Machine Learning et Cybersécurité
SCIA 2023

Sujet : Anomaly detection

Dataset : UGR16

Alexandre Rulleau | Hugo Levy | Massil Ferhani

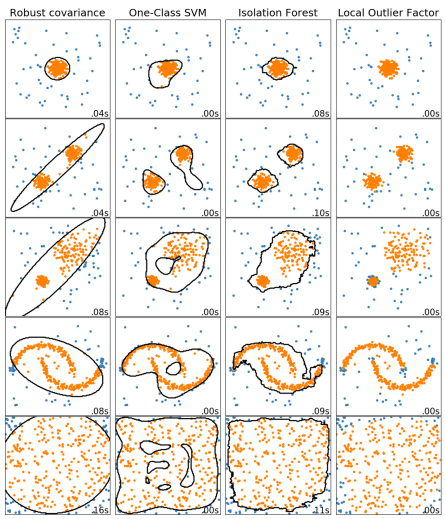
Table des matières

1 Introduction

Dans ce rapport, nous présenterons les spécifications et notre analyse du projet de MLSECU à EPITA. Pour ce dernier, nous avons choisi le sujet de détection d'anomalies sur le jeu de données UGR16.

L'objectif de ce projet est d'analyser une quantité importante d'échanges réseaux afin de pouvoir détecter les échanges potentiellement malveillants au sein de notre jeu de données.
Afin de mener cette analyse, nous avons utilisé différents algorithmes de Machine Learning dont la nature et les résultats seront explicités ci-dessous.

Il existe plusieurs algorithmes de détection d'anomalies disponibles dans la bibliothèque SciKit-Learn en Python :



Parmi ces algorithmes, nous avons utilisé Robust covariance, Isolation Forest et Local Outlier Factor.
Nous présenterons dans la suite de ce rapport une comparaison détaillée des résultats que nous avons obtenu en les utilisant.

2 Le jeu de données

Ce jeu de données UGR16 contient un historique des flux sur le réseau d'un fournisseur d'accès à internet sur une période de 4 mois. Chaque flux est caractérisé par plusieurs attributs.

Voici les caractéristiques de ces flux sur lesquelles nous nous sommes appuyés pour mener à bien notre analyse :

— L'horodatage de l'envoi
— La durée de l'échange
— L'adresse IP de l'expéditeur
— L'adresse IP du destinataire
— Le port de source
— Le port de destination
— Le protocole de transmission (TCP, UDP, etc.)
— Le nombre de paquets échangés
— Le label de l'échange, pouvant correspondre au type d'attaque

Le jeu de données UGR16 est constitué de réels échanges réseaux, dont les données ont été anonymisées afin de respecter la vie privée des personnes concernées.

Ce dernier est scindé en deux parties :
— Une partie de calibration ne contenant que des données réelles
— Une partie de test contenant des données réelles ainsi que des données synthétisées par le fournisseur d'accès.

L'enregistrement des données pour la partie de calibration à été effectué sur 4 mois (de Mars à Juin 2016), alors que la partie de test à été enregistrée sur 2 mois (de Juillet à Août 2016).
Chaque partie à ensuite été scindée par période d'une semaine afin de rendre le résultat plus facilement exploitable.

2.1 Labélisation

Lors des deux périodes certains types d'attaques ont pu être recensés, notamment :

— Des connexions depuis des IP blacklistés : **blacklist**
— Des scans en SSH suspicieux : **anomaly-sshscan**
— Des spams de connexions effectué par des machines virtuelles : **anomaly-spam**

Cependant pour certaines de ces attaques, la fournisseur d'accès à internet ne peut pas être certain qu'il s'agit d'attaques. Par exemple, pour les attaques labélisées comme étant **blacklist**, bien que les adresses IP soit sur référencées sur une liste noire (blacklist), on ne peut pas être sûr que la totalité des échanges sont malveillants.

Ainsi, pour y pallier dans sa partie de test, le fournisseur d'accès à généré artificiellement des attaques en interne afin d'avoir un jeu de données complet.
On peut ainsi y retrouver d'autres types d'attaques tels que :

— Des botnets : **nerisbotnet**
— Des connexions, où 1 assaillant s'attaque à 1 réseau : **scan11**
— Des connexions, où 4 assaillants s'attaquent à 4 réseaux simultanément : **scan44**
— Des attaques DOS (Denial of Service) : **dos**

Les autres flux ont été labélisés comme des connexions d'arrière-plan (**background**). Néanmoins, tout comme pour les attaques, le fournisseur d'accès ne peut pas certifier qu'il n'y a pas d'attaques dans ce groupement de données.
C'est pourquoi on pourra trouver parmi les flux d'arrière-plan observés des échanges avec des caractéristiques relativement différentes des autres.

Cependant, dans l'analyse des données on considérera que tout échange qui n'est pas labélisé comme **background** est une attaque. Autrement cela sera considéré comme un échange normal.

2.2 Pré-traitement du jeu de données

Nous avons donc décider d'utiliser le jeu de calibration pour entraîner nos modèles et le jeu de test pour les tester. On peut ainsi observer si nos modèles sont capables de détecter des échanges malveillants sur lesquelles ils ne sont pas entraînés.

