Compte rendu projet supervision KGB

1. Introduction

Ces dernières années, la criminalité via le vecteur informatique s'est renforcée. Cette augmentation se traduit notamment par une hausse des intrusions dans les systèmes d'information, des attaques par déni de service ou encore des vols en interne. Cette évolution rend indispensable l'utilisation de moyens de détection permettant de détecter des comportements malveillants. L'article que nous avons étudié, *Identifying suspicious users in corporate networks (T. Pevny et al.)*, décrit un détecteur basé sur l'identification des métadonnées liées aux paquets réseaux. L'objectif de ce rapport est de présenter l'implémentation que nous avons faite de cet algorithme, et de le tester sur un scénario contenant du trafic normal et malveillant pour en mesurer la performance.

2. Motivation et sélection d'un ensemble de données sous partie

Dans un premier temps, nous avons choisi de nous concentrer sur le scénario numéro 10, car il nous semblait particulièrement pertinent en raison de la diversité des situations qu'il englobe, notamment l'IRC, le DDOS et l'US dans le cadre d'une attaque DDoS UDP. Cette étude nous permet de réaliser une analyse des comportements malveillants des botnets. Il se distingue par une charge de trafic conséquente (1,3 million de NetFlows, 73 GB de données de paquet total) et un nombre élevé de bots (10), permettant d'observer des interactions variées. La proportion significative de Botnet Flows (8,11%) facilite l'étude des signatures d'attaques, tandis que la faible occurrence des C&C Flows (0,0028%) met en évidence la complexité de la détection des communications furtives. Ce scénario constitue ainsi un cas d'étude représentatif des défis actuels en cybersécurité.

3. Statistical dataset analysis to understand its structure and guide the expected behavior of the subsequent detection algorithm

Nous avons utilisé le dataset du scénario 10 nommé CTU-Malware-Capture-Botnet-51 à l'URL :
https://www.stratosphereips.org/datasets-ctu13

Le dataset CTU-13 est un dataset capturé à l'Université CTU en République Tchèque en 2011 qui contient un mélange de trafic de botnet avec le trafic normal et celui du background.

Ce dossier contient plusieurs fichiers ;

- README.txt
- rbot.exe
- botnet-capture-20220818-bot.pcap
- capture20110818.binetflow

Le fichier README décrit la nature des autres fichiers avec des infos génériques. On apprend notamment que le flux est capturé se situe autour de l'adresse IP 147.32.84.165.
Le hash MD5 de l'executable RBOX.exe est 2467b3c8b259cecd6ce2d5c31009df10.

Rbox.exe correspond à l'exécutable du Malware du botnet nommé RBOX.

Le fichier botnet-capture.pcap représente la capture des paquets définissant le périmètre de l'étude du botnet. Celui-ci provient d'un fichier capture.pcap de 73 GB pour 90 389 782 paquets qui n'est pas rendu public pour des raisons de respect de la vie privée des utilisateurs sur le réseau.

Le fichier binetflow pour bidirectional NetFlow files a été généré par Argus sur la base du fichier pcap. Le bidirectionnel est une version améliorée de l'unidirectionnel NetFlow qui permet de mieux différencier le client et le serveur mais aussi de mieux détailler les labels détaillant le type de paquet considéré. C'est ce fichier que nous allons

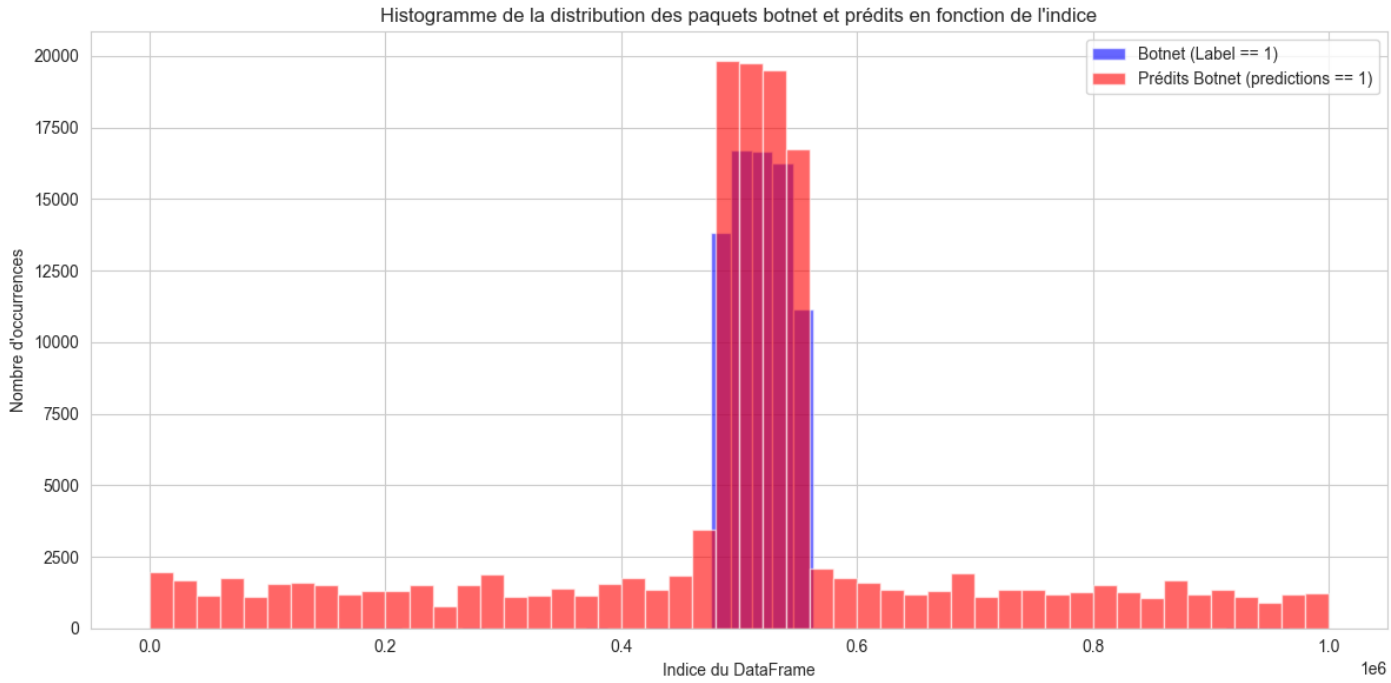
utiliser pour entraîner notre algorithme KGB. Le format des lignes est le suivant : StartTime, Dur, Proto, SrcAddr, Sport, Dir, DstAddr, Dport, State, sTos, dTos, TotPkts, TotBytes, SrcBytes, Label

