

THÈSE

En vue de l'obtention du
DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue par

Thi Quynh NGUYEN

Le 11 décembre 2023

Apprentissage automatique non supervisé pour la détection de trafics illégitimes

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Abdelmalek BENZEKRI et Romain LABORDE

Jury

M. Guy PUJOLLE, Rapporteur
M. David ESPES, Rapporteur
M. Romain LABORDE, Co-directeur de thèse
Mme Nathalie AUSSENAC-GILLES, Présidente

Remerciements

Je n'aurais jamais pu terminer ma thèse sans les conseils, l'aide et le soutien des personnes aimables qui m'entourent, à quelques-unes seulement qu'il est possible de mentionner ici en particulier.

Je voudrais tout d'abord remercier mes deux encadrants : messieurs **Abdelmalek Benzekri** et **Romain Laborde** qui m'ont donné l'opportunité de réaliser cette thèse. Sincères remerciements pour leur soutien, leur disponibilité et leur patience tout au long de cette thèse. C'est en leur compagnie que j'ai pu apprendre à renforcer mon esprit critique et mon sérieux.

Je tiens à remercie Monsieur **Guy Pujolle**, Professeur émérite à l'Université Sorbonne, pour l'honneur qu'il m'a fait en acceptant d'être rapporteur de mon travail et pour ses remarques qui m'ont permis d'améliorer la qualité de ce mémoire. Je tiens à lui exprimer mes remerciements pour l'honneur qu'il me fait en participant à ce jury.

Je tiens également à remercier Monsieur **David Espes**, Professeur à l'Université de Bretagne Occidentale, pour avoir accepté d'être rapporteur de mon travail et pour ses observations qui m'ont permis d'améliorer la qualité de ce mémoire. Je tiens à lui exprimer mes remerciements pour l'honneur qu'il me fait en participant à ce jury.

Je remercie Madame **Nathalie Aussenac Gilles**, Directrice de recherche, CNRS Toulouse – IRIT, pour l'intérêt qu'elle a porté à mes travaux en examinant ce mémoire, pour ses conseils avisés et pour l'honneur qu'elle me fait en participant à ce jury.

J'adresse un merci aux personnes à **AKKODIS**. Je les remercie pour leur aider, leur disponibilité, leurs conseils, leurs retours précieux. Cette thèse a été rendue possible grâce au financement d'AKKODIS.

Monsieur **Bruno Qu'hen**, Consultant MOA à AKKODIS, pour m'avoir accueilli au sein de l'entreprise à Paris et ses conseils sur mon travail.

Monsieur **Louis Garcia** et Monsieur **Charles Fortunet** pour m'avoir accueilli au sein de l'entreprise **MODIS** à Toulouse afin de pouvoir travailler à proximité de mes encadrants.

Monsieur **Mehdi Mounsif**, AI Tech Lead à AKKODIS, pour sa gentillesse ainsi que ses remarques et conseils lors de mon travail de thèse.

Monsieur **Benoit Baurens**, Delivery manager innovation à AKKODIS, pour m'avoir accueilli au sein de l'entreprise AKKODIS et nos petites discussions sur mon travail.

Je veux aussi remercier à tous les membres, ou anciens membres, de l'équipe SIERA que j'ai pu rencontrer pour leur soutien.

Je remercie spécialement mes parents, mes frères et sœurs pour leurs disponibilités et leurs encouragements.

Je remercie tout particulièrement mon mari et mes deux enfants pour son encouragement, son aide et sa grande patience.

Merci à toutes et à tous pour tout !