UGR16 contient une quantité importante de données, c'est pourquoi nous avons décidé de le découper en tronçons pour pouvoir moduler plus facilement les dataframes sur lesquelles nos modèles vont s'entraîner.

Ainsi pour le jeu de données de calibration et celui de test, nous avons sélectionné une semaine d'enregistrement puis découper les données de cette dernière en différent CSV en fonction du label.
Puis nous avons tronqué chaque fichier à 50MB pour pouvoir nous partager les données facilement (puisque nous ne pouvions pas utiliser l'ensemble des données de toute façon).

L'avantage de ce pré-traitement est qu'il nous a permis d'expérimenter plus facilement différentes configurations du jeu de données pour nous nos algorithmes.
Ce découpage nous octroie un contrôle important sur la granularité de nos jeux de données, notamment au niveau du ratio d'attaques et d'échanges normaux.
Cela nous permet également d'avoir une proportion équivalente de chaque type d'attaque dans chacun de nos jeu de données.

Voici à quoi ressemble le jeu de données une fois que nous l'avons chargé dans un dataframe pandas sur Python :

Out[33]:

	end_epoch	duration	src_ip	dst_ip	src_port	dst_port	protocol	flags	status	service_type	packets	bytes	attack
0	2016-04-18 00:02:16	0.996	42.219.158.242	60.56.180.24	33421	80	TCP	.AP.SF	0	0	6	437	blacklist
1	2016-04-18 00:02:16	0.852	42.219.158.242	60.56.180.24	35297	443	TCP	.AP.SF	0	0	52	3030	blacklist
2	2016-04-18 00:02:17	0.936	60.56.180.24	42.219.158.242	80	33421	TCP	.AP.SF	0	0	4	565	blacklist
3	2016-04-18 00:02:17	0.804	60.56.180.24	42.219.158.242	443	35297	TCP	.AP.SF	0	0	72	86267	blacklist
4	2016-04-18 00:02:30	0.000	42.219.152.242	88.205.102.190	38531	25	TCP	....S.	0	0	1	60	blacklist

3 Analyse des données

Une fois l'extraction des données importantes. Voici les données les plus pertinentes. Nous avions à disposition un jeu de données de test de 10000 éléments avec 1000 attaques et 9000 échanges inconnus.

— Les protocoles utilisés pour les attaques :

- TCP
- UDP
- ICMP

— Les protocoles utilisés pour les échanges inconnus :

- TCP
- UDP
- GRE
- ESP
- ICMP
- IPv6

— Après analyse du dataset de train et de test nous constatons que les évènements observés sont similaires.

— Pour les options, après analyse nous constatons que certains ne sont pas utilisés pour des attaques.

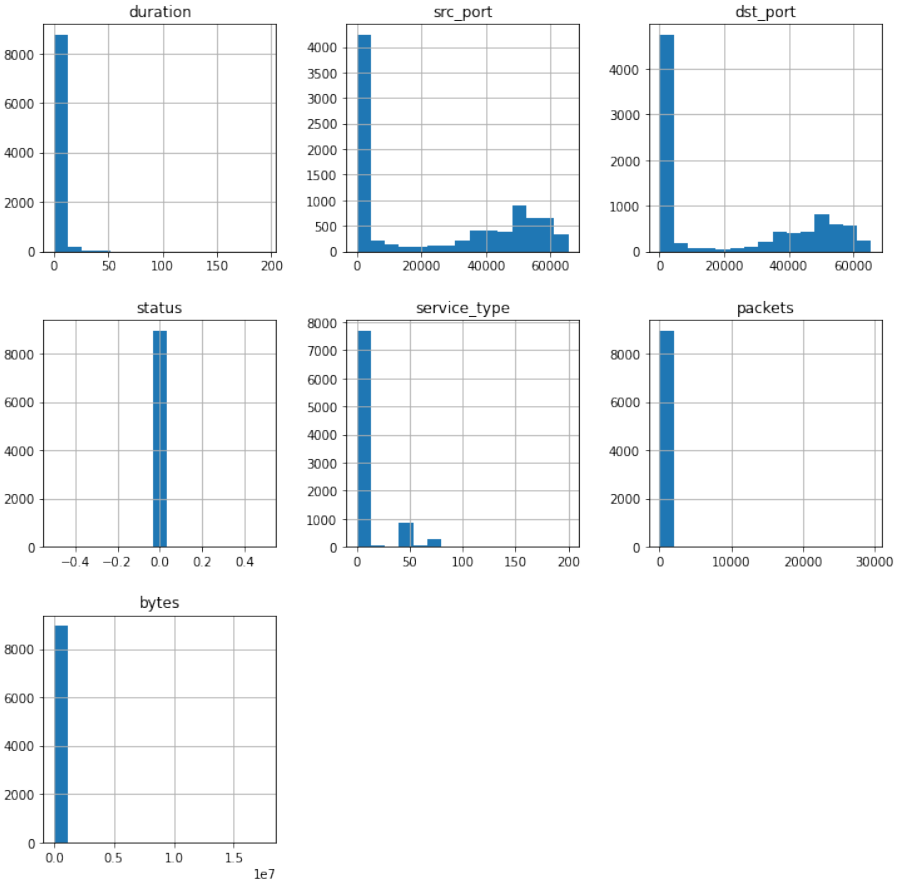
— Il n'y a que 4 services utilises pour les attaques (0, 40, 72, 8). Contrairement au échanges inconnus qui en ont 17 différents.