Ex : 2011/08/18 10:21:46.633335,1.060248,tcp,93.45.239.29,1611, -
>,147.32.84.118,6881,S_RA,0,0,4,252,132,flow=Background-TCP-Attempt

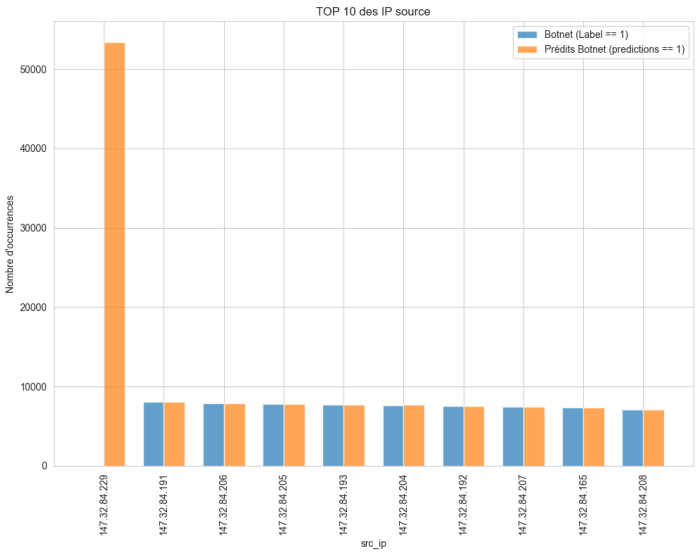
Le label correspond à la catégorie de donnée à prédire par le modèle dans cet algorithme de prédiction.

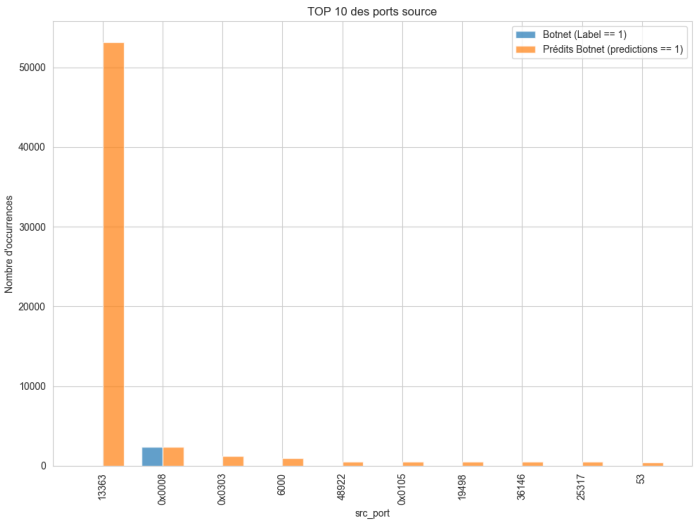
Analyse des données :

Nous allons analyser les données de notre dataset du scénario 10 tout en le comparant avec les prédictions de détection faite par l'algorithme KGB pour mieux comprendre la structure de ce dataset.

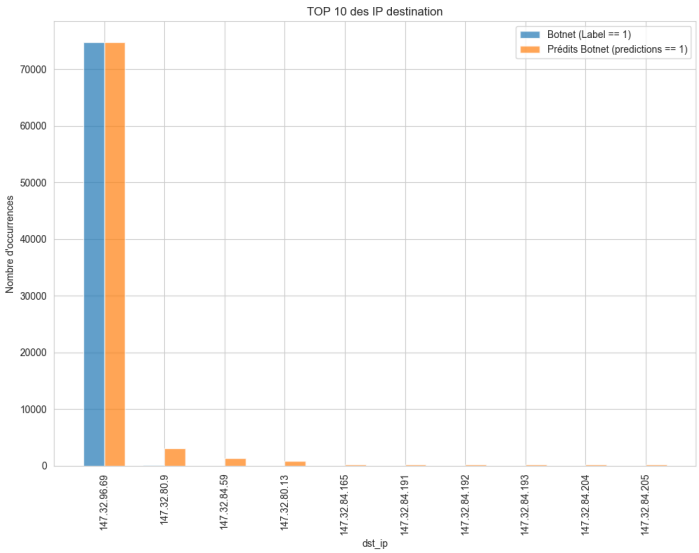


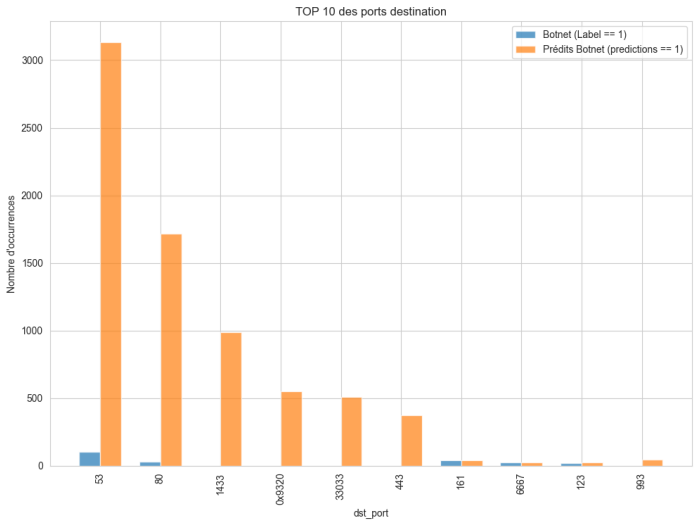
Ici, nous remarquons que l'ensemble des paquets provenant du Botnet se situe au milieu du dataset. Cela explique pourquoi il était nécessaire de prendre en plus au moins 500 000 paquets à analyser pour en détecter effectivement. Nous remarquons de même que la distribution de faux positifs est quasi-uniforme sur tout le dataset.



