Université de Paris

Mini Projet

Network Monitoring (groupe : 9)
Pcap : groupe-9-19.pcap

Présenté par :
Amar HENNI
Lounes Kheris

Année universitaire : 2021/2022

Introduction

Dans ce projet nous avons analysé le contenu d'un trafic réseau depuis un fichier PACAP **groupe-9-19.pcap** que vous nous avez fournis, dans un premier temps, on s'intéresse à faire des statistiques sur ce fichier, par le suite on s'intéressera à la géneration d'alertes en utilisant **Suricata** et enfin on va répondre à la partie questions du projet.

Outils utilisés

— Wireshark : Pour avoir une première vue sur le fichier .pcap et il nous à permis aussi d'avoir des informations pour cibler nos propres alertes crée avec Suricata
— Langage de programmation : Python
— Librairie : scapy , matplotlib , pyx, numpy, pyvis.network, networkx, collections, nfstream, socket, pandas

Partie 1 : Prétraitement des données et résultats

Dans cette première section nous allons interpréter les résultats obtenus lors de la partie du prétraitement du fichier PCAP fournit **(groupe-9-19.pcap)**

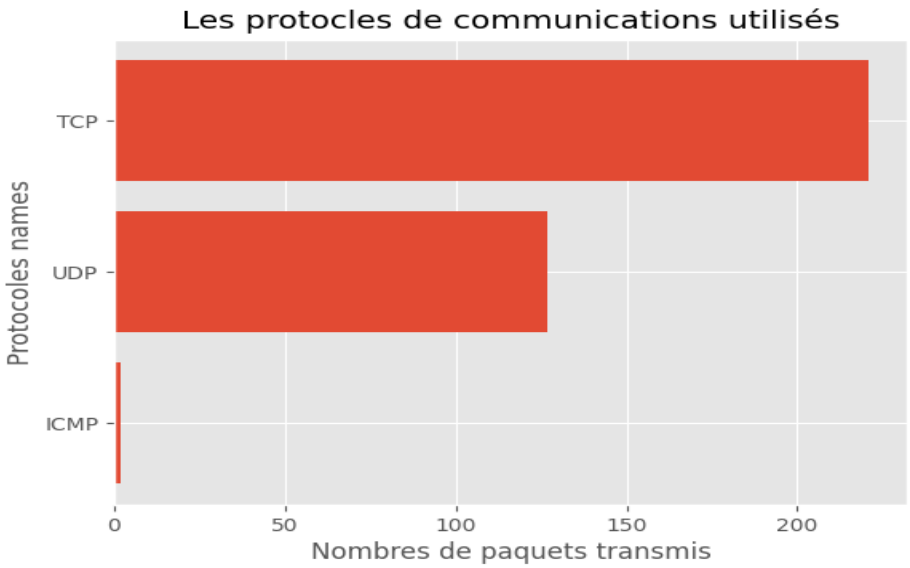
Avant de commencer l'analyse en profondeur de notre fichier PCAP, on a d'abord affiché le nombre de paquets présents au niveau de ce derniers :

```
amar@ufrinfo-Latitude-5490:~/M2-DATA/Ing-Protocoles/MiniProjet$ /t

 --- Debut de l'analyse du fichier pcap ---

Le nombre de paquets présents dans le fichier est :  342  Paquets
```

On a également affiché les informations concernant les protocoles utilisés (protocole de niveau application et protocole de communication) on a eu les résultats suivant :

Protocoles de niveau communication

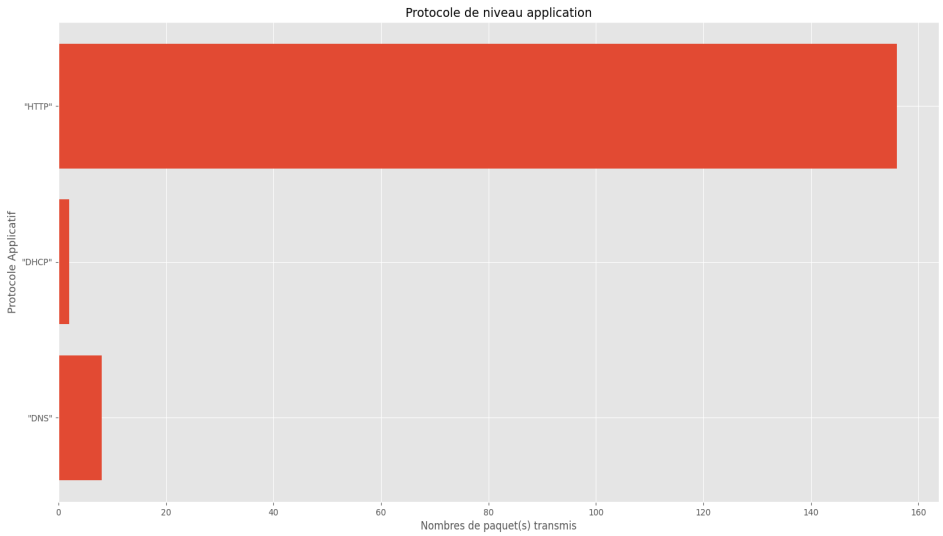
```
1) Les noms des protocoles de communications présent dans le fichier
Protocole :  ICMP  ==>  2  paquet(s)
Protocole :  UDP  ==>  127  paquet(s)
Protocole :  TCP  ==>  221  paquet(s)
```



On remarque qu'au niveau des protocoles de communication, le protocole TCP est le plus utilisé sur 221 paquets ensuite viens le protocole UDP qui est utilsé pour le transferts de 127 paquets.

Remarque : *Le protocole ICMP est également présent au niveau des protocoles de communications or ce derniers est un protocole de niveau 3* ***(Réseau)****, donc on pense que* ***NFStream*** *classe tout ce qui vient juste après IP en tant que protocoles de couche de transport et cela represente une erreur de la part de NFStream. Ce dernier fait donc une erreur lors de la classification du protocole.*

Protocole de niveau Applicatication



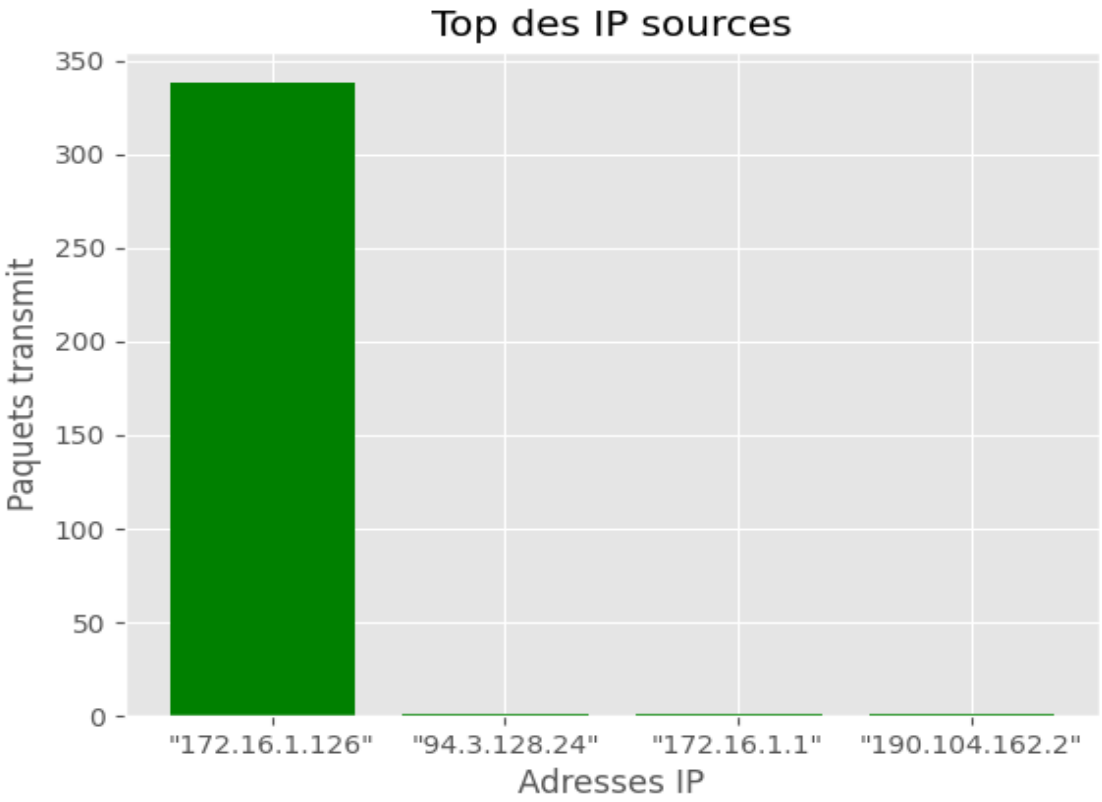
On remarque que 3 protocoles sont présents niveau application :

— HTTP avec 156 paquets.
— DNS avec 8 paquets
— DHCP avec 2 paquets

Le trafic HTTP est plus répondu et plus utilisé.