— Après avoir récupéré les informations sur le tableau, nous constatons que la colonne "status" n'a qu'un seule valeur pour l'ensemble des échanges.

— On peut supposer qu'il y a une redondance des ports utilisés pour les attaques compte tenu de leur faible nombre contrairement aux échanges inconnus (251 contre 2018)

— On peut supposer qu'il y a une redondance des ports et des IP utilisés pour les attaques compte tenu de leur faible nombre contrairement aux échanges inconnus (251 contre 2018)

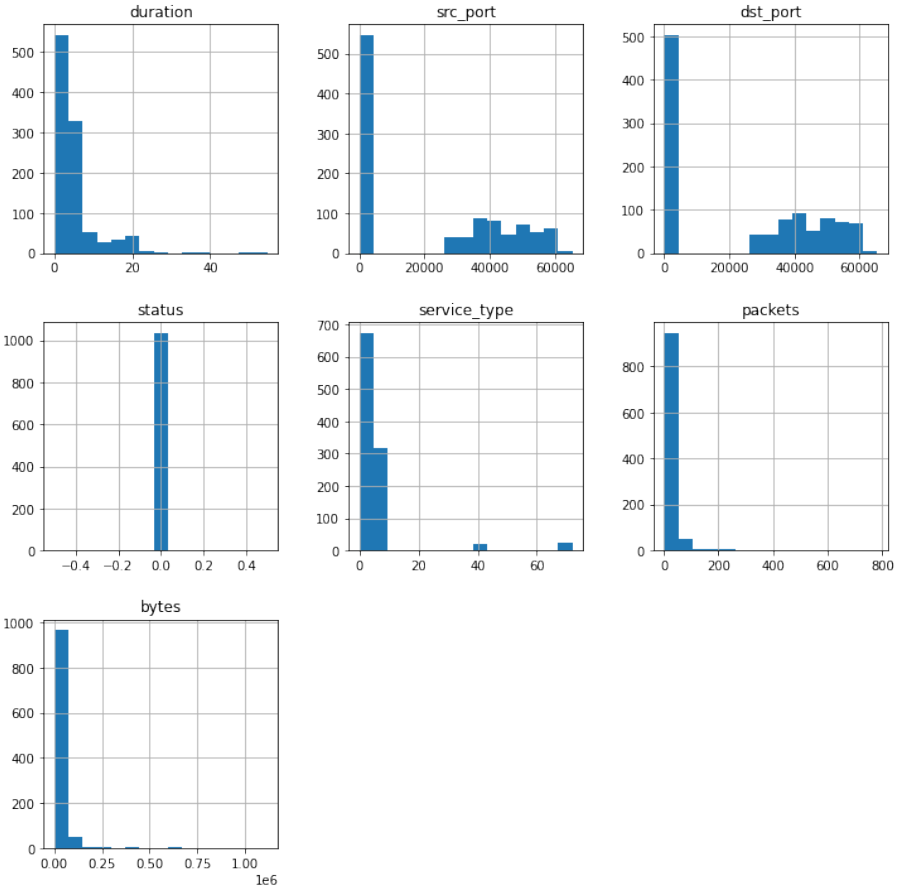
Répartition des données des flux bénins



Details of quantitative features of other exchanges:

	duration	packets	bytes
count	8965.000000	8965.000000	8.965000e+03
mean	1.109757	15.008589	7.854498e+03
std	5.411496	374.938073	2.400408e+05
min	0.000000	1.000000	3.100000e+01
25%	0.000000	1.000000	7.600000e+01
50%	0.000000	1.000000	1.310000e+02
75%	0.324000	6.000000	7.280000e+02
max	194.796000	29544.000000	1.765682e+07

Répartition des données des flux malveillants



Details of quantitative features of known attacks:

	duration	packets	bytes
count	1035.000000	1035.000000	1.035000e+03
mean	4.668881	24.860870	1.485774e+04
std	5.745437	63.051175	7.601059e+04
min	0.000000	1.000000	4.000000e+01
25%	0.888000	5.000000	5.650000e+02
50%	3.540000	12.000000	1.373000e+03
75%	5.040000	15.000000	1.954000e+03
max	54.712000	782.000000	1.118209e+06

Plusieurs observations peuvent être faites à partir de ces deux figures :

— Les durées des flux malveillants sont globalement moins dispersées que celles des flux bénins (25% entre 0,8 et 3,54 secondes et 25% entre 3,5 et 5 secondes contre plus de 75% entre 0 et 0,32 secondes)
— Les nombres de paquets échangés des flux malveillants sont également plus dispersés que ceux des flux bénins (25% entre 1 et 5, 25% entre 5 et 12 et 25% entre 12 et 15 contre plus de 50% à 1 et 25% entre 1 et 6)

On peut aussi remarquer que parmi les flux considérés comme étant non malveillants, les valeurs maximales sont beaucoup plus élevées que le troisième quartile, ce qui est normal, car on rappelle que parmi les flux considérés non malveillants (**background**), on peut trouver des flux malveillants.

