to the default (http and 5601)  
# In case you specify an additional path, the scheme is required: http://localhost:5601/path  
# IPv6 addresses should always be defined as: https://[2001:db8::1]:5601  
host: "10.0.2.14:5601"  
  
# Kibana Space ID  
# ID of the Kibana Space into which the dashboards should be loaded. By default,  
# the Default Space will be used.  
#space.id:
```

Figure 40 : Kibana host

```
# ----- Elasticsearch Output -----  
output.elasticsearch:  
# Array of hosts to connect to.  
hosts: ["10.0.2.14:9200"]  
  
# Protocol - either `http` (default) or `https`.  
#protocol: "https"  
  
# Authentication credentials - either API key or username/password.  
#api_key: "id:api_key"  
username: "elastic"  
password: "JfbAWx82n03aejGgFPqj"
```

Figure 41 : Elasticsearch output

Du même pour ‘elasticsearch.yml’, on spécifie les chemins où on va stocker notre données et nos logs :

```
# ----- Paths -----  
#  
# Path to directory where to store the data (separate multiple locations by comma):  
#  
path.data: /var/lib/elasticsearch  
#  
# Path to log files:  
#  
path.logs: /var/log/elasticsearch
```

Figure 42 : Elasticsearch.yml logs path

Et maintenant on simule notre attaque en générant quelques logs malveillantes depuis le fichier ‘smallFlaws.pcap’ à l’aide de l’utilité *TCP Replay* :