Maintenant qu'ont sait quels protocoles sont utilisé dans notre communication on va s'intéressé à quelles sont les adresses IP qui sont responsables de la géneration de ce trafic réseaux. Pour cela on a réaliser 2 affichages possible . . .un affichage en mode console et un affichage graphique plus représentatif comme suit :

```
5) Les top ip sources  dans le fichier pcap ( voir le plot pour mieux visualiser) :

IP source :  "172.16.1.126"  ==>  338  paquet(s)
IP source :  "94.3.128.24"  ==>  1  paquet(s)
IP source :  "172.16.1.1"  ==>  1  paquet(s)
IP source :  "190.104.162.2"  ==>  1  paquet(s)
```



On remarque donc qu'il y'a 4 adresses IP Sources, ce qui veux dire 4 machines Sources qui initient la communication :

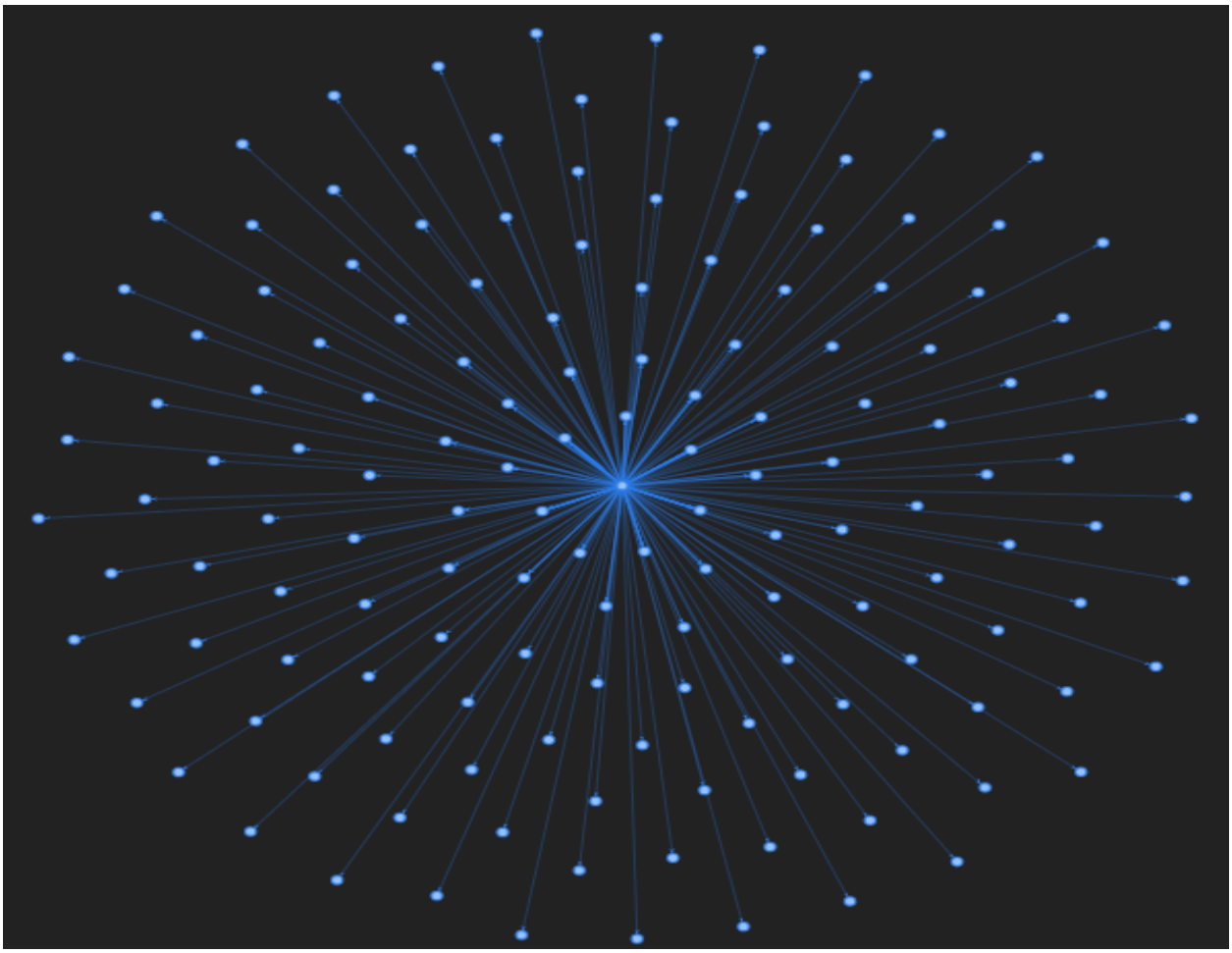
— Machine 1 : **172.16.1.126**
— Machine 2 : **94.3.128.24**
— Machine 3 : **172.16.1.1**
— Machine 4 : **190.104.162.2**

On regardant attentivement le graphique ainsi que l'affichage console on peut en déduire qu'il y'a une machine qui envoie beaucoup de requêtes et c'est la machine avec l'adresse IP : **172.16.1.126** avec un total de 338 paquets envoyés. Et les 3 autres machines envoie chaqu'une 1 paquet.

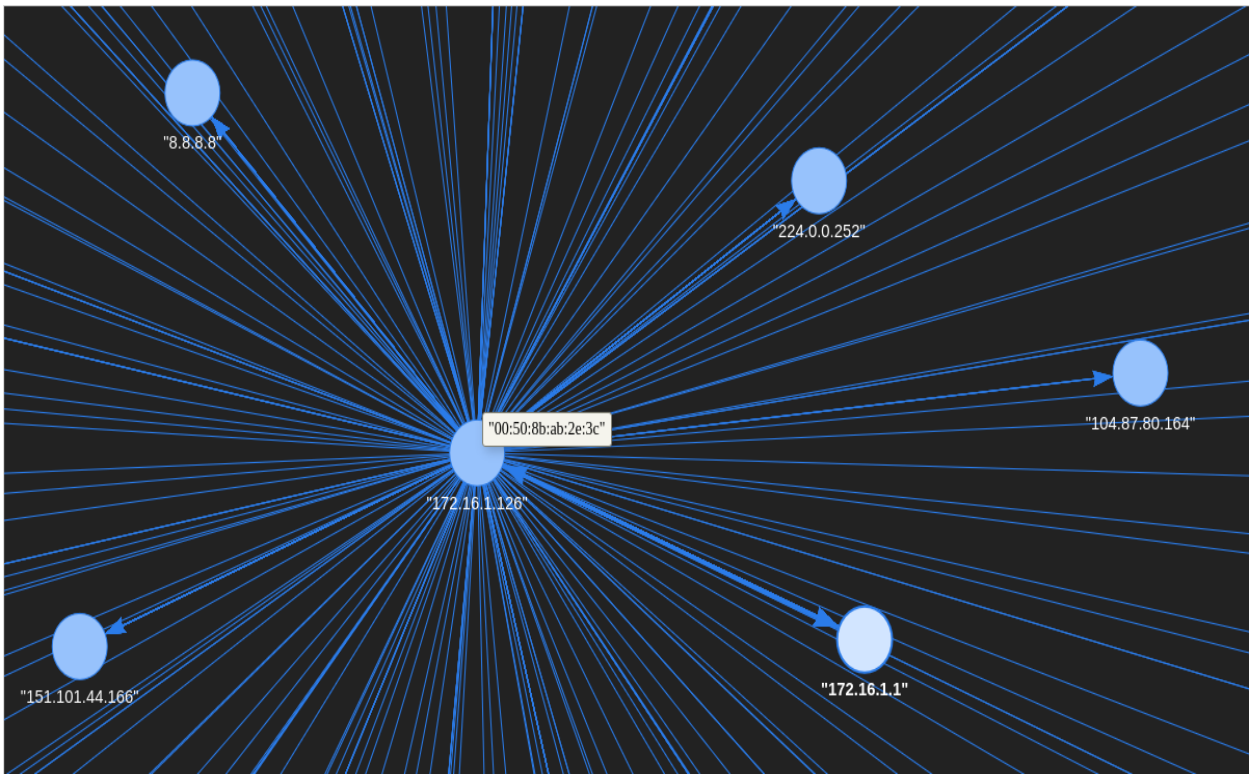
Afin de représenter au mieux les liens de communications ainsi que les adresses IP intérogées par ces machines, dans un premier temps on a fait un affichage en console qui met en évidences les différentes adresses Sources et leurs Distinations :

```
 3) Les IP qui envoi ou reçoit des données ainsi que leurs adresses mac (voir le plot pour mieux visualiser)
"172.16.1.126" ------> "255.255.255.255"
"172.16.1.126" ------> "224.0.0.252"
"172.16.1.126" ------> "172.16.1.1"
"172.16.1.126" ------> "23.218.156.26"
"172.16.1.126" ------> "224.0.0.252"
"172.16.1.126" ------> "224.0.0.252"
"172.16.1.126" ------> "184.107.174.122"
"172.16.1.126" ------> "172.16.1.1"
"172.16.1.126" ------> "172.16.1.1"
"172.16.1.126" ------> "172.16.1.1"
"172.16.1.126" ------> "224.0.0.252"
"172.16.1.126" ------> "185.118.67.195"
"172.16.1.126" ------> "136.243.110.83"
"172.16.1.126" ------> "172.16.1.1"
```

Et afin de mettre mieux en évidence les communications on a réaliser un graphe où les noeuds représentes les adresses IP et les arcs orientés le sens de communication. **(Lors du survole des noeuds via la souris l'adresse MAC des ces derniers sera visible)** :



On regardant de plus prés les noeuds et les liens :

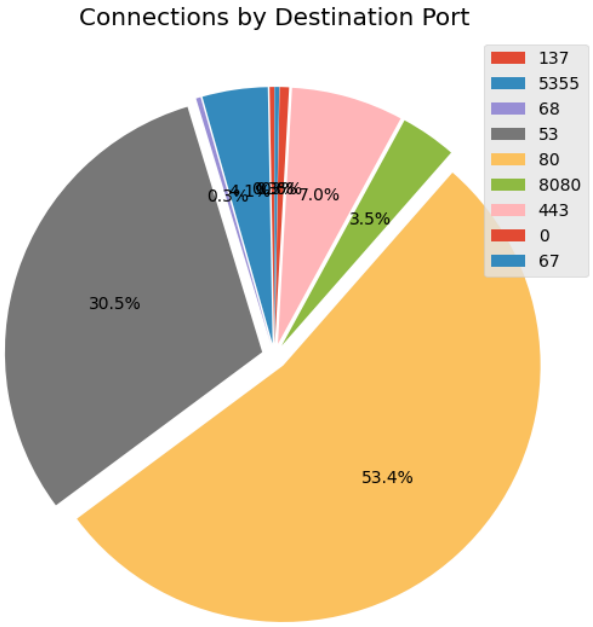


On remarque par exemple que la machine (IP : 172.16.1.1) envoie une requête à la machine (IP : 172.16.1.126) et vice-versa.