4 Nettoyage des données

Comme nous l'avons vu précédemment, la colonne **status** à la même valeur à travers toutes les lignes de nos jeu de données, on peut donc la retirer puisqu'elle ne donne pas d'informations nécessaire.

```
train_df.drop(['status'], axis=1, inplace=True)
test_df.drop(['status'], axis=1, inplace=True)
```

Pour avoir des données exploitables, nous avons utilisé l'algorithme de **one-hot-encoding** de scikit-learn. Cependant cela produisait un nombre massif de colonnes, parce que certaines colonnes avaient trop de valeurs différentes, notamment la colonne avec contenant l'horodatage. Nous avons encoder certaines colonnes à la main.

Concernant la colonne d'horodatage, nous avons extrait l'heure à laquelle les échanges ont eu lieu pour ne pas biaiser nos modèles avec la date à laquelle ils ont été enregistrés. Puis nous avons converti les valeurs de la colonne en temps UNIX pour avoir une valeur numérique.

Pour les IP de sources et de destinations, nous avons utilisé la bibliothèque python **ipaddress** pour convertir nos IPv4 en valeurs numériques.

Nous avons également retiré la colonne avec le label dans cette étape, afin de pas biaiser nos modèles au moment de l'entraînement.
Ainsi seul les colonnes de protocole et d'options ont été encodé avec **one-hot-encoding** rendant le résultat plus sensé et exploitable.

5 Visualisation des données

Voici quelques représentations graphiques de nos jeu de données (d'entraînement et de test) que nous avons utilisé, avec les flux d'attaques en rouge et les flux bénins en bleu selon différents critères :

— L'heure, le nombre de bytes échangés et la durée de l'échange
— Le nombre de paquets, le nombre de bytes échangés et la durée de l'échange
— Le port de destination, le port source et la durée de l'échange

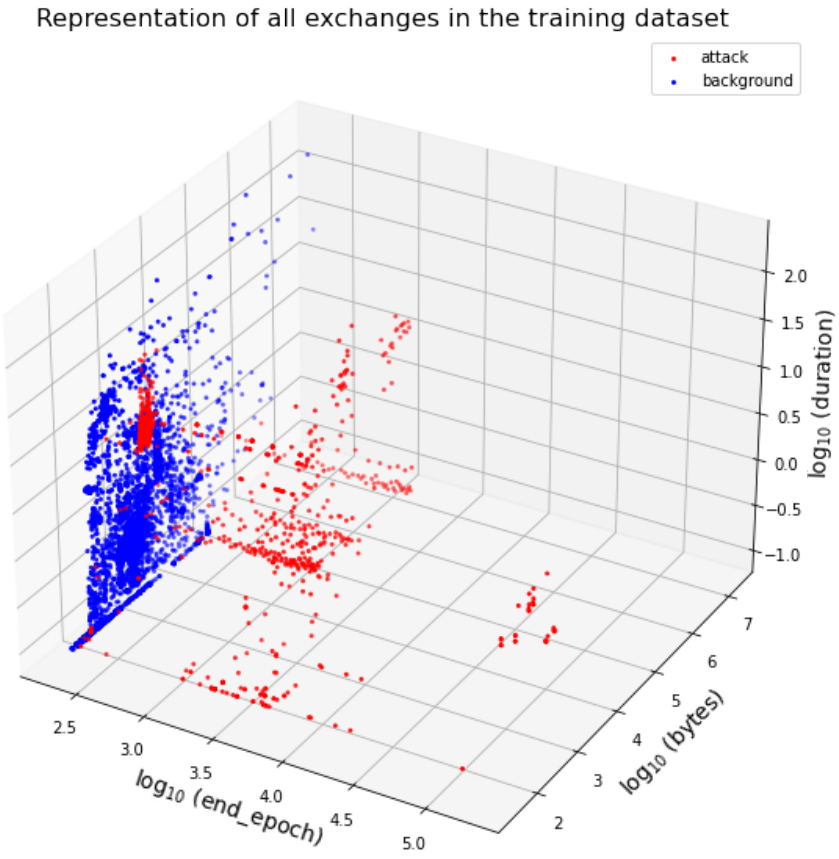
Avec ces 3 représentations, on peut observer les phénomènes suivants :