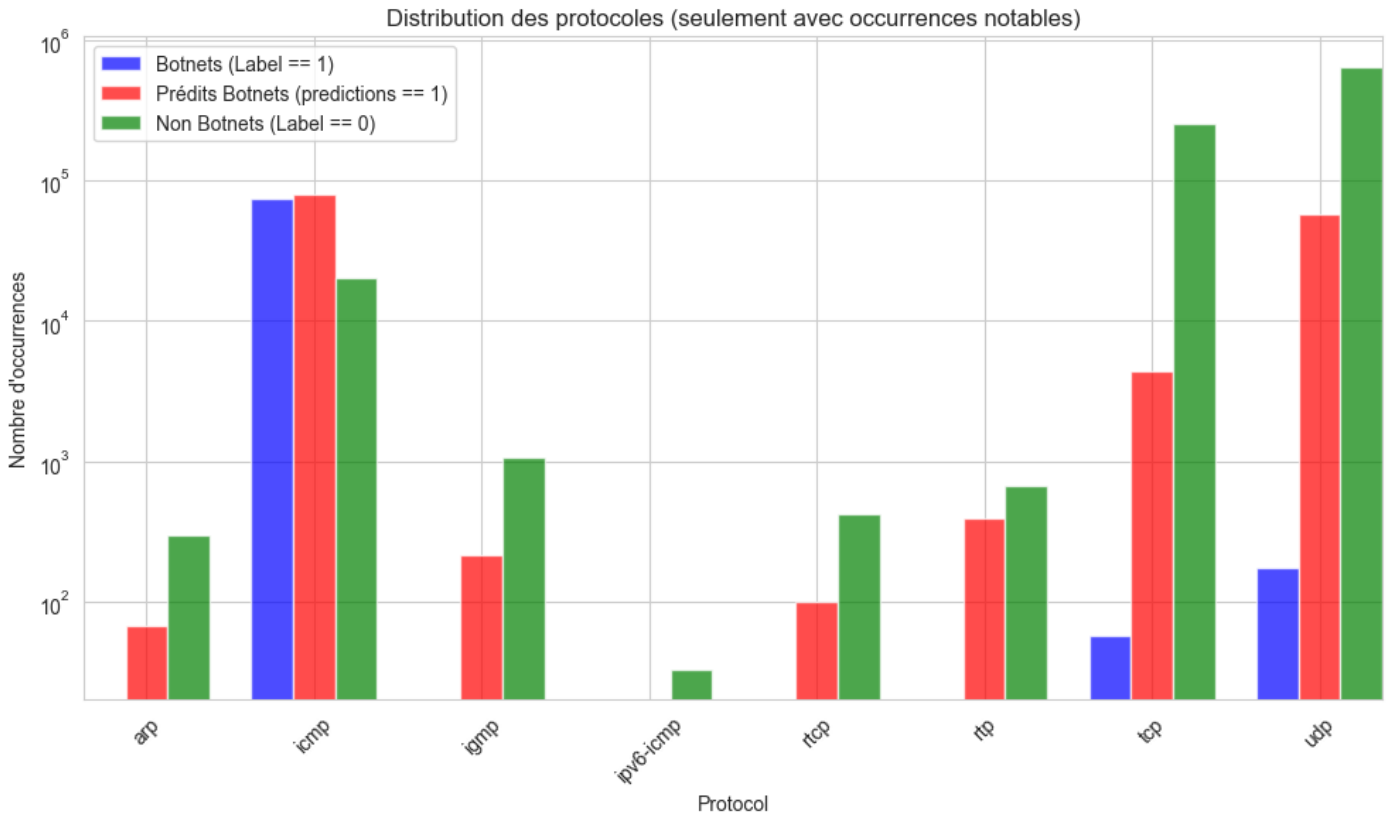


On remarque que la grande majorité des faux positifs proviennent de l'IP source 147.32.84.229 sur le port 13363. En filtrant hors de cette IP source, il serait possible d'améliorer considérablement les scores de performance de notre algorithme. Ceci est une piste d'amélioration. De plus, on remarque que la distribution des adresses IP source associée au botnet est uniforme. Nous comprenons alors ici que ce botnet de type DDOS se constitue d'une multitude de bot devant envoyer un nombre équivalent de requête à la cible.