On remarque également que la machine (IP : 172.16.1.126) envoie une requête à la machine (IP : 224.0.0.252) par exemple.

On peut également oservé les adresses MAC des différentes machines. Par exemple la machine avec l'IP 172.16.1.126 a comme adresse **MAC : 00 :50 :8b :ab :2e :3c**

Afin d'observé au mieux les liens des connections, on a affiché les différents port de connection (Port des machines de destinations) afin de savoir quels types de trafic réseaux passaient par ces connections et donc quels types de flux était en transit dans le réseau. On a obtenu le graphe suivant :

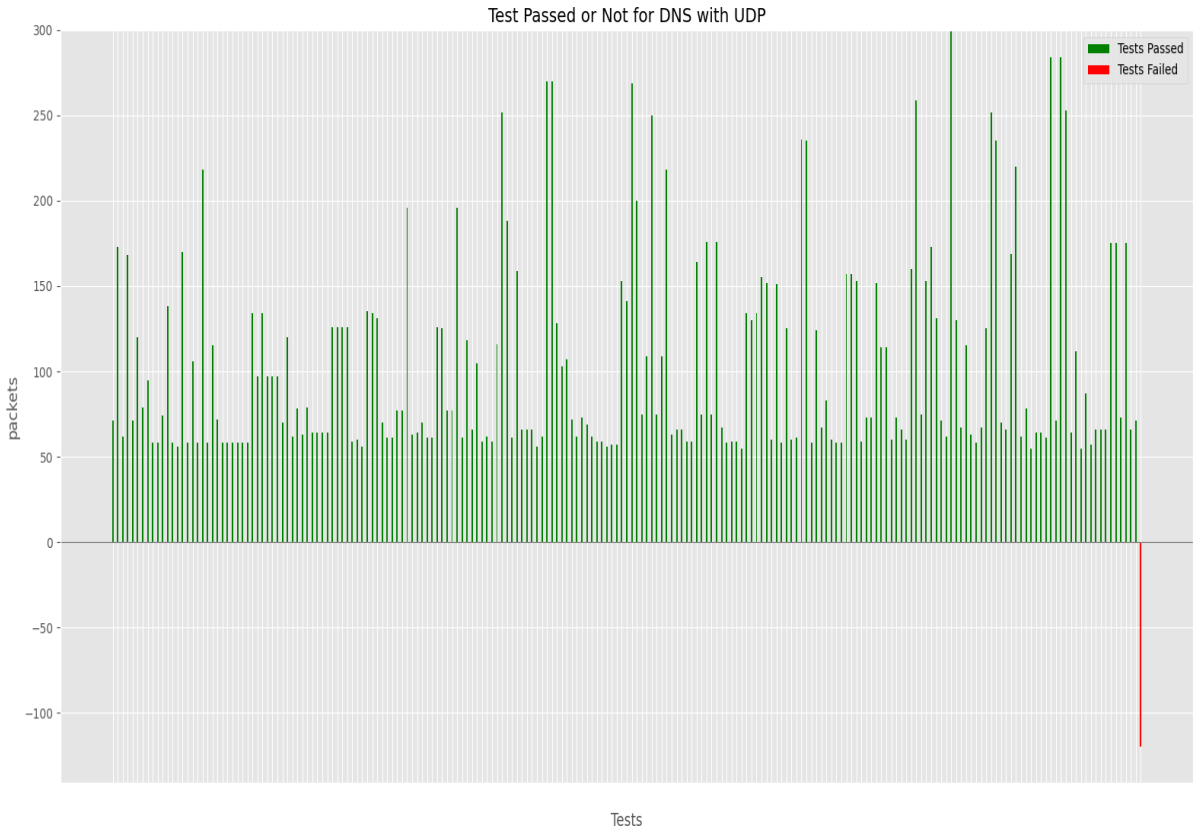


On analysant le graphe on remarque que les ports de destinnations sont classés comme suit (suivant leurs taux de stimulation par un ordre décroissant) :

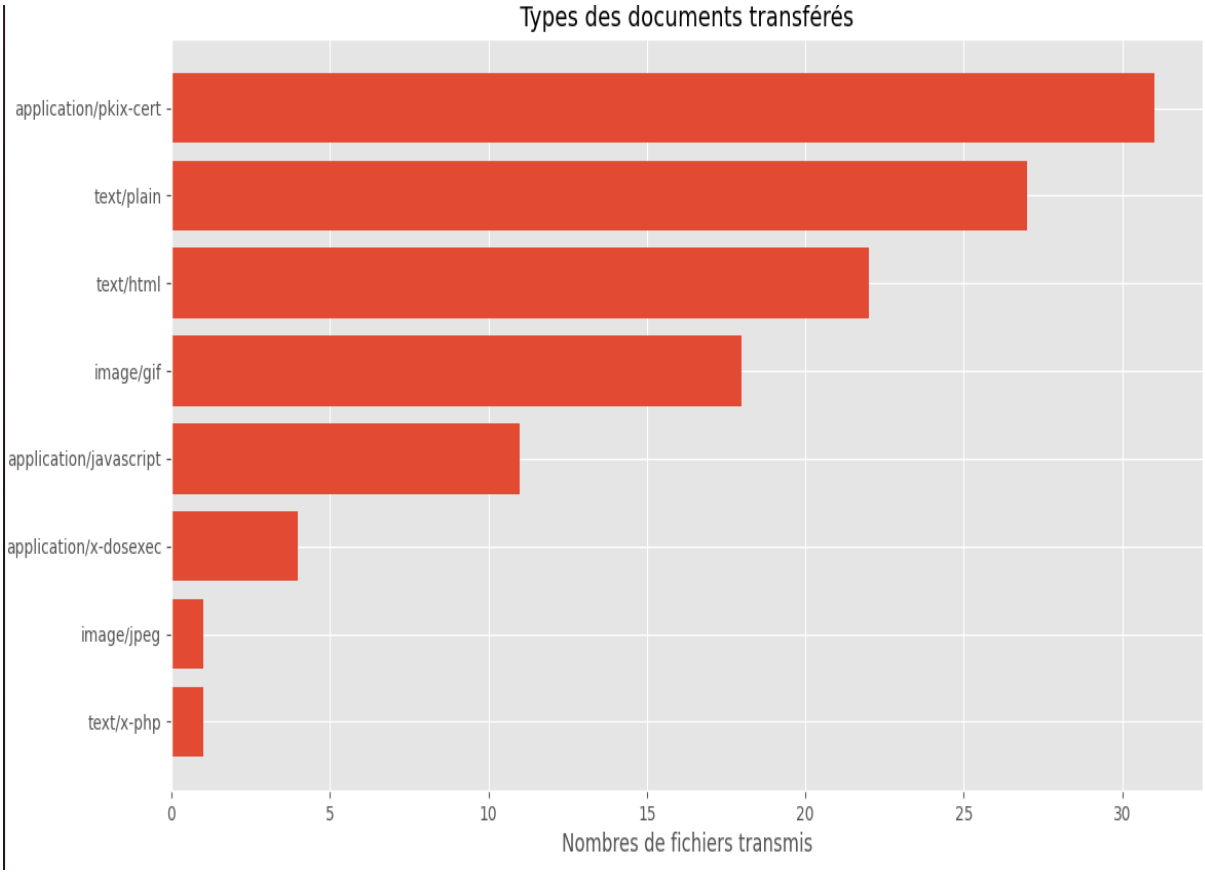
— Port 80 : avec 184 connections
— Port 53 : avec 104 connections
— Port 443 : avec 24 connections
— Port 5355 : avec 14 connections
— Port 8080 : avec 13 connections
— Port 67 : avec 6 connections
— Port 137 : avec 3 connections
— Port 68 : avec 1 connections
— Port 0 : avec 1 connections

Durant l'analyse du fichier PCAP on a voulu vérifié est-ce que tous les paquets DNS générés au cours de la communication respectent-ils la norme d'être inferieur à **512 octects**. Car Si une réponse dépasse cette taille, la norme prévoit que la requête doit être renvoyée sur le port **TCP 53**. Ce cas est cependant rare et évité .