Résumé

Les cyber-attaques de plus en plus sophistiquées, préméditées et ciblées telles que les menaces persistantes avancées (APTs) peuvent être perpétrées sur des périodes de temps longues avant d'être divulguées ou découvertes. Pour cela, les attaquants mettent en œuvre des stratégies pour camoufler le plus longtemps possible leurs activités malveillantes comme la mise œuvre de canaux de communication entre des machines infectées et un serveur de commande et de contrôle (C&C) afin de pouvoir exfiltrer des données sensibles ou contrôler à distance des machines zombies. Une des techniques utilisées consiste à encapsuler le trafic C&C dans des protocoles réseau autorisés (comme le protocole du système de noms de domaine (DNS), le protocole de transfert hypertexte sécurisé (HTTPS), etc.) pour outrepasser les mécanismes de contrôle de sécurité. La détection de ces flux malicieux par les méthodes de détection traditionnelles, telles que les systèmes de gestion des événements et des informations de sécurité (SIEM), est limitée. Le trop grand nombre de paramètres à considérer pour définir manuellement des indicateurs fiables étant le principal frein.

Pour répondre à ce défi, nous proposons dans cette thèse une approche basée sur l'apprentissage automatique non supervisé et plus particulièrement les algorithmes de détection d'anomalies que nous appliquons à la détection de tunnels DNS. Le choix d'un algorithme d'apprentissage non supervisé est guidé par le coût trop élevé pour obtenir un jeu de données d'apprentissage exhaustif qui serait labélisé par des experts sécurité et qui est indispensable aux algorithmes d'apprentissage supervisé. Ensuite, les attaques que nous ciblons ont pour objectif de rester sous les seuils de détection. Par conséquent, les événements ou flux réseau malicieux seront rares.

Une première étude que nous avons menée nous a permis de mettre en avant l'algorithme DBSCAN (Density-Based Spatial Clustering of Applications with Noise). Cependant, DBSCAN exige de trouver expérimentalement les valeurs de deux hyper-paramètres. Afin d'automatiser la détection de tunnels DNS, nous proposons un algorithme amélioré appelé AutoRoC-DBSCAN qui peut déterminer automatiquement les valeurs de ces hyper-paramètres. Nous avons comparé ses performances avec 5 autres algorithmes d'apprentissage non supervisé (K-means, GMM, Isolation Forest, One-class SVM et LOF) sur deux jeux de données différents. Nous avons créé le premier jeu de données qui permet de vérifier la détection de tunnel DNS. Le deuxième jeu de données est CIRA-CIC-DoHBrw-2020 qui est fourni par le projet de l'Institut canadien pour la cybersécurité. Les expérimentations valident la détection de tunnels DNS over HTTPS où les flux malicieux sont doublement encapsulés par DNS puis par HTTPS. Les résultats obtenus lors de nos tests renforcent l'intérêt de notre approche.

Abstract

Increasingly sophisticated, premeditated and targeted cyber-attacks such as Advanced Persistent Threats (APTs) can be perpetrated over long periods of time before being disclosed or discovered. To do this, attackers implement strategies to camouflage their malicious activities as long as possible, such as the implementation of communication channels between infected machines and a command and control (C&C) server in order to exfiltrate sensitive data or remotely control zombie machines. One of the techniques used is to encapsulate the C&C traffic in authorized network protocols (such as the Domain Name System (DNS) protocol, the Secure Hypertext Transfer Protocol (HTTPS), etc.) to bypass security control mechanisms. The detection of these malicious flows by traditional detection methods, such as Security Information and Event Management (SIEM) systems, is limited. The main obstacle is the large number of parameters to consider in manually defining reliable indicators.

To address this challenge, we propose in this thesis an approach based on unsupervised machine learning and more specifically anomaly detection algorithms that we apply to the detection of DNS tunnels. The choice of an unsupervised learning algorithm is guided by the excessively high cost of obtaining a comprehensive learning dataset that would be labeled by security experts and which is essential for supervised learning algorithms. Then, the attacks we target aim to stay below detection thresholds. Therefore, malicious events or network flows will be rare.

An initial study we conducted allowed us to highlight the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm. However, DBSCAN requires experimentally finding the values of two hyperparameters. To automate the detection of DNS tunnels, we propose an improved algorithm called AutoRoC-DBSCAN which can automatically determine the values of these hyperparameters. We compared its performance with 5 other unsupervised learning algorithms (K-means, GMM, Isolation Forest, One-class SVM and LOF) on two different datasets. We created the first dataset that allows for the verification of DNS tunnel detection. The second dataset is CIRA-CIC-DoHBrw-2020 which is provided by the Canadian Institute for Cybersecurity project. The experiments validate the detection of DNS over HTTPS tunnels where malicious flows are doubly encapsulated by DNS then by HTTPS. The results obtained in our tests reinforce the interest in our approach.