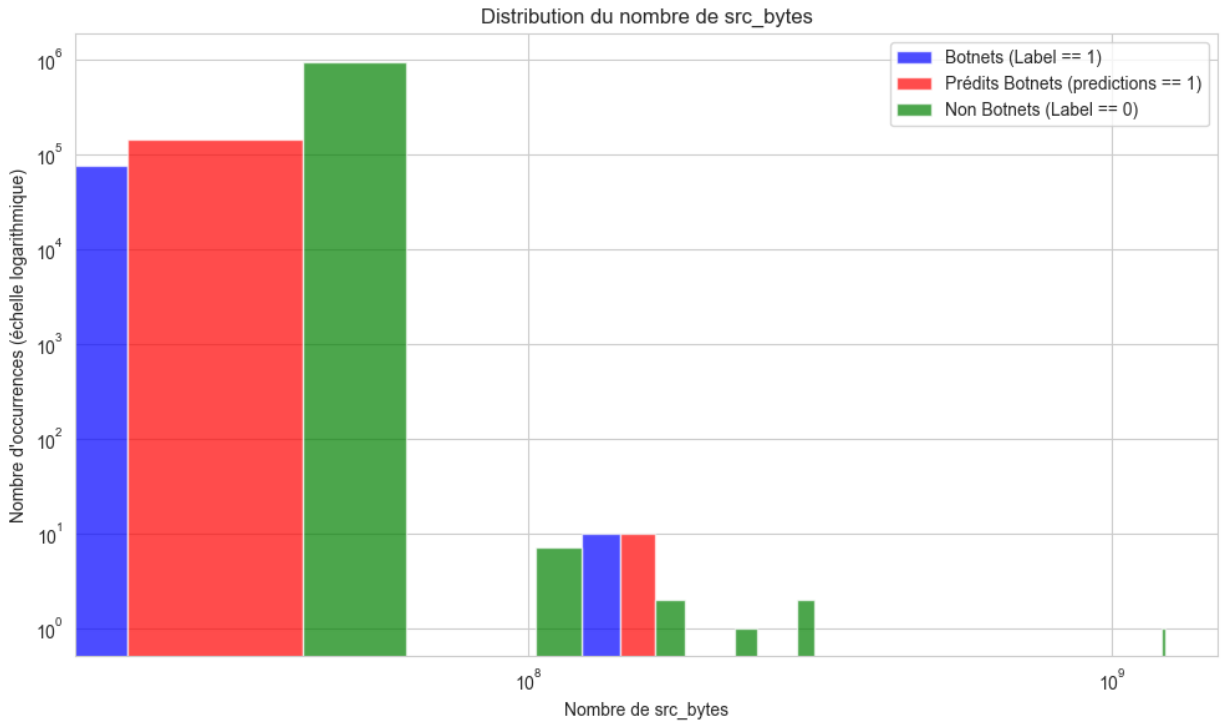




On remarque que l'entièreté de l'adresse IP destination attaquée par le botnet est celle de 147.32.96.69. C'est donc la cible du botnet. De plus, la répartition de la distribution du port de destination des paquets semble quasi-uniforme. Nous comprenons que l'attaquant à fait varier ses ports de destination. Cela ressemble à un scan de réseau comme réalisé par NMAP par exemple. Cette grande variance de valeur pour dst_port permet à notre algorithme KGB basé sur de l'analyse statistique de bien détecter les anomalies.



On remarque que les protocoles utilisés par le botnet sont majoritairement de l'ICMP. S'ensuivent les protocoles UDP et TCP qui restent en proportion moins utilisés que par le background. Le fait que ICMP soit prévu pour des diagnostics réseaux, il est moins fréquent que TCP et UDP (échelle logarithmique). Le fait que le botnet utilise principalement ce protocole ICMP accentue le caractère anormal du botnet. Cela le rend donc plus facilement détectable par notre algorithme.



On remarque que les paquets provenant du botnet possèdent un faible nombre de bytes provenant de la source contrairement au background. Cela accentue le caractère anormal des paquets provenant de botnet

4. Motivation et sélection de l'approche de détection mise en œuvre

Dans le cadre de ce projet, nous avons décidé d'implémenter l'algorithme KGB. L'algorithme KGB est un algorithme de détection d'anomalies qui, via l'analyse en composantes principales (PCA) sur la matrice d'entropie, sépare les grandes variances (composantes majeures) et les petites variances (composantes mineures) pour repérer à la fois les anomalies franches et les comportements anormaux plus subtils. Il a été initialement proposé pour améliorer la gestion des alertes d'intrusion en regroupant automatiquement les alertes similaires et en réduisant le nombre d'alertes à traiter.

Nous avons décidé de choisir ce dernier car il semblait être le plus performant parmi les autres algorithmes si son temps d'exécution d'entraînement est plus lent et énergivore. Cet algorithme reste globalement accessible même s'il n'était tout de même pas très intuitif à implémenter. Enfin, nous avons également souhaité utiliser ce dernier en raison de sa capacité à allier la puissance des algorithmes génétiques et l'efficacité du clustering K-means donc globalement plus axé utilisation dans un SOC d'entreprise.

Dans un environnement SOC, où les analystes font face à un volume massif d'alertes issues de diverses sources (IDS, pare-feux, logs systèmes), l'objectif est de réduire les faux positifs et de mettre en évidence les véritables menaces. KBG répond à cette problématique en regroupant les alertes similaires, ce qui permet de mieux classer les événements et d'identifier les anomalies plus efficacement. Son efficacité prouvée en réduction du bruit et classification des alertes en fait une approche idéale pour renforcer la supervision des systèmes en cybersécurité.

4.1 - Pseudo code

Entrées :

- D : Jeu de données constitué de flux réseau, avec pour chaque flux un ensemble de caractéristiques (par exemple, nombre de paquets, octets, temps de début, etc.) -----
- Pour chaque flux, la source IP à regrouper
- Paramètre 'mode' : "KGBf" pour utiliser les composantes à haute variance,
  "KGBfog" pour utiliser celles à faible variance
- Seuil d'anomalie (threshold) : valeur critique pour décider si un comportement est anormal

Pour chaque source IP unique dans D :