Après le test on a remarqué que tout les paquets DNS respectaient la norme. Afin de voir se qui se passerait si on avait un paquet qui respecte pas la norme, on a fait une simulation à la fin en donnant une taille de paquet plus grande que 512 à notre fonction qui s'occupe du plot, dans notre implémentation on a dis que si un paquet dépasse 512 octect il sera vu comme une valeur négatif d'ou le résultat suivant :



On a également analyser les types de fichiers transferés, on a obtenu les résultats suivants :



Ce graphique affiche donc les **MIME type** qui sont les *Multipurpose Internet Mail Extensions*, c'est un standard permettant d'indiquer la nature et le format d'un document. Il faut savoir que les navigateurs utilisent le plus souvent le type MIME et non l'extension d'un fichier pour déterminer la façon dont ils vont traiter ou afficher un document. Il est donc important que les serveurs puissent correctement attacher le type MIME dans l'en-tête de la réponse qu'ils renvoient.

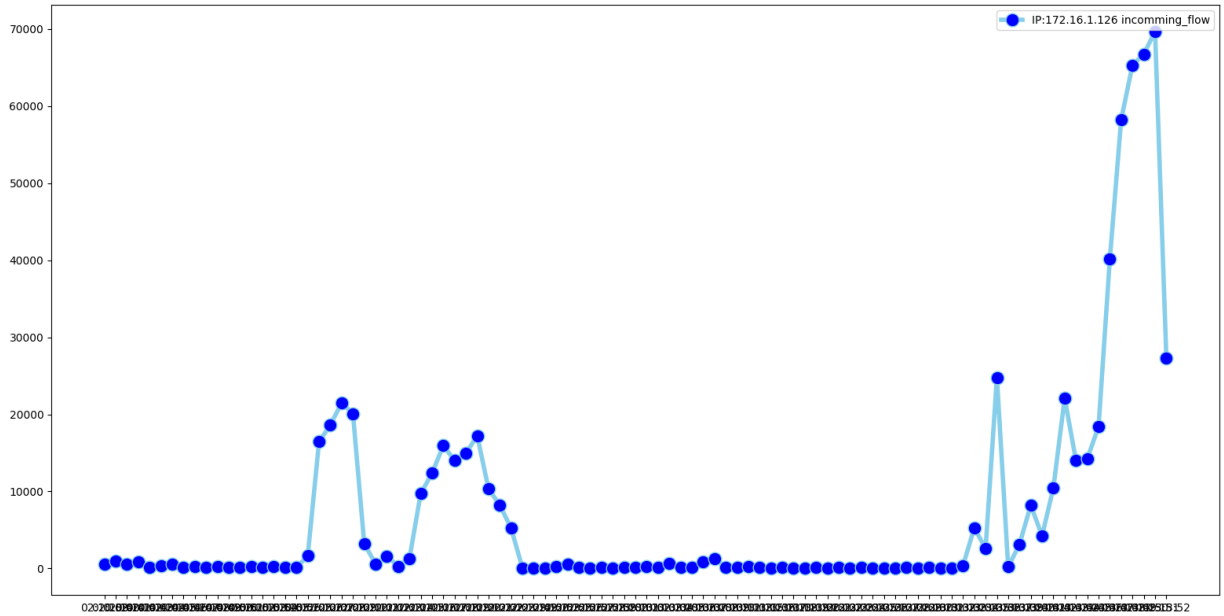
Les MIME Type sont sous forme : **type/sous-type**

Notre plot nous permet donc de visualisé ces MIME Type :

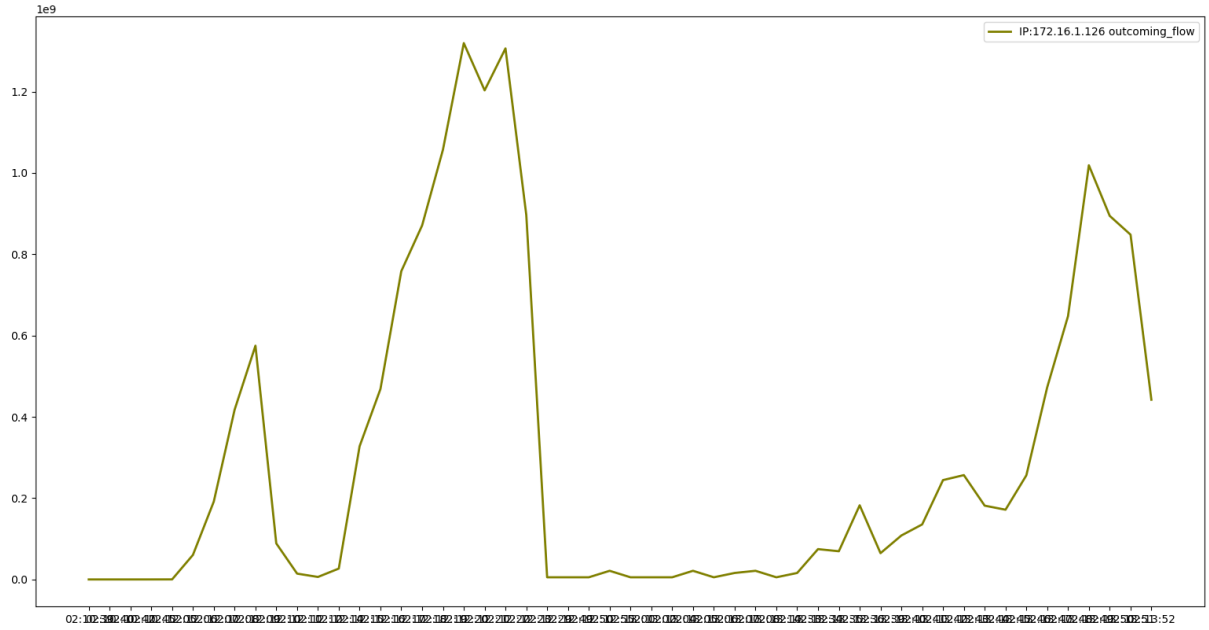
— application/pkix-cert avec 32 fichiers
— text/plain avec 27 fichiers
— text/html avec 22 fichiers
— image/gif avec 18 ficiers
— application/javascript avec 11 fichiers
— application/x-dosexec avec 4 fichiers
— image/jpeg avec 1 fichier
— text/x-php avec 1 fichier

Pour terminé cette analyse du fichier PCAP on va fournir un graphique exposant les flows de données transmit et reçu de/et vers l'hôte avec IP : 172.16.1.126 :

Incoming flow



Outcoming flow



Remarque : Pour réaliser ces 2 graphiques liée au flow de données on a procéder en respectant les étapes suivante tout en utilisant Xireshark afin d'avoir toutes les informations necessaires à notre traitement :

— 1- Choisir une adresse IP pertinente avec laquelle travailler. On a choise pour cela l'adresse IP qui contient plus de paquets, donc qui a plus de trafic (IP : 172.16.1.126)

— Faire la somme de la taille (en bytes) des paquets que cette IP envoie chaque seconde. Ce qui nous donne l'incoming traffic volumen

— Faire la somme de la taille (en bytes) des paquets que cette IP reçoit chaque seconde. Ce qui nous donne l'outgoing traffic volumen

— Réalisation du graphique Incoming traffic volumen (bytes) vs Time (sec)

— Réalisation du graphique Outgoing traffic volumen (bytes) vs Time (sec)

Partie 2 : Effectuer une analyse de sécurité avec Suricata

La version de suricata utilisé est :

```
amar@ufrinfo-Latitude-5490:~/M2-DATA/Ing-Protocoles/MiniProjet/suricata$ suricata -V
This is Suricata version 6.0.4 RELEASE
```

Avant de commencer à analyser notre fichier PACAP et générer des alertes on doit d'abord s'assurer que Suricata est bien configuré, pour ce faire on a exécuter 2 scripts fait par nous mêmes en partie (trouvé sur le net qu'ont a adapté à nos besoins) :

— Le script **setup_suricata-update.sh** : qui nous permet de crée un groupe pour suricata et changé les différentes autorisations des différents répertoires de suricata comme la documentation l'indique et à la fin on ajoute notre utilisateur au groupe suricata crée afin de bien avoir accés aux fichiers générés par Suricata et qu'ont aient pas de problème lors de l'execution.

— Le scipt **suri-ingest-pcap.sh** : Ce fichier interviens pour dire comment Suricata dois se comporté en cas d'absensse de jeu de régles, car par defaut il est configuré pour rechercher une régle dans un emplacement spécifique et si nous n'avons pas encore télécharger le jeu de régles, il est probable que cela va générer une erreur. Et se script rend plus facile la manière comment traiter un fichier pcap et cela dans le but de générer nos alertes.