Table des matières

REMERCIEMENTS **2**
RESUME **4**
ABSTRACT **5**
TABLE DES MATIERES **6**
TABLE DES FIGURES **9**
LISTE DES TABLEAUX **12**
INTRODUCTION **13**
1 PROBLEMATIQUE **13**
2 HYPOTHESE DE LA THESE **15**
3 STRUCTURE DE LA THESE **16**
CHAPITRE 1. SITUATION ACTUELLE DE LA CYBERSECURITE **18**
1 INTRODUCTION **18**
2 ILLUSTRATION DU TRAFIC AUTORISE ILLEGITIME **18**
2.1 TUNNEL DNS MALVEILLANT 19
2.2 TUNNEL DNS MALVEILLANT SUR HTTPS 21
2.3 OUTILS DE TUNNEL DNS 22
3 APT/C&C **22**
4 DIRECTIVE NIS 2 ..... **24**
4.1 GOUVERNANCE 25
4.2 PROTECTION 27
4.2.1 Zero Trust *27*
4.2.2 Defense-in-depth *28*
4.2.3 Pare-feu *28*
4.2.4 Pare-feu d'application web (Website Application Firewall - WAF) *29*
4.2.5 Virtual Private Network *30*
4.2.6 Gestion des identités et des accès (Identity and Access Management - IAM) *31*
4.3 DEFENSE 32
4.3.1 Système de détection d'intrusion (Intrusion Detection System - IDS) *32*
4.3.2 Analyse du comportement des utilisateurs et des entités (User and Entity Behavior Analytics - UEBA) *33*
4.3.3 HoneyPot *34*
4.3.4 Système de gestion des informations et des événements de sécurité (Security Information and Event Management - SIEM) *35*
4.3.5 Équipe d'intervention en cas d'urgence informatique (Computer Emergency Response Team - CERT) *35*
4.3.6 Centre d'opérations et de sécurité (Security Operation Center - SOC) *36*
4.3.7 MITRE ATT&CK *37*

4.4 RESILIENCE 38
5 ARCHITECTURE DETAILLEE 39
6 CONCLUSION 41
CHAPITRE 2. TECHNIQUES D'APPRENTISSAGE AUTOMATIQUE NON SUPERVISEES 43
1 INTRODUCTION 43
2 K-MEANS 43
3 GAUSSIAN MIXTURE MODELS (GMM) 44
4 ISOLATION FOREST 46
5 ONE-CLASS SUPPORT VECTOR MACHINE (ONE-CLASS SVM) 47
6 LOCAL OUTLIER FACTOR (LOF) 49
7 DENSITY-BASED SPATIAL CLUSTERING OF APPLICATIONS WITH NOISE (DBSCAN) 51
8 AUTO-ENCODEUR 53
9 SELF ORGANIZING MAP (SOM) 55
10 CONCLUSION 57
CHAPITRE 3. ÉTAT DE L'ART DES TECHNIQUES D'APPRENTISSAGE AUTOMATIQUE POUR LA DETECTION D'ATTAQUES RESEAU 59
1 INTRODUCTION 59
2 APPROCHES NON SUPERVISEES POUR DETECTER DES ATTAQUES DANS LE RESEAU 59
3 DETECTION DE COMMUNICATIONS C&C ENCAPSULEES DANS DNS 66
4 CONCLUSION 72
CHAPITRE 4. AUTOMATISATION DE DBSCAN POUR LA DETECTION DE TUNNELS DNS 73
1 INTRODUCTION 73
2 APPROCHE PROPOSEE UTILISANT DES TECHNIQUES D'APPRENTISSAGE AUTOMATIQUE NON SUPERVISEES 73
3 AUTOROC-DBSCAN 76
4 METHODE DE DETERMINATION AUTOMATIQUE DES PARAMETRES DE DBSCAN 80
5 CONCLUSION 82
CHAPITRE 5. ÉVALUATION DES PERFORMANCES 83
1 INTRODUCTION 83
2 DETECTION DES TUNNELS DNS 84
2.1 DESCRIPTION DU JEU DE DONNEES 84
2.2 PREPARATION 86
2.3 ÉVALUATION DE PERFORMANCE 87
2.4 ÉVALUATION DE TEMPS D'EXECUTION 95
3 DETECTION DES TUNNELS DNS-OVER-HTTPS (DOH) 97
3.1 INFRASTRUCTURES 97