1. Agréger tous les flux associés à la source IP pour constituer un ensemble E
2. Construire un vecteur de caractéristiques X à partir de E. Dans X, on calcule l'entropie de Shannon pour chaque adresse IP source.
3. Appliquer une analyse en composantes principales (PCA) sur le vecteur X normalisé :
   a. Calculer la matrice de covariance de X
   b. Extraire les valeurs propres et vecteurs propres de la matrice de covariance
   c. Classer les vecteurs propres en fonction de leurs valeurs propres (variance)
4. Selon le mode choisi :
   a. Si mode = "KGBf" : Sélectionner les composantes principales avec les **plus grandes variances**
   b. Si mode = "KGBfog" : Sélectionner les composantes principales avec les **plus faibles variances**
5. Calculer le score d'anomalie S pour la source IP :
   a. Projeter le vecteur X sur les composantes sélectionnées
   b. Calculer la déviation (par exemple, la moyenne des écarts) par rapport à une valeur de référence
6. Comparer le score S au seuil :
   a. Si S > threshold : Classer la source IP (ou les flux associés) comme anormal(e)
   b. Sinon : Classer comme normal(e)
7. Enregistrer le résultat de la classification pour la source IP

Sortie :

- Pour chaque source IP, une étiquette (anomalie / normal) basée sur le score calculé

4.2 - Explication de l’algorithme

L’algorithme KGB est basé sur les travaux de l’entropie de Lakhina dont l’anomalie finale est déterminée à partir des écarts moyens. Cette méthode utilise les entropies des distributions d’adresses IP et de ports pour construire deux modes de détections.

Pour ce faire, deux versions du détecteur KGB existent, **KGBf** qui examine les composantes principales avec de fortes variances tandis que **KGBfog** examine les composantes principales avec de faibles variances.

1. Calcul de l'entropie du trafic

Pour chaque ensemble de flux considéré (par exemple par fenêtre temporelle ou par utilisateur), on calcule l'entropie de la distribution des **adresses IP source**, **adresses IP destination**, **ports source** et **ports destination**. L'entropie de Shannon est définie par :

H(X)=−∑kpklog⁡(pk)H(X) = -\sum_{k} p_k \log(p_k)H(X)=−∑kpklog(pk)

où p_k est la probabilité d'occurrence de la k-ième valeur (bin)

En pratique, on détermine la fréquence de chaque valeur (par exemple chaque port source distinct) puis on calcule $p_k =$ fréquence relative. L'entropie mesure le degré d'incertitude ou de diversité de la distribution. Une entropie élevée indique une distribution plus uniforme (beaucoup de valeurs différentes équiprobables), tandis qu'une entropie faible indique qu'une valeur dominante occupe la majorité des occurrences (distribution très déséquilibrée). Dans le contexte réseau, ces entropies caractérisent la variabilité du trafic d'un utilisateur ou d'une fenêtre temporelle. Par exemple, l'article KGB indique utiliser les entropies des distributions d'adresses IP et de ports afin de modéliser le comportement normal du trafic agrégé.

Dans le code, nous allons calculer ces entropies en comptant les occurrences de chaque adresse IP et port, puis en appliquant la formule ci-dessus. Chaque entropie sera stockée (par exemple dans un DataFrame ou des listes) comme caractéristique pour l'observation correspondante.

2. Optimisation des seuils d'anomalie (approche percentiles)

Plutôt que de fixer manuellement un seuil, on le **détermine dynamiquement** en fonction des données observées. Une approche courante est d'utiliser un percentile élevé de la distribution des scores d'anomalie. Par exemple, choisir le **95e percentile** signifie que ~5% des valeurs les plus extrêmes seront considérées comme anomalies

Concrètement, après avoir calculé un score d'anomalie pour chaque observation (voir section suivante), on peut estimer le seuil comme la valeur au-delà de laquelle se trouvent 5% des scores les plus élevés. Ainsi, le seuil s'adapte automatiquement à la dispersion des données actuelles

Si le trafic normal devient plus variable, le seuil augmentera, et inversement il diminuera si le trafic normal est très stable, évitant des faux positifs ou négatifs dus à un seuil inapproprié.

Dans le code, nous utiliserons `numpy.percentile` pour calculer ce seuil. Par exemple, `seuil = np.percentile(scores, 95)` donne la valeur du 95e percentile. Les observations dont le score dépasse ce **seuil adaptatif** seront marquées comme **anormales**.

3. Réduction dimensionnelle avec PCA

Avant d'appliquer la PCA, il est **essentiel de standardiser** les données d'entropie. Cela signifie mettre chaque caractéristique sur une échelle comparable (moyenne nulle, variance unité) afin qu'aucun type d'entropie ne domine les autres juste par son échelle de valeurs. La standardisation assure que *toutes les caractéristiques contribuent de manière égale à la transformation PCA*, ce qui rend les composantes principales plus significatives

Nous utiliserons `StandardScaler` de scikit-learn pour normaliser les vecteurs d'entropie.

Ensuite, on applique la **PCA** pour réduire la dimensionnalité et capturer les corrélations entre ces entropies. On choisira *dynamiquement le nombre de composantes principales* à conserver en fonction d'un seuil de **variance expliquée cumulée**. Typiquement, on vise à conserver par exemple **95% de la variance** totale

Autrement dit, on augmente le nombre de composantes k jusqu'à ce que la proportion de variance expliquée par les k premières composantes atteigne 95%. Scikit-learn permet de le faire en fixant `n_components=0.95` dans PCA, ce qui sélectionnera le nombre minimal de composantes nécessaires pour atteindre au moins 95% de variance conservée. Cette approche évite de garder des dimensions superflues tout en préservant l'essentiel de l'information.

L'objet PCA ainsi entraîné sur les données normales (d'apprentissage) fournit une **base de référence** du "comportement normal" du trafic dans l'espace réduit. Les entropies de trafic ont tendance à être corrélées en situation normale.

La PCA exploite donc ces corrélations pour définir un sous-espace où se concentrent les observations normales.