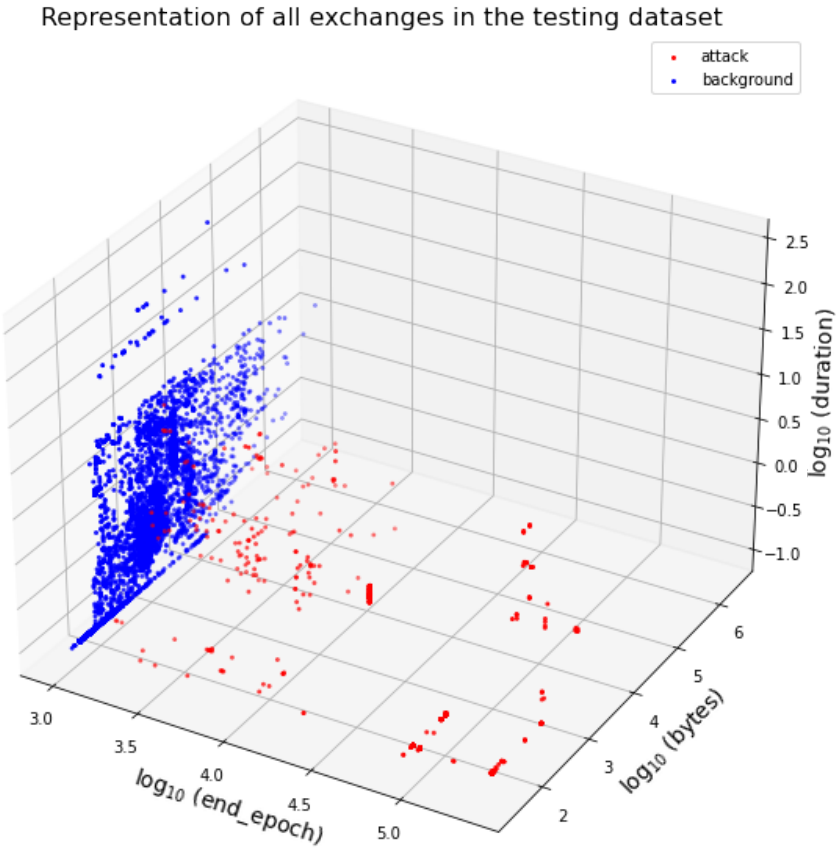
— L'horodatage diffère énormément entre les attaques et les échanges bénins.
— Le nombre de bytes et de packets échangés ne semblent pas être déterminant.
— Les attaques ont lieu sur un jeu réduit de port de source et d'origine.

Concernant la colonne d'horodatage, on peut expliquer la différence par le fait que dans le jeu de données URG16, le nombre d'attaque est bien inférieur au nombre d'échanges bénins. Ainsi ils sont plus dispersés dans le temps.

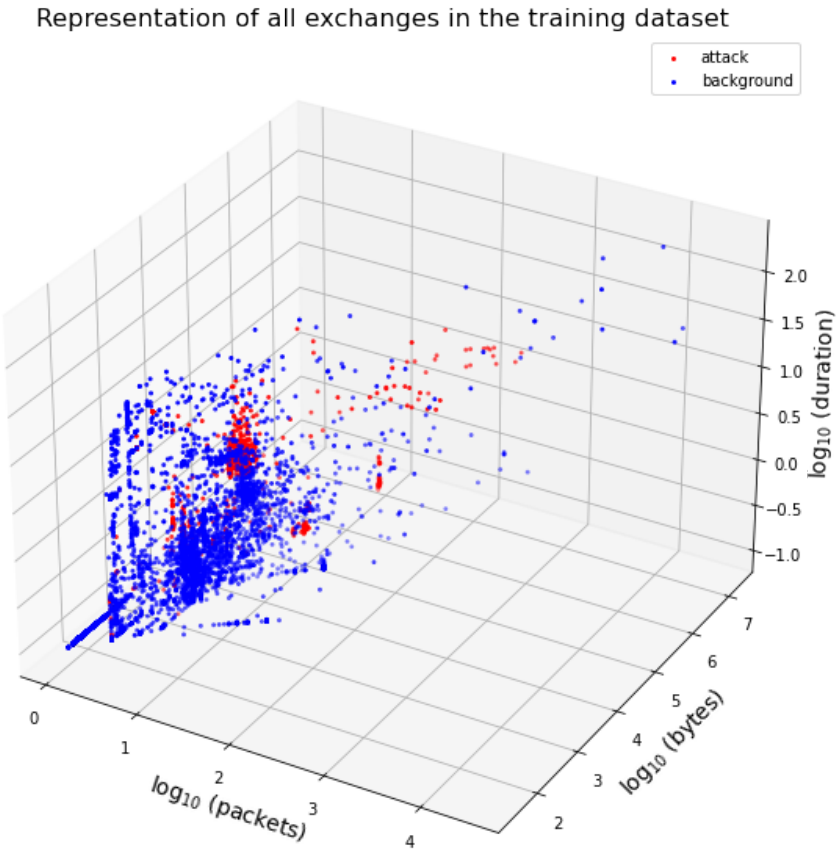
Ainsi en prenant les 9000 premiers flux du jeu de données bénins, et des jeux de données d'attaque, on obtient cet écart de valeur. La totalité de nos flux labélisés comme **background** ont été enregistrés entre 0h et 0h06.