Aprés l'execution de ces fichiers on a les erreurs suivantes :

```
amar@ufrinfo-Latitude-5490:~/M2-DATA/Ing-Protocoles/MiniProjet/suricata$ sudo ./suri-ingest-pcap.sh group-9-19.pcap
19/12/2021 -- 13:39:19 - <Notice> - This is Suricata version 6.0.4 RELEASE running in USER mode
19/12/2021 -- 13:39:19 - <Error> - [ERRCODE: SC_ERR_FOPEN(44)] - Error opening file: "/var/log/suricata//fast.log": Permission denied
19/12/2021 -- 13:39:19 - <Warning> - [ERRCODE: SC_ERR_INVALID_ARGUMENT(13)] - output module "fast": setup failed
19/12/2021 -- 13:39:19 - <Error> - [ERRCODE: SC_ERR_FOPEN(44)] - Error opening file: "/var/log/suricata//eve.json": Permission denied
19/12/2021 -- 13:39:19 - <Warning> - [ERRCODE: SC_ERR_INVALID_ARGUMENT(13)] - output module "eve-log": setup failed
19/12/2021 -- 13:39:19 - <Error> - [ERRCODE: SC_ERR_FOPEN(44)] - Error opening file: "/var/log/suricata//stats.log": Permission denied
19/12/2021 -- 13:39:19 - <Warning> - [ERRCODE: SC_ERR_INVALID_ARGUMENT(13)] - output module "stats": setup failed
19/12/2021 -- 13:39:19 - <Warning> - [ERRCODE: SC_WARN_NO_STATS_LOGGERS(261)] - stats are enabled but no loggers are active
19/12/2021 -- 13:39:19 - <Warning> - [ERRCODE: SC_ERR_NO_RULES(42)] - No rule files match the pattern /var/lib/suricata/rules/suricata.rules
19/12/2021 -- 13:39:19 - <Warning> - [ERRCODE: SC_ERR_NO_RULES_LOADED(43)] - 1 rule files specified, but no rules were loaded!
19/12/2021 -- 13:39:19 - <Notice> - all 9 packet processing threads, 2 management threads initialized, engine started.
19/12/2021 -- 13:39:19 - <Notice> - Signal Received.  Stopping engine.
19/12/2021 -- 13:39:19 - <Notice> - Pcap-file module read 1 files, 9130 packets, 6254270 bytes

Alerts:
```

Une erreur dans notre script à été révélé au niveau du chemin **/var/log/suricata/fast.log** et qu'ont a réglé, et une autre erreur est présente où on nous dit qu'aucun fichier de règles ne correspond au pattern, donc un ficier de régle a été spécifié mais aucune règes n'a été chargée, c'est donc un problème. Pour régler ce problème il fallait taper la commande :
sudo suricata update

```
19/12/2021 -- 21:58:22 - <Info> -- Loaded 31636 rules.
19/12/2021 -- 21:58:22 - <Info> -- Disabled 14 rules.
19/12/2021 -- 21:58:22 - <Info> -- Enabled 0 rules.
19/12/2021 -- 21:58:22 - <Info> -- Modified 0 rules.
19/12/2021 -- 21:58:22 - <Info> -- Dropped 0 rules.
19/12/2021 -- 21:58:23 - <Info> -- Enabled 131 rules for flowbit dependencies.
19/12/2021 -- 21:58:23 - <Info> -- Backing up current rules.
19/12/2021 -- 21:58:24 - <Info> -- Writing rules to /var/lib/suricata/rules/suricata.rules: total: 31636; enabled: 24262; added: 0; removed 0;
 modified: 0
```

On obtient alors des informations sur la façon dont suricata traite les différentes sources et régles, ainsi que le nombre totale de règles qui ont été écrites (31636) et activé (24262).

Aprés l'execution des mise à jour on peut revenir a l'analyse de notre fichier PCAP, on remarque que l'execution prend un peu plus de temps et que nous avons des aletres qui aparaissent et qui confirmeront que tout semble être correctement configuré.

```
amar@ufrinfo-Latitude-5490:~/M2-DATA/Ing-Protocoles/MiniProjet/suricata/pcaps$ sudo ./suri-ingest-pcap.sh  group-9-19.pcap
19/12/2021 -- 22:09:46 - <Notice> - This is Suricata version 6.0.4 RELEASE running in USER mode
19/12/2021 -- 22:10:21 - <Notice> - all 9 packet processing threads, 4 management threads initialized, engine started.
19/12/2021 -- 22:10:21 - <Notice> - Signal Received.  Stopping engine.
19/12/2021 -- 22:10:21 - <Notice> - Pcap-file module read 1 files, 9130 packets, 6254270 bytes

Alerts:
```

On remarque qu'aucune alerte n'a été soulevé. On ajoute donc toutes les sources par défault :

```
amar@ufrinfo-Latitude-5490:~/M2-DATA/Ing-Protocoles/MiniProjet/suricata/pcaps$ suricata-update list-enabled-sources
19/12/2021 -- 22:41:25 - <Info> -- Using data-directory /var/lib/suricata.
19/12/2021 -- 22:41:25 - <Info> -- Using Suricata configuration /etc/suricata/suricata.yaml
19/12/2021 -- 22:41:25 - <Info> -- Using /etc/suricata/rules for Suricata provided rules.
19/12/2021 -- 22:41:25 - <Info> -- Found Suricata version 6.0.4 at /usr/bin/suricata.
Enabled sources:
  - malsilo/win-malware
  - ptresearch/attackdetection
  - et/open
  - tgreen/hunting
  - sslbl/ssl-fp-blacklist
  - oisf/trafficid
  - etnetera/aggressive
  - sslbl/ja3-fingerprints
```

On remarque maintenant après la mise à jour avec **suricata update** les ajouts de règles correspondant à ces sources par défault (5871 régles ajoutés) :

```
19/12/2021 -- 22:44:32 - <Info> -- Writing rules to /var/lib/suricata/rules/suricata.rules: total: 37507; enabled: 29911; added: 5871; removed 0; modified: 0
```

Une fois que toutes les rèles sont ajoutées, on scan notre fichier pcap en utilisant le script précèdent **suri-ingest-pcap.sh** et on a les alertes suivantes :

```
amar@ufrinfo-Latitude-5490:~/M2-DATA/Ing-Protocoles/MiniProjet/suricata/pcaps$ sudo ./suri-ingest-pcap.sh  group-9-19.pcap
20/12/2021 -- 00:32:15 - <Notice> - This is Suricata version 6.0.4 RELEASE running in USER mode
20/12/2021 -- 00:32:50 - <Notice> - all 9 packet processing threads, 4 management threads initialized, engine started.
20/12/2021 -- 00:32:50 - <Notice> - Signal Received.  Stopping engine.
20/12/2021 -- 00:32:50 - <Notice> - Pcap-file module read 1 files, 9130 packets, 6254270 bytes

Alerts:

"2016-07-06T02:12:05.848856+0200 | 1:2022692:4 | ET MALWARE JS/Nemucod requesting EXE payload 2016-03-31 | A Network Trojan was detected | 172.16.1.126:49158 -> 184.107.174.122:80"
"2016-07-06T02:12:05.848856+0200 | 1:2024035:5 | ET MALWARE WS/JS Downloader Mar 07 2017 M1 | A Network Trojan was detected | 172.16.1.126:49158 -> 184.107.174.122:80"
"2016-07-06T02:12:06.166349+0200 | 1:2018959:4 | ET POLICY PE EXE or DLL Windows file download HTTP | Potential Corporate Privacy Violation | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:06.166349+0200 | 1:2021954:2 | ET MALWARE JS/Nemucod.M.gen downloading EXE payload | A Network Trojan was detected | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:06.166349+0200 | 1:2022053:2 | ET MALWARE Likely Evil EXE download from MSXMLHTTP non-exe extension M2 | A Network Trojan was detected | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:06.166349+0200 | 1:2023671:5 | ET MALWARE JS/WSF Downloader Dec 08 2016 M3 | A Network Trojan was detected | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:06.166349+0200 | 1:2023672:5 | ET MALWARE JS/WSF Downloader Dec 08 2016 M4 | A Network Trojan was detected | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:06.166349+0200 | 1:2014520:7 | ET INFO EXE - Served Attached HTTP | Misc activity | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:14.762492+0200 | 1:2018582:12 | ET ADWARE_PUP Miuref/Boaxxe CheckIn | Possibly Unwanted Program Detected | 172.16.1.126:49160 -> 185.118.67.195:80"
"2016-07-06T02:12:06.772071+0200 | 1:2022692:4 | ET MALWARE JS/Nemucod requesting EXE payload 2016-03-31 | A Network Trojan was detected | 172.16.1.126:49158 -> 184.107.174.122:80"
"2016-07-06T02:12:06.772071+0200 | 1:2024035:5 | ET MALWARE WS/JS Downloader Mar 07 2017 M1 | A Network Trojan was detected | 172.16.1.126:49158 -> 184.107.174.122:80"
"2016-07-06T02:12:06.864929+0200 | 1:2021954:2 | ET MALWARE JS/Nemucod.M.gen downloading EXE payload | A Network Trojan was detected | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:06.864929+0200 | 1:2022053:2 | ET MALWARE Likely Evil EXE download from MSXMLHTTP non-exe extension M2 | A Network Trojan was detected | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:06.864929+0200 | 1:2023671:5 | ET MALWARE JS/WSF Downloader Dec 08 2016 M3 | A Network Trojan was detected | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:06.864929+0200 | 1:2023672:5 | ET MALWARE JS/WSF Downloader Dec 08 2016 M4 | A Network Trojan was detected | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:06.864929+0200 | 1:2014520:7 | ET INFO EXE - Served Attached HTTP | Misc activity | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:07.140154+0200 | 1:2022692:4 | ET MALWARE JS/Nemucod requesting EXE payload 2016-03-31 | A Network Trojan was detected | 172.16.1.126:49158 -> 184.107.174.122:80"
"2016-07-06T02:12:07.140154+0200 | 1:2024035:5 | ET MALWARE WS/JS Downloader Mar 07 2017 M1 | A Network Trojan was detected | 172.16.1.126:49158 -> 184.107.174.122:80"
"2016-07-06T02:12:07.231197+0200 | 1:2021954:2 | ET MALWARE JS/Nemucod.M.gen downloading EXE payload | A Network Trojan was detected | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:07.231197+0200 | 1:2022053:2 | ET MALWARE Likely Evil EXE download from MSXMLHTTP non-exe extension M2 | A Network Trojan was detected | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:07.231197+0200 | 1:2023671:5 | ET MALWARE JS/WSF Downloader Dec 08 2016 M3 | A Network Trojan was detected | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:07.231197+0200 | 1:2023672:5 | ET MALWARE JS/WSF Downloader Dec 08 2016 M4 | A Network Trojan was detected | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:07.231197+0200 | 1:2014520:7 | ET INFO EXE - Served Attached HTTP | Misc activity | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:07.339562+0200 | 1:2022692:4 | ET MALWARE JS/Nemucod requesting EXE payload 2016-03-31 | A Network Trojan was detected | 172.16.1.126:49158 -> 184.107.174.122:80"
"2016-07-06T02:12:07.339562+0200 | 1:2024035:5 | ET MALWARE WS/JS Downloader Mar 07 2017 M1 | A Network Trojan was detected | 172.16.1.126:49158 -> 184.107.174.122:80"
"2016-07-06T02:12:07.414761+0200 | 1:2021954:2 | ET MALWARE JS/Nemucod.M.gen downloading EXE payload | A Network Trojan was detected | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:07.414761+0200 | 1:2022053:2 | ET MALWARE Likely Evil EXE download from MSXMLHTTP non-exe extension M2 | A Network Trojan was detected | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:07.414761+0200 | 1:2023671:5 | ET MALWARE JS/WSF Downloader Dec 08 2016 M3 | A Network Trojan was detected | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:07.414761+0200 | 1:2023672:5 | ET MALWARE JS/WSF Downloader Dec 08 2016 M4 | A Network Trojan was detected | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:07.414761+0200 | 1:2014520:7 | ET INFO EXE - Served Attached HTTP | Misc activity | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:10.240569+0200 | 1:2610133:1 | TGI HUNT PHP magic bytes in HTTP response | Potentially Bad Traffic | 184.107.174.122:80 -> 172.16.1.126:49158"
"2016-07-06T02:12:10.240898+0200 | 1:2022692:4 | ET MALWARE JS/Nemucod requesting EXE payload 2016-03-31 | A Network Trojan was detected | 172.16.1.126:49158 -> 184.107.174.122:80"
"2016-07-06T02:12:10.240898+0200 | 1:2024035:5 | ET MALWARE WS/JS Downloader Mar 07 2017 M1 | A Network Trojan was detected | 172.16.1.126:49158 -> 184.107.174.122:80"
"2016-07-06T02:10:39.266619+0200 | 1:2018358:10 | ET HUNTING GENERIC SUSPICIOUS_POST to Dotted Quad with Fake Browser 1 | Potentially Bad Traffic | 172.16.1.126:49188 -> 90.149.107.130:80"
```