4. Détection des anomalies (distance ou score statistique)

Une fois les données projetées dans l'espace des composantes principales, il faut définir un **score d'anomalie** pour chaque observation. Plusieurs méthodes sont possibles :

- **Distance au centre dans l'espace PCA** : Si l'on considère que les données normales forment un nuage autour du centre (moyenne) dans l'espace transformé, on peut mesurer la distance de chaque point à ce centre. Une mesure classique est la **distance de Mahalanobis**, qui tient compte de la dispersion (covariance) de chaque dimension. Dans le cas de PCA, la distance de Mahalanobis d'un vecteur transformé z au centre est

  DM2(z)=∑j=1kzj2λjD_M^2(z) = \sum_{j=1}^{k} \frac{z_j^2}{\lambda_j}DM2(z)=∑j=1kλjzj2

  où λ_j est la variance expliquée (valeur propre) de la j-ième composante principale. Cette formule revient à **normaliser chaque composante** par son écart-type, ce qui est effectué automatiquement si on utilise l'option `whiten=True` de PCA (chaque composante principale normalisée à variance 1). Ainsi, on peut calculer le score d'anomalie comme la somme des carrés des composantes normalisées de chaque observation. Plus ce score est grand, plus l'observation est éloignée du comportement moyen.

- **Erreur de reconstruction** : Une autre approche consiste à projeter l'observation dans l'espace PCA réduit puis à reconstruire ses valeurs originales et mesurer l'erreur de reconstruction. Un écart de reconstruction important signale que l'observation comporte des variations qui n'ont pas été capturées par les composantes principales retenues (c'est un signe d'anomalie). Cette méthode est équivalente à mesurer la part de variance située dans les composantes mineures ignorées (appelée $f_{\perp}$ dans l'article, somme des variances résiduelles)

Dans notre cas, nous opterons pour la **distance au centre dans l'espace PCA**. Chaque observation transformée aura un score calculé (via la distance de Mahalanobis ou euclidienne dans l'espace blanchi) indiquant son degré d'anomalie. Ensuite, en comparant ce score au **seuil** défini par le percentile (section précédente), nous classons l'observation comme *normale* (score sous le seuil) ou *anormale* (score au-dessus du seuil).

L'article KGB original utilise deux mesures orthogonales (variance sur composantes majeures et mineures) et suggère de fixer des seuils adaptatifs pour chacune. Ici, pour simplifier, une seule mesure de score sera utilisée, et le seuil est déterminé empiriquement par percentile des scores comme discuté.

5. Stockage des résultats

Enfin, les observations détectées comme anormales seront **exportées dans un fichier CSV** pour une analyse ultérieure. Ce fichier contiendra par exemple l'identifiant de l'observation (ex: l'horodatage de la fenêtre temporelle ou l'IP de l'utilisateur) ainsi que son score d'anomalie, pour permettre d'approfondir l'étude de chaque cas. L'utilisation d'un format tabulaire CSV facilite le tri, le filtrage et l'import dans d'autres outils d'analyse.

1. **Agréger les flux par source IP**
   Chaque source IP est traitée indépendamment. On regroupe tous les flux associés à une même source pour obtenir un ensemble EEE. Cela permet de créer un profil de comportement pour chaque machine.
   Objectif : Regrouper les données afin d'analyser le comportement global d'un hôte.

2. **Construction du vecteur de caractéristiques**
   À partir de l'ensemble EEE, on construit un vecteur XXX qui résume les propriétés de la source. Par exemple, on peut calculer la somme ou la moyenne de certaines mesures (nombre de paquets, octets, durée, etc.).
   Objectif : Fournir une représentation compacte du comportement de la source.

3. **Application de la PCA**
   On effectue une Analyse en Composantes Principales (PCA) sur le vecteur XXX afin de réduire la dimensionnalité tout en conservant la majeure partie de l'information.

   - **Calcul de la matrice de covariance :** Cette étape permet d'identifier les relations entre les différentes caractéristiques.
   - **Extraction des valeurs et vecteurs propres :** Ces éléments indiquent les directions principales dans lesquelles les données varient.
   - **Classement par variance :** Les valeurs propres classées par ordre décroissant permettent de savoir quelles composantes capturent le plus de variabilité dans les données.

4. **Sélection des composantes selon le mode**
   Le mode choisi détermine quelle partie de l'information est utilisée pour détecter les anomalies :

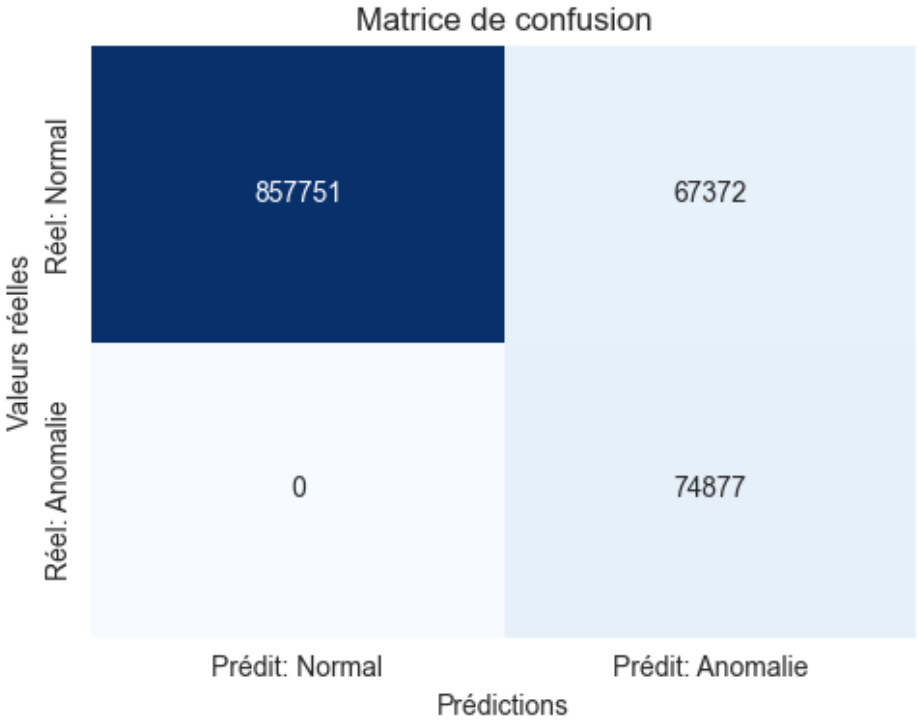
   - **KGBf (composantes à haute variance) :** Ici, on part du principe que les comportements anormaux se manifestent par des écarts marqués dans les dimensions dominantes du comportement.
   - **KGBfog (composantes à faible variance) :** Cette variante cherche à repérer des anomalies subtiles qui apparaissent dans les dimensions moins variées.
     Objectif : Adapter l'analyse aux types d'anomalies recherchées.