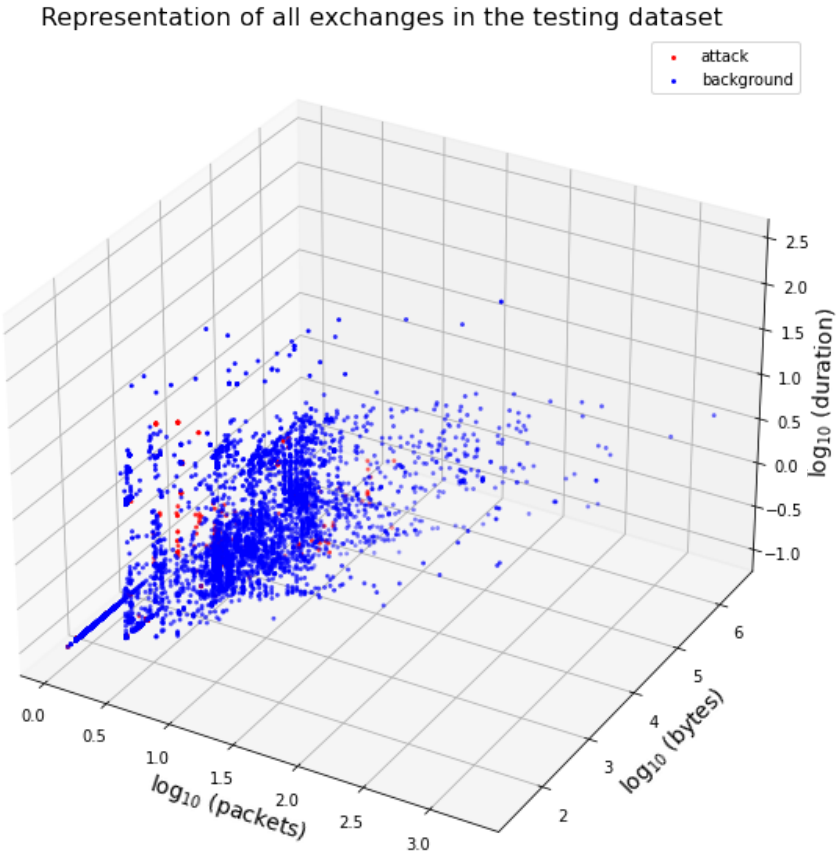
Selon l'heure, le nombre de bytes échangés et la durée de l'échange