3.2 PREPARATION 98
3.3 ÉVALUATION DE PERFORMANCE 98
3.4 ÉVALUATION DE TEMPS D'EXECUTION 106
4 VALIDATION DE LA VALEUR D'INITIALISATION DE AUTOROC-DBSCAN 108
5 CONCLUSION 112
CHAPITRE 6. CONCLUSION ET PERSPECTIVES 113
ACRONYMES 116
REFERENCES 120
ANNEXES 127

Table des figures

Figure 1. Processus de tunnel DNS malveillant.....19
Figure 2. Requête DNS normale.....20
Figure 3. Tunnel DNS malveillant généré par DNScat2.....20
Figure 4. Requête DNS Pisloader : envoi d'une balise [11].....20
Figure 5. Réponse DNS Pisloader : réponse TXT par le serveur C&C [11].....20
Figure 6. Processus de tunnel DNS malveillant sur HTTPS.....21
Figure 7. Cycle de la vie d'une APT [4].....23
Figure 8. Les règles de la directive NIS [22].....24
Figure 9. Une démarche itérative en 5 ateliers [24].....25
Figure 10. Architecture de la passerelle Internet sécurisée [46].....42
Figure 11. K-means.....44
Figure 12. Classification par K-means et GMM [50].....46
Figure 13. Isolation Forest.....47
Figure 14. One-class SVM.....49
Figure 15. Local Outlier Factor.....51
Figure 16. DBSCAN.....53
Figure 17. Réseaux de neurones artificiels (ANN) [59].....53
Figure 18. Créer un nouveau nœud [59].....53
Figure 19. Auto-Encodeur [57].....55
Figure 20. Calculer la distance entre le vecteur d'entrée et les nœuds [62].....57
Figure 21. La taille du voisinage de la BMU diminue après chaque itération [63].....57
Figure 22. Phases de la méthode proposée.....75
Figure 23. AutoRoC-DBSCAN partie 1.....78
Figure 24. AutoRoC-DBSCAN Partie 2.....79
Figure 25. AutoRoC-DBSCAN Partie 3.....80
Figure 26. Topologie du réseau utilisé pour capturer le trafic.....85
Figure 27. Comparaison des taux de détection dans notre jeu de données.....88
Figure 28. Comparaison des taux de faux positifs dans notre jeu de données.....88
Figure 29. Comparaison des taux de détection dans notre jeu de données – Chrome....91