On remarque que nous avons un certain nombre d'alertes qui ont été générés. Pour chaque alertes on a :

— Des informations sur le timestamp
— L'adresse IP source
— L'adresse IP destination

— La signature (**Exemple** : *ET POLICY PE EXE or DLL Windows file download HTTP)*

On remarque qu'il y'a plusieurs type d'alertes :

— **MALWARE** (c'est le plus présent) Il correspond à un cheval de troie intercepté sur le traffic par exemple entre les deux hotes "172.16.1.126" et "184.107.174.122", on peut voir la règle de l'alerte définit dans le fichier **"suricata.rules"** correspond à :

```
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ET MALWARE Likely Evil EXE download from MSXMLHTTP non-exe extension
M2"; flow:established,to_client; file_data; content:"MZ"; within:2; byte_jump:4,58,relative,little; content:"PE|00 00|";
distance:-64; within:4; flowbits:isset,et.MS.XMLHTTP.no.exe.request; classtype:trojan-activity; sid:2022053; rev:2;
metadata:created_at 2015_11_09, former_category CURRENT_EVENTS, updated_at 2015_11_09;)
```

— **ET POLICY**
A notre avis, c'est essentiellement qu'il voit du trafic non http passer par le port 80, qui est réservé au trafic http.
Ceci dit, cela veux dire que de mauvais acteurs sont à l'intérieur du réseau, ils veulent utiliser des ports communs pour "exfiltrer" des données, en espérant qu'elles se mélangent au trafic légitime.

— **ET INFO EXE**
Au cours de la coopération entre les deux hotes, l'hote emmetrice lui fournit souvent diverses information complémentaires.

— **TGI HUNT**
Cette catégorie de règles englobe le trafic qui sort définitivement de l'ordinaire et qui est potentiellement le signe d'un système compromis. Les règles de réponse aux attaques entrent dans cette catégorie.

— **ET ADWARE_PUP**
on pense probablement que c'est un programme potentiellement indésirable ou un logiciel qui contient des logiciels publicitaires, installe des optimiseurs de système, modifie les paramètres de notre navigateur ou a d'autres objectifs peu clairs. Pour rajouter, cette phrase : "Potentially Unwanted Program" PUP (programme potentiellement indésirable) a été créé par McAfee pour éviter de qualifier les programmes téléchargeables de logiciels malveillants.

Creation de trois autres Alerts :

Les régles fournit par Suricata nous alertes en cas de trafic malicieux dans notre réseau. Pour cela il y'a des rules que l'ont peut implémenté gratuitement afin de garantir la sécurité de notre réseau mais il y'a également des rules payant et plus adapté à nos besoin. Cependant Suricata nous permet de crée nos propres régles (rules) afin de rechercher un comportement suspect dans notre réseau.

Dans un premier, on accéde au dossier source contenant l'ensemble régles :

```
lounes98kheris@hack98king:~$ cd /var/lib/suricata/rules/
lounes98kheris@hack98king:/var/lib/suricata/rules$ ls
classification.config  local.rules  suricata.rules
lounes98kheris@hack98king:/var/lib/suricata/rules$
```

on va ensuite créer notre fichier "local.rules" qui contiendra nos propres régles qu'on définira par la suite :

```
lounes98kheris@hack98king:/var/lib/suricata/rules$ sudo nano local.rules
lounes98kheris@hack98king:/var/lib/suricata/rules$ cat local.rules
# Our rules "HERE"
```

En suite, On doit mettre à jour notre fichier de configuration afin que notre "local.rules" soit inclu dans la liste des fichiers de régles de "suricata.yaml".

```
default-rule-path: /var/lib/suricata/rules

rule-files:
#default rules
  - suricata.rules
# - Custom Test Rules
  - local.rules
```

Remarque : *Les régles doivent être écrites sur une seul ligne dans les fichiers ".rules", Si on veux écrire une régles sur plusieurs lignes il faut ajouter un backslash à la fin de la ligne.*

Prémière alerte : Web Application Attack

Notre première régle consiste à prendre en considération le cas ou aprés une requête du client il n'y a aucune réponse du serveur (le serveur répond par défault par une erreur 404 Not Found)

```
alert http $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"Dinihoum Worm"; flow:to_server,established;
content:"POST";http_method; content:"is-ready";http_uri;endswith; content:"|3c 7c 3e|nan-av|3c 7c 3e|"; http_user_agent;
reference:url,threats.kaspersky.com/en/threat/Worm.VBS.Dinihou/; classtype:trojan-activity;sid:1000001;rev:1;)
```

notre type d'action est **alert** cela signifie que seul une alerte sera générée mais aucune autre activité supplémentaire ne sera bloquée, on a choisi d'utiliser ce type d'action pour qu'ont utilise suricata comme un système de détection d'intrusion et non de prévention. Ici on se concentre sur le trafic http de la couche application car comme on la vue précédament c'est le plus répondu de la couche application. Et on vérifié le trafic allant du réseaux local de n'importe quel port au réseau externe vers le port 81.

Dans notre alerte on vérifie la présence un cheval de troie (plus exactement Dinihoum Worm) et ci se dernier est présent on affiche une discription de message **"Dinihoum Worm"**.

Le mot clé content dans notre cas de rechercher un mot de publication dans la méthode http cela avec **content :"POST" ;http_method ;** puis on vérifie si l'uri de la demande se termine par le mot clé **endswith** qui signifie que c'est la dernière partie de l'uri. On vérifie ensuite que cette séquence de 3 octect **|3c 7c 3e|nan-av|3c 7c 3e|** est présente, car c'est une séquence qui doit être respectée dans le champs **user-agent** de la requête.

La partie référence de notre règle contient notre règle de métadonnées, c'est à

dire que reference elle contient le lien vers l'article avec la description de se ver (Worm).

> **Remarque :** *Toute les type de références possible peuvent être trouvés dans* ***/etc/suricata/reference.config***

Classtype signifie le type de la menace et sa gravité.

> **Remarque :** *Il y'a une liste des types de classes communs que l'ont peut trouvés dans* ***/etc/suricata/classification.config***, on peut également spécifier des types de classes personnalisés nous même

sid est un numéro de signature qui doit être unique.

rev est le diminutif de rules version ou révision, rev :1 signifie donc que cette règle n'a pas été modifiée depuis sa création

Deuxiéme alerte : Web Application Attack

```
alert http any any -> any any (msg:"Web Application Attack"; flow:to_server,established;
content:"GET"; http_method; content:"/sync/img?"; startswith; http_uri; classtype:web-application-attack;
sid:20204001; rev:1;)
```

```
Alerts:

"2016-07-06T02:13:48.317341+0200 | 1:20204001:1 | Web Application Attack | Web Application A
ttack | 172.16.1.126:49285 -> 74.121.142.105:80"
"2016-07-06T02:13:48.910144+0200 | 1:20204001:1 | Web Application Attack | Web Application A
ttack | 172.16.1.126:49285 -> 74.121.142.105:80"
"2016-07-06T02:13:50.735614+0200 | 1:20204001:1 | Web Application Attack | Web Application A
ttack | 172.16.1.126:49317 -> 74.121.142.105:80"
"2016-07-06T02:13:51.765042+0200 | 1:20204001:1 | Web Application Attack | Web Application A
ttack | 172.16.1.126:49317 -> 74.121.142.105:80"
lounes98kheris@hack98king:~/Desktop/Net-monitoring/Network_Monitoring-.suricata/pcap$
```

On pense que c'est une des attaques les plus connues et la plus courante contre les systémes d'authentification, il s'agit bien de l'attaue par force brute. Dans ce cas de figure, l'attauant bombarde la page d'authentification avec des valeurs pid et mdp jusqu'à ce qu'il obtienne l'accés à l'application.