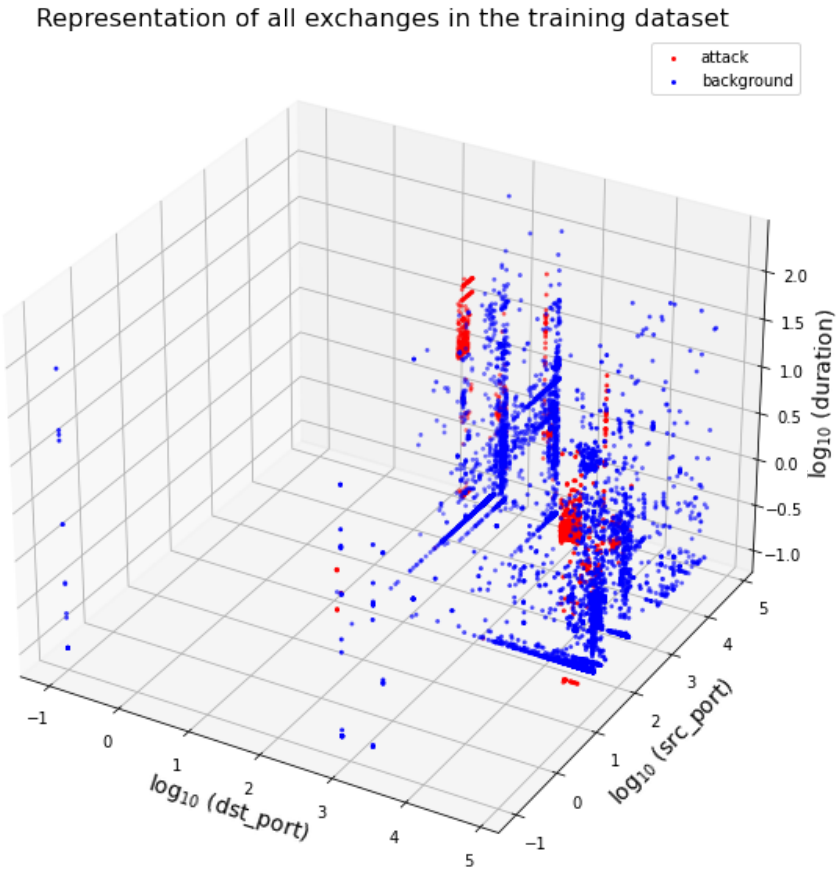


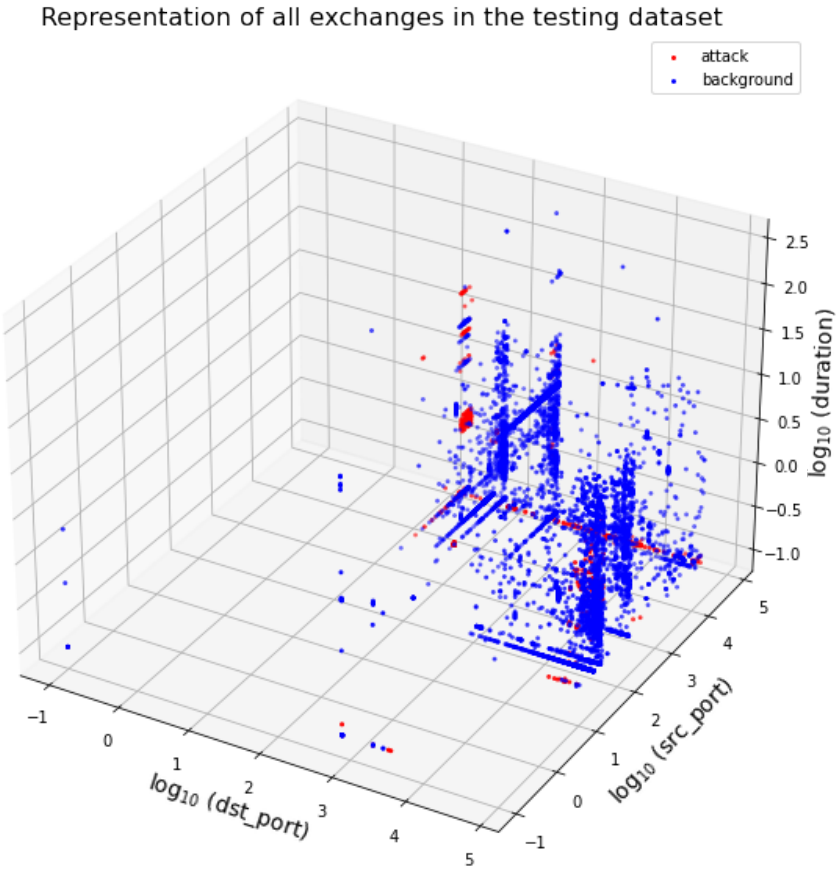
Selon le nombre de paquets, le nombre de bytes échangés et la durée de l'échange





Selon le port de destination, le port source et la durée de l'échange





6 Algorithmes de détection d'anomalies utilisés

L'objectif de l'analyse était de pouvoir repérer les outliers (valeurs aberrantes) avec 3 algorithmes de détection différents.
Une fois que les valeurs aberrantes ont été repérées, nous les avons corrélées avec les attaques labellisées dans le jeu de données UGR16.

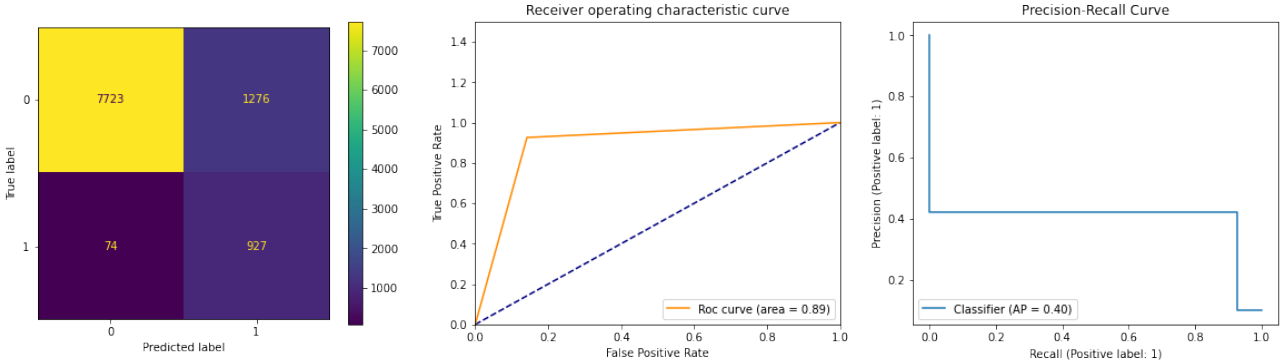
Les 3 algorithmes de détection d'anomalies que nous avons utilisés sont :

Algorithme	Anagramme	Supervisé
Isolation Forests	IF	Non
Local Outlier Factor	LOF	Non
Elliptic Envelope	EE	Non

6.1 Elliptic Envelope

L'algorithme Elliptic Envelope est basé sur le principe que les données suivent une distribution connue.
Cet algorithme modélise les données comme une distribution gaussienne avec une éventuelle covariance entre les caractéristiques.
Il trouve ensuite une limite ayant une forme elliptique qui entoure la plupart des points de données à l'aide du ratio d'outliers donnée en entrée. Tous les points se trouvant à l'extérieur de l'ellipse sont considérés comme des anomalies.

Voici les résultats que nous avons pu obtenir en utilisant cet algorithme :

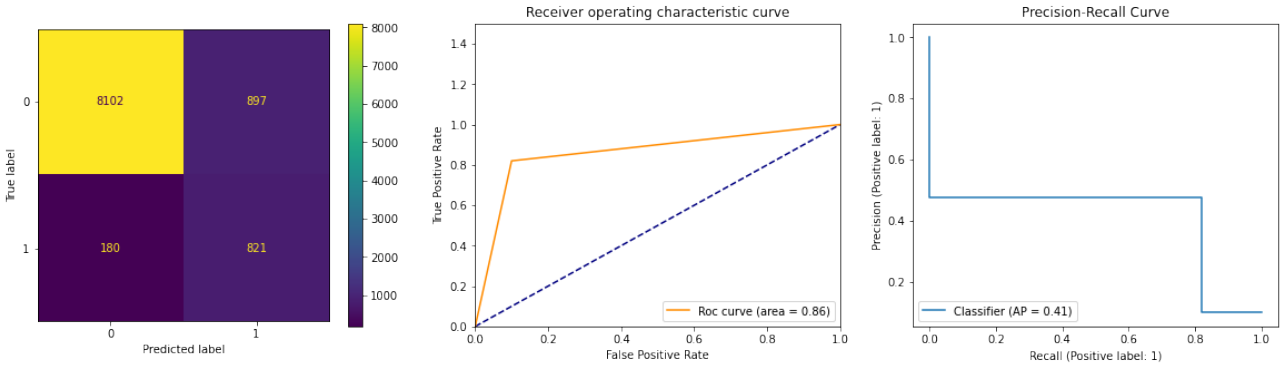


Avec Elliptic Envelope, on obtient peu de faux négatifs (74 sur 10 000), mais beaucoup de faux positifs (1276 sur 10 000, soit 12 %)