Figure 30. Comparaison des taux de faux positifs dans notre jeu de données – Chrome 91
Figure 31. Comparaison des taux de détection dans notre jeu de données – Firefox 93
Figure 32. Comparaison des taux de faux positifs dans notre jeu de données – Firefox 93
Figure 33. Temps d'exécution sur notre jeu de données 95
Figure 34. Temps d'exécution sur notre jeu de données Chrome 95
Figure 35. Temps d'exécution sur notre jeu de données Firefox 96
Figure 36. Topologie de réseau utilisé pour capturer le trafic pour le jeu de données CIRA-CIC-DoHBrw-2020 [86] 97
Figure 37. Comparaison des taux de détection dans CIRA-CIC-DoHBrw-2020 99
Figure 38. Comparaison des taux de faux positifs dans CIRA-CIC-DoHBrw-2020 99
Figure 39. Comparaison des taux de détection dans CIRA-CIC-DoHBrw-2020 – Chrome 102
Figure 40. Comparaison des taux de faux positifs dans CIRA-CIC-DoHBrw-2020 – Chrome 102
Figure 41. Comparaison des taux de de détection dans CIRA-CIC-DoHBrw-2020 – Firefox 104
Figure 42. Comparaison des taux de faux positifs dans CIRA-CIC-DoHBrw-2020 – Firefox 104
Figure 43. Temps d'exécution sur CIRA-CIC-DoHBrw-2020 107
Figure 44. Temps d'exécution sur CIRA-CIC-DoHBrw-2020 Chrome 107
Figure 45. Temps d'exécution sur CIRA-CIC-DoHBrw-2020 Firefox 108
Figure 46. Comparaison des taux de détection lors du changement de *minPts* dans notre jeu de données 109
Figure 47. Comparaison des taux de faux positifs lors du changement de *minPts* dans notre jeu de données 109
Figure 48. Comparaison des taux de détection lors du changement de *minPts* dans notre jeu de données - Chrome 110
Figure 49. Comparaison des taux de faux positifs lors du changement de *minPts* dans notre jeu de données - Chrome 110
Figure 50. Comparaison des taux de détection lors du changement de *minPts* dans notre jeu de données - Firefox 110
Figure 51. Comparaison des taux de faux positifs lors du changement de *minPts* dans notre jeu de données – Firefox 110

Figure 52. Comparaison des taux de détection lors du changement de *minPts* dans CIRA-CIC-DoHBrw-2020 111

Figure 53. Comparaison des taux de faux positifs lors du changement de *minPts* dans CIRA-CIC-DoHBrw-2020 111

Figure 54. Comparaison des taux de détection lors du changement de *minPts* dans CIRA-CIC-DoHBrw-2020 - Chrome 111

Figure 55. Comparaison des taux de faux positifs lors du changement de *minPts* dans CIRA-CIC-DoHBrw-2020 - Chrome 111

Figure 56. Comparaison des taux de détection lors du changement de *minPts* dans CIRA-CIC-DoHBrw-2020 - Firefox 112

Figure 57. Comparaison des taux de faux positifs lors du changement de *minPts* dans CIRA-CIC-DoHBrw-2020 - Firefox 112

Liste des tableaux

Tableau 1. Approches non supervisées pour détecter des attaques dans le réseau.....64
Tableau 2. Approches de détection de communications C&C encapsulées dans DNS..70
Tableau 3. Nombre d'octets échangés des trafics normaux, Dnscat2 et Dns2tcp.....86
Tableau 4. Quantité de chaque type de trafic – notre base de données.....87
Tableau 5. Résultats de notre jeu de données 89
Tableau 6. Comparaison des résultats de notre jeu de données avec des taux d'attaque allant jusqu'à 3%.....90
Tableau 7. Résultats de notre jeu de données – Chrome.....92
Tableau 8. Résultats de notre jeu de données – Firefox 94
Tableau 9. Comparaison des temps d'exécution sur notre jeu de données 96
Tableau 10. Quantité de chaque type de trafic – CIRA-CIC-DoHBrw-2020 98
Tableau 11. Résultats de CIRA-CIC-DoHBrw-2020 100
Tableau 12. Comparaison des résultats de CIRA-CIC-DoHBrw-2020 avec des taux d'attaque allant jusqu'à 3% 101
Tableau 13. Résultats de CIRA-CIC-DoHBrw-2020-Chrome.....103
Tableau 14. Résultats de CIRA-CIC-DoHBrw-2020-Firefox 105
Tableau 15. Comparaison des temps d'exécution sur CIRA-CIC-DoHBrw-2020 108
Tableau 16. Caratéristiques du jeu de données 127

Introduction

Aujourd'hui, Internet compte environ 5 milliards d'utilisateurs dans le monde, soit 63% de la population mondiale totale [1]. Il est utilisé dans de nombreux domaines tels que l'économie, le commerce, la culture, le social et le gouvernement des pays [1]. Cependant, parallèlement au développement d'Internet, le nombre de cyberattaques s'est intensifié. 87% des organisations ont été victimes d'une cyberattaque au cours des trois dernières années [2]. Spécialement, le nombre de cyberattaques par semaine sur les réseaux d'entreprise a augmenté de 50% en 2021 par rapport à 2020 [3]. Les cybercriminels continuent de concevoir de nouvelles stratégies pour lancer des attaques de plus en plus sophistiquées et cette tendance s'aggravera dans le futur. Par conséquent, la protection d'une organisation contre les cyberattaques est un effort continu qui nécessite une vigilance constante.