```
ids@ids-VirtualBox:~$ sudo tcpreplay -i enp0s3 -t -K ~/Downloads/smallFlows.pcap
[sudo] password for ids:
File Cache is enabled
Actual: 14261 packets (9216531 bytes) sent in 0.531882 seconds
Rated: 17328149.8 Bps, 138.62 Mbps, 26812.33 pps
Flows: 1209 flows, 2273.06 fps, 14243 flow packets, 18 non-flow
Statistics for network device: enp0s3
Successful packets:      14261
Failed packets:          0
Truncated packets:      0
Retried packets (ENOBUFS): 0
Retried packets (EAGAIN): 0
```

Figure 43 : TCP Replay

- Pour Suricata, elle affiche les alertes suivantes :



Figure 44 : Suricata events

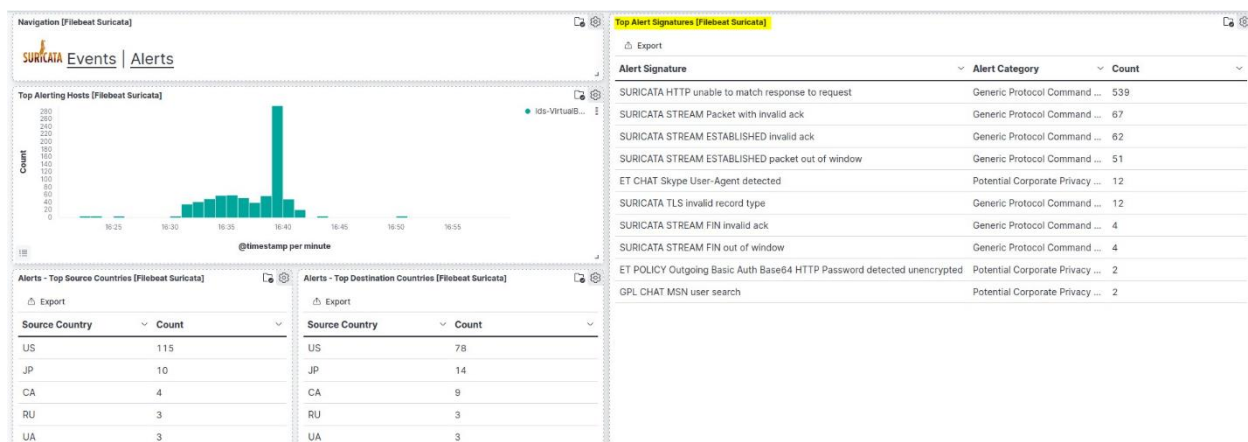


Figure 45 : Suricata alertes

- Et pour Zeek :

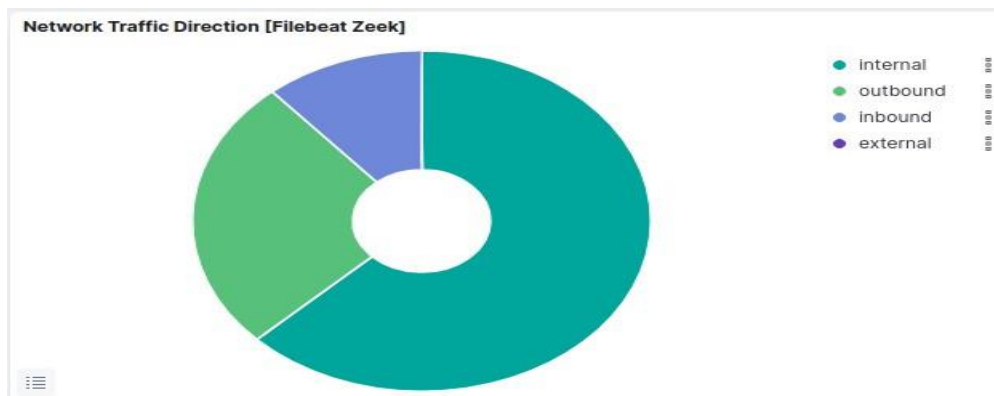


Figure 46 : Network traffic direction

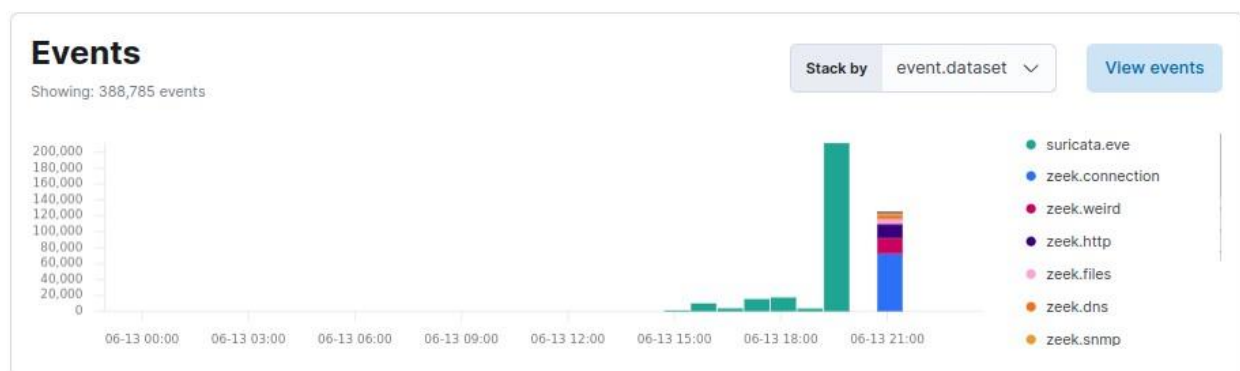


Figure 47 : Zeek events

Ici Zeek nous dresse la liste des domaines avec lesquels notre host a connecté :

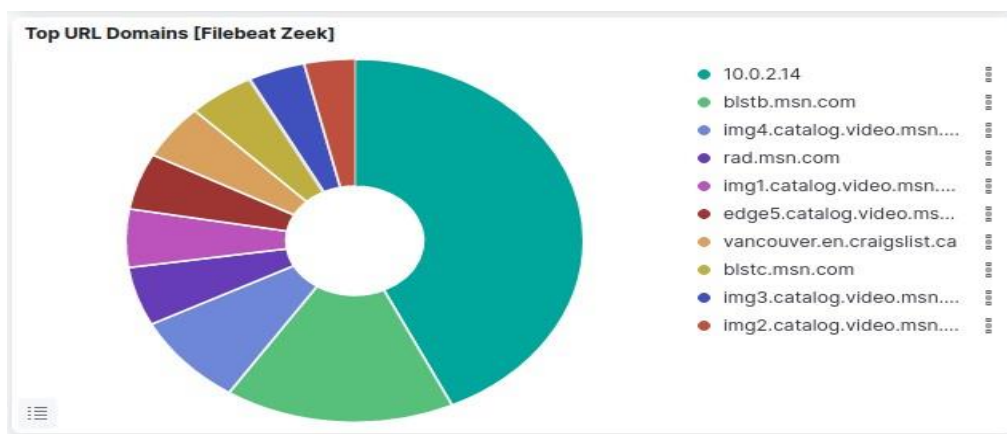


Figure 48 : TOP URL domains

# Conclusion générale

Le travail qui a été réalisé durant ce projet sort du commun. En effet, il s'agissait de travailler sur un sujet de construction d'un système de détection des intrusions réseaux qui regroupe l'usage de Suricata et Zeek avec la pile ELK.

Suricata qui fait le rôle de détection d'intrusion, et de supervision de sécurité réseau. Zeek qui offre une analyse réseau en temps réel et permet de garantir encore mieux la pérennité du réseau. Elasticsearch extrait les données, Logstash normalise les données temporelles et Kibana est un outil de visualisation.

On a réalisé deux cas, le premier dont lequel on a simulé une attaque de bruteforcing SSH en basant sur les logs générées par syslog. Et le deuxième, on a simulé une attaque externe à l'aide de *TCP Replay*.

Ce projet était une chance pour nous pour savoir comment établir un IDS à zéro sur notre réseau local afin de détecter toutes les anomalies qui prouvent passer sur notre réseau.

# Webographie

- [1] Wikipédia : École Nationale des Sciences Appliquées d'Oujda
- [2] <https://www.geeksforgeeks.org/intrusion-detection-system-ids>
- [3] <https://www.checkpoint.com/cyber-hub/network-security/what-is-an-intrusion-detection-system-ids/ids-vs-ips>
- [4] <https://www.spiceworks.com/it-security/network-security/articles/ids-vs-ips>
- [5] <https://serverspace.io/support/help/how-to-install-suricata-on-ubuntu-20-04>
- [6] <https://docs.zeeq.org/en/master/install.html>
- [7] <https://www.elastic.co/fr/what-is/elasticsearch>
- [8] <https://www.elastic.co/fr/what-is/logstash>
- [9] <https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-overview.html>
- [10] <https://www.elastic.co/guide/en/elasticsearch/reference/current/deb.html>
- [11] <https://www.elastic.co/guide/en/logstash/current/installing-logstash.html>
- [12] <https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-installation-configuration.html>
- [13] <https://www.elastic.co/guide/en/kibana/current/install.html>
- [14] <https://tcpreplay.appneta.com/wiki/tcpreplay-man.html>