6.2 Isolation Forest

Isolation Forest identifie les anomalies dans un jeu de données en isolant les anomalies car elles sont peu nombreuses et dispersées au lieu d'entourer les points de données normaux.
Cet algorithme consiste à caractériser les données normales puis à identifier les instances qui ne sont pas conformes au profil normal comme des anomalies.

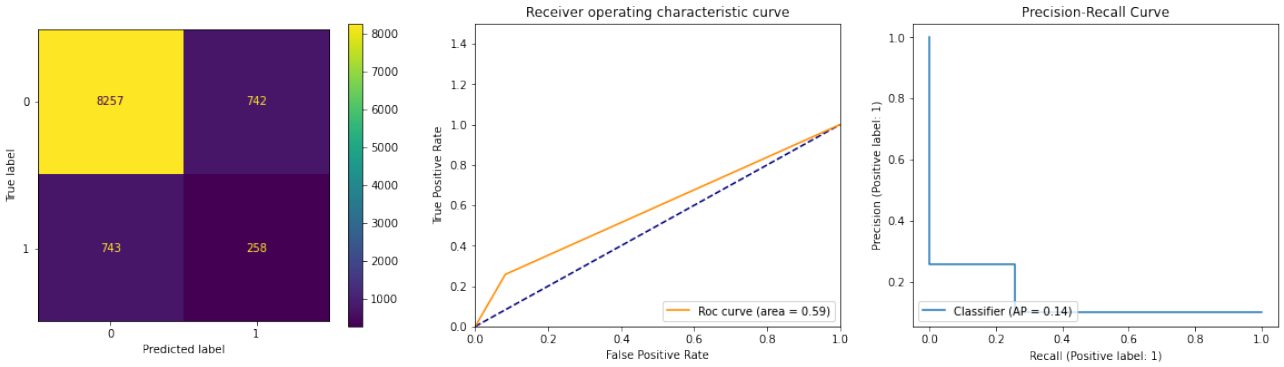
Voici les résultats que nous avons pu obtenir en utilisant cet algorithme :



Avec Isolation Forest, on obtient plus de faux négatifs (180 sur 10 000) et moins de faux positifs (897 sur 10 000) que Isolation Forest.

6.3 Local Outlier Factor

Local Outlier Factor est basé sur la densité qui permet d'identifier les anomalies dans un jeu de données à plusieurs dimensions en indiquant le degré d'anomalie et en quantifiant la caractéristique aberrante de la donnée.
LOF identifie les anomalies en comparant la densité d'un point avec celles de ses voisins. Si le point présente une densité nettement inférieure à celles de ses voisins observés, il est considéré comme une anomalie.



Local Outlier Factor est l'algorithme avec lequel on obtient le moins de faux positifs, mais qui a quasiment autant de faux négatifs(743 sur 10 000) que de faux positifs (742 sur 10 000).

6.4 Comparaison des algorithmes utilisés

On s'intéresse à la complémentarité des 3 algorithmes qu'on a utilisé pour mener à bien notre analyse.

La complémentarité de deux algorithmes se mesure en fonction du nombre d'outliers qu'ils ont en commun et du nombre d'outliers qu'ils n'ont pas en commun :

Deux algorithmes sont complémentaires si l'intersection de l'ensemble des outliers détectés par le premier et l'ensemble des outliers détectés par le second contient peu d'éléments et que leur union contient un maximum d'anomalies.

Pour chaque algorithme nous avons obtenu les résultats suivants :

— Nombre d'attaques détectées par Local Outlier Factor : 258
— Nombre d'attaques détectées par Isolation Forest : 821
— Nombre d'attaques détectées par Elliptic Envelope : 927

Voici les résultats que nous avons obtenu en nous intéressant à la complémentarité des 3 algorithmes que nous avons utilisé :

— Nombre d'attaques détectées par les 3 algorithmes : 202
— Nombre d'attaques détectées par Local Outlier Factor et Elliptic Envelope : 203
— Nombre d'attaques détectées par Isolation Forest et Elliptic Envelope : 770
— Nombre d'attaques détectées par Isolation Forest et Local Outlier Factor : 244
— Nombre de flux de fond détectés comme des attaques par les 3 algorithmes : 68

Avec ces chiffres, on se rend compte que Local Outlier Factor détecte beaucoup moins d'attaques que les deux autres que nous avons utilisé (Elliptic Envelope et Isolation Forest).

Les 68 flux détectés comme des attaques par les 3 algorithmes sont probablement des attaques qui ont été labélisées comme des flux d'arrière plan.