Preuve de concept

```
GET /pull_sync?pid=contextweb HTTP/1.1
Host: www.wtp101.com
Connection: keep-alive
Accept: image/webp,image/*,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/51.0.2704.103 Safari/537.36
Referer: http://bh.contextweb.com/bh/visitormatch?tag=462601&pid=558667
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
```

Troisieme alerte : dns.query

```
alert dns any any -> any any (msg:"DNS LOOKUP for casalemedia"; dns.query;
content:"dsum.casalemedia.com"; sid:1000001;)
```

```
Alerts:

"2016-07-06T02:13:48.317341+0200 | 1:20204001:1 | Web Application Attack | Web Application Attack | 172.16.1
.126:49285 -> 74.121.142.105:80"
"2016-07-06T02:13:48.910144+0200 | 1:20204001:1 | Web Application Attack | Web Application Attack | 172.16.1
.126:49285 -> 74.121.142.105:80"
"2016-07-06T02:13:52.170257+0200 | 1:1000001:0 | DNS LOOKUP for casalemedia |  | 172.16.1.126:51848 -> 8.8.8
.8:53"
"2016-07-06T02:13:52.170268+0200 | 1:1000001:0 | DNS LOOKUP for casalemedia |  | 172.16.1.126:51849 -> 8.8.8
.8:53"
"2016-07-06T02:13:52.365450+0200 | 1:1000001:0 | DNS LOOKUP for casalemedia |  | 172.16.1.126:51851 -> 8.8.8
.8:53"
"2016-07-06T02:13:50.735614+0200 | 1:20204001:1 | Web Application Attack | Web Application Attack | 172.16.1
.126:49317 -> 74.121.142.105:80"
"2016-07-06T02:13:51.765042+0200 | 1:20204001:1 | Web Application Attack | Web Application Attack | 172.16.1
.126:49317 -> 74.121.142.105:80"
"2016-07-06T02:13:52.170273+0200 | 1:1000001:0 | DNS LOOKUP for casalemedia |  | 172.16.1.126:51850 -> 8.8.8
.8:53"
lounes98kheris@hack98king:~/Desktop/Net-monitoring/Network_Monitoring-.suricata/pcap$
```

Ici, la signature suivante examine le trafic DNS à la recherche de tout packet avec le contenu "casalemedia" et génére une alerte. On peut le voir clairement ci-dessusqu'il a capturé quatre packets .

Quatrième alerte : (ip) external-ip-check

```
alert ip any any -> any any (msg:"Device Retrieving External IP Address Detected";
classtype:external-ip-check ;sid:210054; rev:1;)
```

```
External IP Address Detected | 8.8.8.8:53 -> 172.16.1.126:51833"
"2016-07-06T02:13:49.674576+0200 | 1:210054:1 | Device Retrieving External IP Address Detected | Device Retrieving
External IP Address Detected | 104.87.100.224:443 -> 172.16.1.126:49337"
"2016-07-06T02:13:50.074610+0200 | 1:210054:1 | Device Retrieving External IP Address Detected | Device Retrieving
External IP Address Detected | 172.16.1.126:51834 -> 8.8.8.8:53"
"2016-07-06T02:13:50.115196+0200 | 1:210054:1 | Device Retrieving External IP Address Detected | Device Retrieving
External IP Address Detected | 8.8.8.8:53 -> 172.16.1.126:51834"
"2016-07-06T02:13:49.927074+0200 | 1:210054:1 | Device Retrieving External IP Address Detected | Device Retrieving
External IP Address Detected | 172.16.1.126:49345 -> 104.87.80.164:80"
"2016-07-06T02:13:49.957588+0200 | 1:210054:1 | Device Retrieving External IP Address Detected | Device Retrieving
External IP Address Detected | 104.87.80.164:80 -> 172.16.1.126:49345"
"2016-07-06T02:13:52.015647+0200 | 1:210054:1 | Device Retrieving External IP Address Detected | Device Retrieving
External IP Address Detected | 172.16.1.126:49375 -> 151.101.44.166:80"
"2016-07-06T02:13:52.069143+0200 | 1:210054:1 | Device Retrieving External IP Address Detected | Device Retrieving
External IP Address Detected | 151.101.44.166:80 -> 172.16.1.126:49375"
"2016-07-06T02:13:52.074473+0200 | 1:210054:1 | Device Retrieving External IP Address Detected | Device Retrieving
External IP Address Detected | 172.16.1.126:49378 -> 54.192.6.145:80"
"2016-07-06T02:13:52.106705+0200 | 1:210054:1 | Device Retrieving External IP Address Detected | Device Retrieving
External IP Address Detected | 54.192.6.145:80 -> 172.16.1.126:49378"
"2016-07-06T02:13:50.751391+0200 | 1:210054:1 | Device Retrieving External IP Address Detected | Device Retrieving
External IP Address Detected | 172.16.1.126:49358 -> 96.17.202.67:80"
"2016-07-06T02:13:51.419783+0200 | 1:210054:1 | Device Retrieving External IP Address Detected | Device Retrieving
External IP Address Detected | 172.16.1.126:49365 -> 64.233.169.156:443"
"2016-07-06T02:13:51.420030+0200 | 1:210054:1 | Device Retrieving External IP Address Detected | Device Retrieving
External IP Address Detected | 172.16.1.126:49368 -> 68.67.152.221:80"
"2016-07-06T02:13:51.469261+0200 | 1:210054:1 | Device Retrieving External IP Address Detected | Device Retrieving
External IP Address Detected | 64.233.169.156:443 -> 172.16.1.126:49365"
"2016-07-06T02:13:51.483746+0200 | 1:210054:1 | Device Retrieving External IP Address Detected | Device Retrieving
External IP Address Detected | 68.67.152.221:80 -> 172.16.1.126:49368"
```

Il s'agit de l'adresse ou des adresses qui nous sont fournies par le fournisseur d'accès Internet (FAI).

Ici vers toute la fin, on recharge nos nouvelles régles avec :

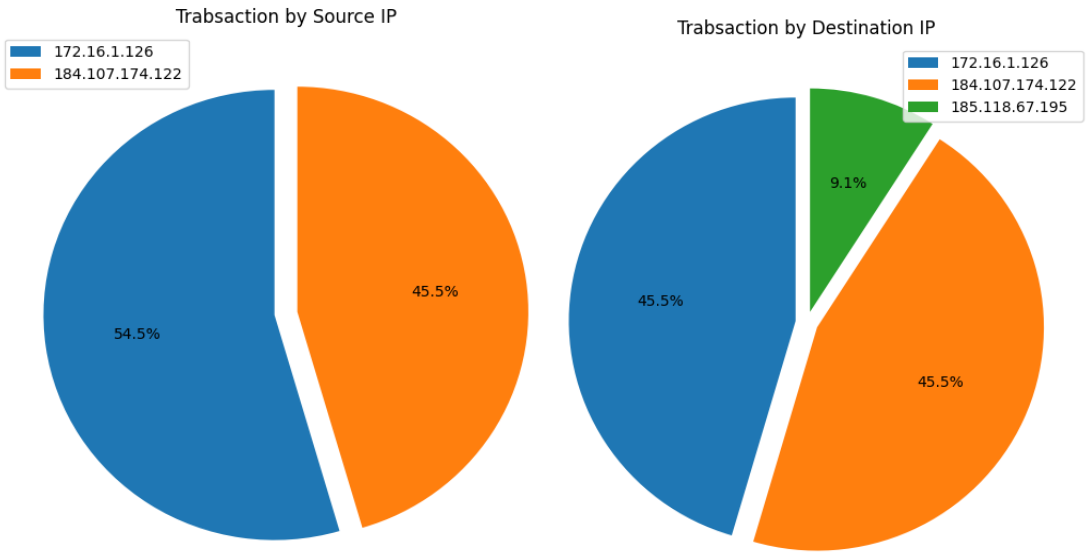
```
lounes98kheris@hack98king:/var/lib/suricata/rules$ sudo suricata-update
22/12/2021 -- 21:44:57 - <Info> -- Using data-directory /var/lib/suricata.
22/12/2021 -- 21:44:57 - <Info> -- Using Suricata configuration /etc/suricata/su
ricata.yaml
```