1 Problématique

Loin de l'image du hackeur solitaire essayant de vandaliser des systèmes informatiques par opportunisme ou pour la gloire, les menaces persistantes avancées (Advanced Persistent Threats - APTs) [4] sont des attaques sophistiquées, préméditées et ciblées. Elles sont exécutées par des groupes d'attaquants structurés et hautement qualifiés combinant diverses compétences pour atteindre des objectifs malicieux spécifiques et prédéterminés. Les attaques APTs ne cherchent pas forcément un résultat visible et immédiat. Ces attaques visent plutôt à rester non détectées pendant longtemps afin de compromettre un système sur le long terme. Ainsi, la complexité des attaques APTs les rend difficiles à identifier.

Pour mieux comprendre ces attaques complexes, MITRE a proposé ATT&CK [5], une base de connaissances open source sur les tactiques, techniques et procédures (Tactics, Techniques and Procedures - TTPs) de l'adversaire construite sur des observations du monde réel. MITRE ATT&CK contient 14 tactiques et 222 techniques différentes couvrant l'ensemble du processus d'attaque. Les tactiques sont les objectifs qu'un attaquant vise à atteindre. MITRE ATT&CK répertorie 14 tactiques : reconnaissance, développement, accès initial, exécution, persistance, escalade de privilèges, évasion de la défense, accès aux identifiants, découverte, mouvement latéral, collecte, commande et contrôle (Command and Control - C&C), exfiltration, et impact. Pour atteindre ces objectifs, les attaquants appliquent une ou plusieurs techniques. Par exemple, APT32 [6] a ciblé plusieurs industries du secteur privé dans des pays d'Asie du Sud-Est comme le Vietnam, le Laos et le Cambodge. Il a utilisé 55 techniques de menace différentes pour attaquer leur cible. Lors de l'étude d'APT32, il a été démontré que les attaquants ont la fonctionnalité de tunnel de commande et contrôle (C&C) de

Cobalt Strike [7] pour masquer le trafic malveillant dans le trafic réseau autorisé DNS, HTTP et HTTPS.

La mise en place d'un canal de communication avec un serveur commande et contrôle (C&C) peut avoir deux objectifs : 1) exfiltrer des informations sensibles hors du réseau attaqué mais aussi 2) contrôler à distance le(s) logiciel(s) malveillant(s) en envoyant des ordres ou des mises à jour. Une stratégie courante pour dissimuler ce canal de communication consiste à encapsuler le trafic C&C dans des protocoles réseaux autorisés pour outrepasser les contrôles de sécurité comme dans le cas de l'APT32.

Le protocole DNS, étant autorisé et indispensable au bon fonctionnement des réseaux, est un moyen simple de déployer un canal de communication avec un serveur C&C. En effet, le protocole DNS, qui a été initialement conçu pour traduire des noms de domaine en adresses IP, nécessite une communication entre les équipements du réseau interne et les équipements externes au réseau de l'entreprise. En conséquence, les systèmes d'information de sécurité des entreprises doivent autoriser les flux de données DNS, ce qui fait du DNS un bon candidat pour masquer les communications C&C en encapsulant les messages illicites dans le flux DNS aussi appelé tunnel DNS. Il est alors possible d'utiliser le fonctionnement du protocole DNS pour distribuer les messages illicites jusqu'à un serveur C&C. Normalement, les entreprises et les organisations maintiennent des listes noires de noms de domaine malveillants connus comme l'un des indicateurs de compromission. Si une requête DNS est dirigée vers l'un de ces domaines, une alerte sera immédiatement activée. Toutefois, il est simple de réserver un nom de domaine et de le fermer aussi rapidement. Par conséquent, ce type de détection n'est pas fiable.