5. **Calcul du score d'anomalie S**
   Une fois les composantes sélectionnées, on projette le vecteur XXX sur ces axes. On calcule ensuite un score S qui quantifie la déviation du comportement observé par rapport à un comportement normal attendu (souvent défini par une moyenne ou une distribution de référence).
   Objectif : Quantifier l'anomalie en un nombre qui pourra être comparé à un seuil critique.

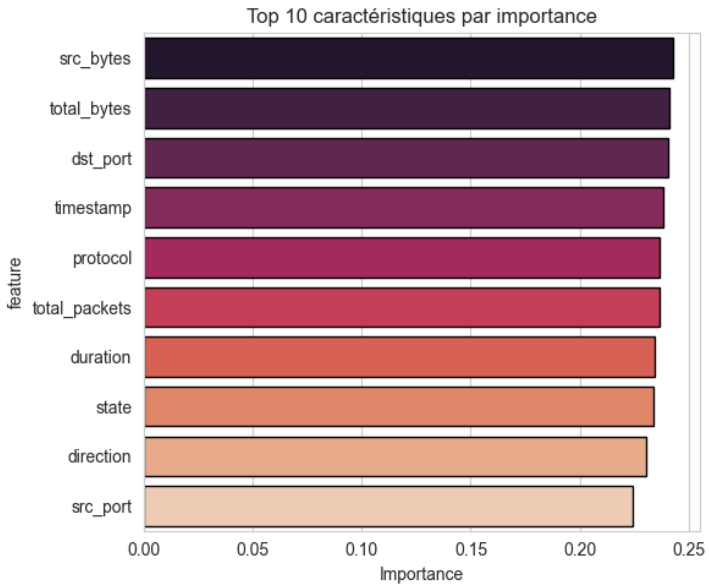
6. **Décision de classification**
   Le score S est comparé à un seuil prédéfini :
   - Si S dépasse le seuil, le comportement de la source IP est considéré comme anormal.
   - Sinon, il est classé comme normal.
     Objectif : Prendre une décision automatique sur la base du score d'anomalie.

7. **Enregistrement du résultat**
   Pour chaque source IP, le résultat (normal ou anormal) est enregistré. Ces résultats pourront ensuite être utilisés pour générer des alertes ou pour alimenter un rapport de détection d'anomalies dans un SOC.

5 . Analyse des résultats de la performance de détection selon les métriques standards



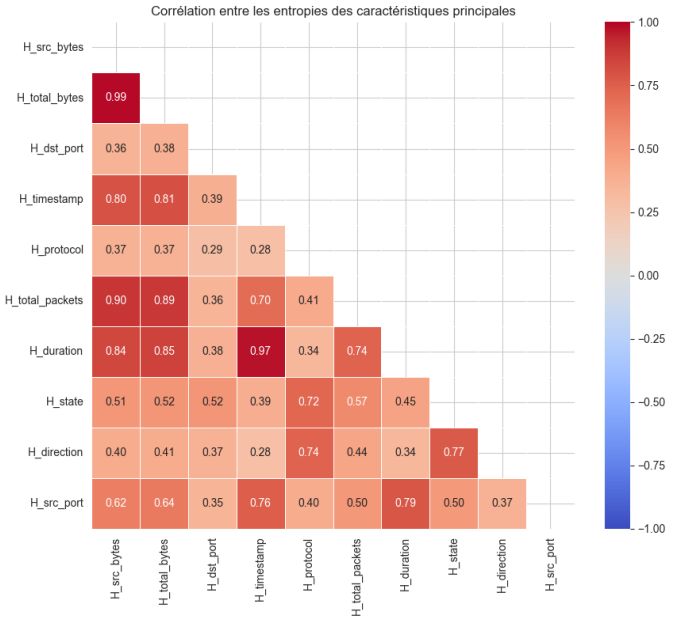
La matrice de confusion illustre les performances de l'algorithme KGB dans la détection des botnets. Notre modèle démontre un rappel parfait en identifiant correctement l'ensemble des 74 877 flux malveillants, sans aucun faux négatif. Cependant, il génère un nombre important de faux positifs (67 372), classifiant des flux normaux comme malveillants. Ceci résulte en une précision d'environ 53%, signifiant que près de la moitié des alertes sont erronées. Cette configuration privilégie clairement une détection exhaustive au détriment de la précision, une approche souvent privilégiée en cybersécurité où manquer une attaque réelle peut avoir des conséquences graves.