Partie 3 : Réponses aux questions

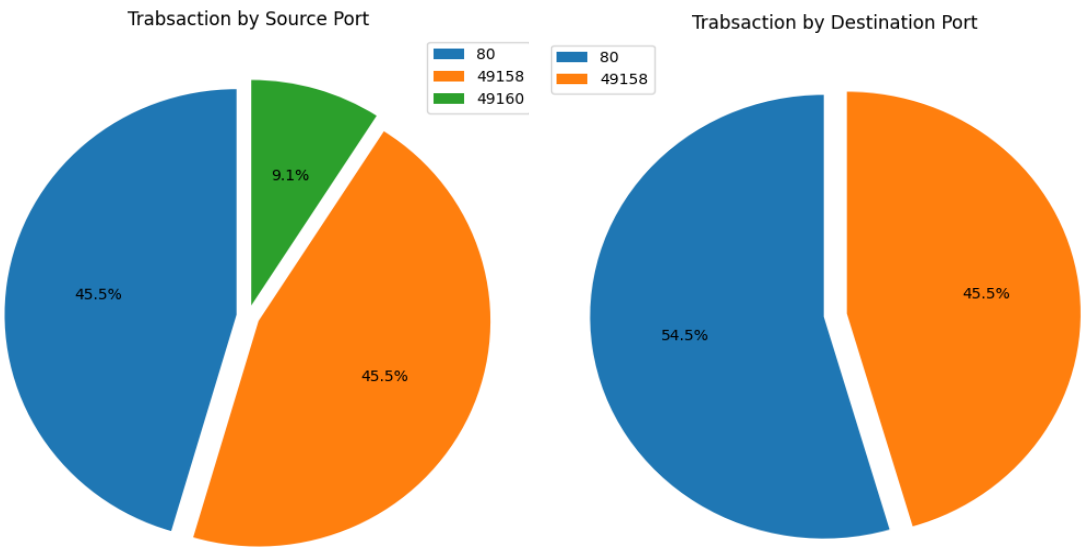
— **Quelles sont les personnes dont nous avons capturé le trafic ? envoyaient-ils des e-mail ? travaillez-ils à distance ? regaraient-ils un film ?**

Les personnes dont nous avons capturé un trafic malveillant/suspicieu sont les hôtes 172.16.1.126 et 184.107.174.122, OÙ la machine 172.16.1.126 était entrain de consulter un site internet **bapanivato.abjibapanichhatedi.org** dont l'adresse IP **184.107.174.122** situé au **CANADA** et lors du telechargement d'image il c'est fait implenté un cheval de troie **(TROJAN)** et on remarque cela car le hôte 172.16.1.126 communique beaucoup avec 184.107.174.122 et lors que la communication les port principalement utilisés sont le port 80 et le port 49158
Afin de mieux voire ces résultats concernant le trafic malveillant ont a extrés les informations suivante liées à ce trafic :

1- Graphique démontrons les transactions par les adresses IP Sources, adresses IP destinations :



2- Graphique démontrons les transactions par les Port Sources, Port destinations :



— **Dans quel pays sont-ils ?**

l'hôte 184.107.174.122 est au canada.

l'hôte 172.16.1.126 ça location est inconnu

— **Effectuez-vous Deep Packet Inspection ? Expliquer**

Oui on fait de l'inspection approfondie de paquets, mais plus concraitement en fait une partie de la DPI, car la DPI golobalement consiste dans un premier temps à identifier le trafic indésirable et de l'isolé, or nous, que se soit en inspectant le contenu de chaque paquet de données avec wireshark, ou l'identifications de trafic malveillant avec suricata en effectuant des alertes, on fait que de l'identification car inspecter les en-têtes des paquet, mais aussi les données qu'il contient c'est de la DPI. Identifier les protocoles non conformes, les virus, les spam, les intrusions ou toute autre caractéristique définie afin de déterminer si le paquet inspecté peut être acheminé ou s'il doit être renvoyé vers une autre destination c'est de la DPI et oui on fait tout cela lors de nos analyses Wireskark ou Suricata.

— **Effectuez-vous une surveillance passive ou active ? Expliquer**

Que se soit avec wireskark ou bien Suricata on effectue une surveillance Passive. Vu qu'ont interroge pas les machines du réseau nous mêmes. On se contente juste de surveiller le trafic et de soulever des alertes si nécessaire afin de dire qu'il y'a quelque chose d'anormale qui se passe sur le réseau. Notre analyse ne perturbe en aucun cas le trafic réseau.

— **Un incident de sécurité s'est produit pendant qu'ils faisaient cette activité ? Expliquez**

Lorsque l'hôte 172.16.1.126 surfait sur le net et lorsqu'il a télécharger une image sur le site **bapanivato.abjibapanichhatedi.org** dont l'adresse IP **184.107.174.122** un cheval de troie à été inséré en utilisant cette image se qui a fait en sorte d'une manière ou d'une autre à ce que l'hôte 172.16.1.126

communique beaucoup avec l'hôte 184.107.174.122 en lui envoyant des requêtes grâce au cheval de troie.

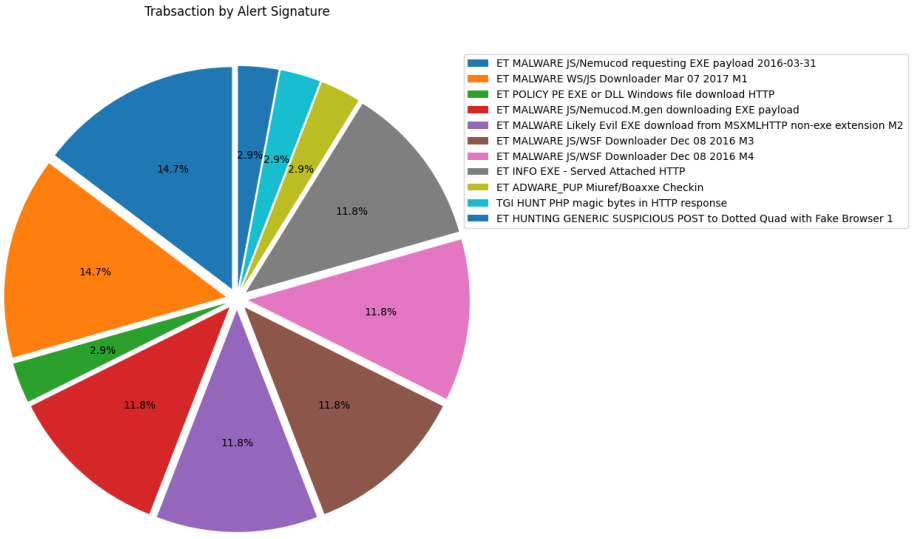
— **Pourriez-vous fournir l'adresse MAC de l'ordinateur infecté ?**

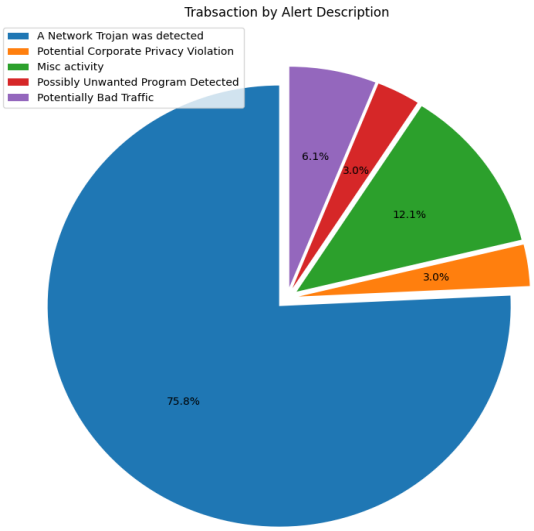
L'adresse MAC de l'ordinateur infecté est : **00 :50 :8b :ab :2e :3c**

— **Utilisez-vous des techniques de surveillance de la sécurité basées sur les signatures ou effectuez-vous une détection d'anomalie ? Expliquer**

Dans notre cas on utilisent une surveillance de sécurité basé sur les signatures car les alertes de suricata nous donne des informations justement sur les signatures de ces alertes. On peut donc facilement déduire quel type de trafic malveillant circule dans notre réseau et quelles IP il est issus et vers quelle source il est déstinée, on peut même avoire la date et l'heure à la quelle ce comportement suspect à été initier et quelques informations complèmentaire su la nature de l'attaque.

Ci-dessous un graphique démontrons les transactions par signature d'alertes et par leurs déscriptions :





— **Selon vous, qui (adresse IP et/ou MAC) est l'attaquant et pourquoi ?**

Selon nous l'attaquant est le serveur **184.107.174.122** avec l'adresse MAC **00 :1d :7e :6c :a2 :9f** car on remarque via les alertes que le telechargement de données ce fait de l'adresse 184.107.174.122 vers 172.16.1.126 et c'est à cause du telechargement d'une image que le cheval de troie à pu être implenté.

— **De quel type d'attaque s'agit-il ? Et quel principe de sécurité (c'est-à-dire la confidentialité, l'intégrité et la disponibilité) tente-t-il de violer ?**

Ont pense que l'attaque vise l'usurpation d'identité car si on analyse les alertes on peut remarqué que aprés avoir parlé avec le serveur (l'attaquant) le hôte (IP : 172.16.1.126) communique avec 185.118.67.195 en lui envoyant un programme malicieux, on peut observé cela grâce à l'alerte Suricata suivante :

ET ADWARE_PUP Miuref/Boaxxe Checkin | Possibly Unwanted Program Detected | 172.16.1.126 :49160 > 185.118.67.195 :80

et on peut ajouter du sens à notre hypoyhèse grâce à cette alerte :

ET HUNTING GENERIC SUSPICIOUS POST to Dotted Quad with Fake Browser 1 | Potentially Bad Traffic | 172.16.1.126 :49188 > 90.149.107.130 :80

Et la on remarque qu'on a un trafic perturbé après que le hôte infecté (IP : 172.16.1.126) aie envoyé une requête POST vers l'adresse d'un autre hôte 90.149.107.130 et cela pour potentiellement avoir des informations indésirables sur se dernier.

On supose donc que notre attaquant (le serveur 184.107.174.122) utilise le hôte (IP : 172.16.1.126) afin d'envoyé des requête malisieuse et indésirable à d'autres machine du réseau.