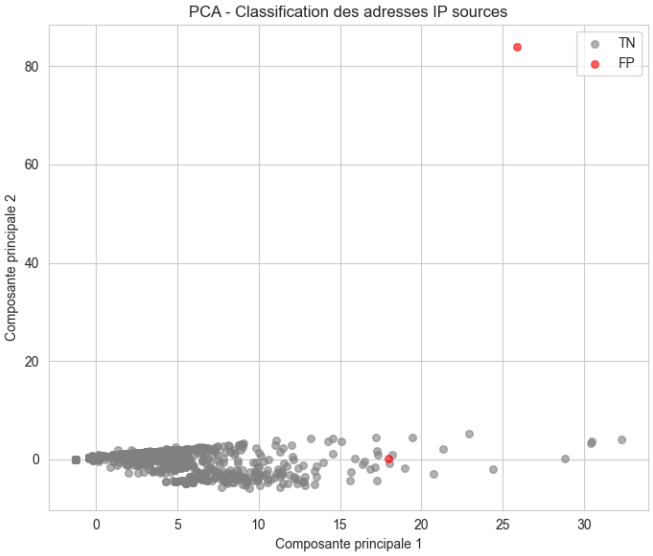


L'analyse des caractéristiques les plus importantes révèle que les métriques volumétriques dominent la détection des comportements malveillants. Les octets source et totaux arrivent en tête, indiquant que les modèles de transfert de données des botnets présentent des signatures distinctives. Les ports de destination figurent également parmi les indicateurs cruciaux, confirmant que les botnets ciblent des ports spécifiques pour leurs communications. La temporalité des échanges (timestamp et durée) ainsi que le protocole utilisé constituent également des signaux forts pour identifier les activités suspectes. Cette hiérarchie d'importance guide l'optimisation de notre modèle en concentrant l'attention sur les caractéristiques les plus discriminantes

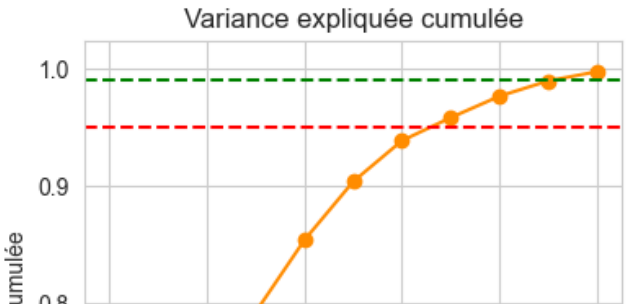
La matrice de corrélation des entropies met en évidence des interdépendances significatives entre les caractéristiques analysées. La corrélation presque parfaite (0.99) entre l'entropie des octets source et celle des octets totaux suggère une forte redondance d'information, indiquant que la majorité du trafic est unidirectionnel. La relation étroite (0.97) entre l'entropie des timestamps et celle des durées révèle que les schémas temporels d'activité suivent des motifs cohérents. En revanche, les corrélations plus faibles impliquant le protocole démontrent que cette caractéristique apporte une information complémentaire précieuse. La corrélation notable entre l'état des connexions et leur direction (0.77) souligne l'existence de signatures réseau spécifiques aux communications botnet. Ces observations permettent

d'affiner notre modèle en exploitant judicieusement l'information unique apportée par chaque caractéristique.


Corrélation entre les entropies des caractéristiques principales
H_src_bytes
H_total_bytes
H_dst_port
H_timestamp
H_protocol
H_total_packets
H_duration
H_state
H_direction
H_src_port
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


PCA - Classification des adresses IP sources
TN
FP
Composante principale 2
Composante principale 1


Variance expliquée par composante


Variance expliquée cumulée

Le PCA réduit la dimension des attributs en composantes principales ordonnées dans leur capacité à expliquer la variance. La première composante est la direction où la variance est la plus grande. La deuxième composante est la direction où la variance expliquée est la plus grande tout en étant orthogonal à la première. Ainsi de suite, la n+1 ième composantes est la n+1 ième direction où la variance expliquée est la grande et orthogonal aux n premières composantes. La distribution de la variance expliquée pour chaque composante principale est illustrée dans la figure de gauche. Il est cohérent qu'elle soit décroissante exponentiellement. La figure de droite permet de déterminer le nombre de composantes principales nécessaires permettant d'atteindre la variance expliquée cumulée. On remarque qu'avec 8 composantes principales, on dépasse 95% de variance expliquée et avec 11 composantes principales, on dépasse 99% de la variance expliquée. Ce taux de variance à expliquer est un hyper paramètre à choisir qui va induire le nombre de composantes principales considéré par l'algorithme KGB.

5 - Perspectives d'amélioration

Pour améliorer l'algorithme KGB, plusieurs axes peuvent être explorés. Tout d'abord, réduire les faux positifs en filtrant les adresses IP problématiques et en optimisant les seuils d'anomalie. Ensuite, enrichir le dataset avec des scénarios variés et appliquer des techniques de prétraitement avancées pour améliorer la qualité des données. La sélection et l'ingénierie de caractéristiques permettraient également de mieux capturer les comportements malveillants.

L'exploration de variantes de PCA et la combinaison avec d'autres algorithmes de détection d'anomalies, comme les réseaux de neurones, pourraient renforcer la robustesse de l'algorithme.
L'intégration dans un SOC et l'automatisation des alertes faciliteraient l'utilisation par les analystes.

Enfin, l'intelligence artificielle offre des perspectives prometteuses. L'utilisation de modèles d'apprentissage profond et de techniques de fusion de données pourrait améliorer la précision et l'adaptabilité de l'algorithme face à des menaces émergentes.
La combinaison de plusieurs algorithmes pourrait également permettre une détection plus fine et plus fiable des anomalies.

6 - Conclusion

Ce projet nous a permis d'explorer en profondeur les problématiques liées à la détection d'anomalies en particulier face à des attaques de type botnet. En partant d'un scénario (CTU-13, scénario 10), nous avons pu implémenter l'algorithme KGB que nous avons étudié, reposant sur l'analyse en composantes principales de mesures d'entropie, a montré de bonnes performances. Toutefois, le taux élevé de faux positifs souligne la nécessité d'un affinement des seuils ou d'un prétraitement plus fin.

Enfin, ce projet illustre l'intérêt de méthodes d'apprentissage automatique non supervisées dans un environnement SOC en particulier dans le but de réduire la charge des analystes et de prioriser efficacement les alertes. L'algorithme KGB, bien que non trivial à mettre en œuvre, démontre ici son potentiel en tant qu'outil d'aide à la supervision.