Ajouté à cela, un nouveau protocole, appelé DNS-over-HTTPS (DoH) [8], a été proposé en 2018 pour améliorer la protection de la vie privée des personnes naviguant sur Internet en encapsulant les requêtes et les réponses DNS dans les connexions HTTPS. Bien que cette spécification améliore la sécurité du DNS en chiffrant les messages, DoH offre également de nouvelles opportunités aux développeurs de logiciels malveillants pour cacher leurs messages DNS commande et contrôle (C&C) dans un trafic chiffré et autorisé. Cette double encapsulation est d'autant plus difficile à détecter que le résolveur DNS est dans ce cas le serveur DoH qui se trouve en dehors du réseau de l'entreprise. Les équipements de sécurité ne peuvent donc ni inspecter le contenu des messages DNS, ni connaître l'adresse IP des serveurs DNS interrogés. Il est alors impossible de mettre en place des listes noires.

Actuellement, pour prévenir les cyber-attaques, les entreprises, les organisations ou les gouvernements possèdent des systèmes de gestion des événements et des informations de sécurité (Security Information and Event Management - SIEM) [9] pour protéger leur réseau. Les SIEM combinent la gestion des informations de sécurité (Security Information Management - SIM) et la gestion des événements de sécurité

(Security Event Management - SEM) pour permettre la surveillance des événements du système d'information, détecter des incidents, menaces en temps réel en collectant et en intégrant différentes sources de journaux et d'événements (journaux des applications, journaux des systèmes, journaux des dispositifs de sécurité, etc.). Ils recueillent également des informations sur les cyberattaques réalistes telles que les adresses IP suspectes, les activités provenant de régions géographiques étranges, les demandes excessives sur des fichiers importants, etc. Ensuite, ils stockent ces informations comme des indicateurs de compromission (Indicator of Compromise - IoC) et des indicateurs d'attaque (Indicator of Attack - IoA). Les équipes de sécurité rédigent alors les règles de détection en se basant sur ces IoC/IoA. Lorsqu'un SIEM identifie une menace via la surveillance de la sécurité du réseau, il génère une alerte basée sur ces règles prédéterminées. Cependant, écrire des règles de détection pour détecter les communications C&C basées sur le DNS est une tâche extrêmement difficile qui nécessite de combiner de nombreux IoC très complexes pour lesquels il faut déterminer un seuil spécifique. Comme les APTs sont exécutées par des attaquants qualifiés, les paramètres d'attaque sont contrôlés afin de passer sous ces seuils pour que les attaques ne soient pas détectées.

2 Hypothèse de la thèse

Dans cette thèse, nous voulons faire face aux difficultés pour détecter des attaques complexes qui sont organisées par des groupes criminels de grande qualité. Dans cette thèse, nous démontrons l'efficacité des algorithmes d'apprentissage automatique dans la détection de technique de tunnel DNS. Les techniques d'apprentissage automatique peuvent aider dans une tâche de détection aussi complexe parce qu'elles permettent de prendre en compte et combiner de nombreux critères de détection. Il existe principalement deux grandes catégories de techniques d'apprentissage automatique : les méthodes supervisées et celles non supervisées. Les méthodes supervisées peuvent apprendre des modèles à partir de jeux de données d'apprentissage qui comprennent les caractéristiques d'entrée et les résultats attendus. Les algorithmes supervisés sont efficaces pour des tâches spécifiques telles que la détection d'images. Cependant, cette approche présente deux inconvénients dans notre contexte. Premièrement, les performances de l'algorithme dépendent de l'exhaustivité du jeu de données d'apprentissage. Un algorithme entraîné avec un ensemble spécifique de techniques de menaces ne peut pas détecter les menaces inconnues. De plus, un algorithme entraîné sera aussi spécifique à l'organisation où le jeu de données d'entrainement a été capturé car le jeu de données doit comprendre des trafics légitimes. Par exemple, le réseau d'une entreprise va donner un type de jeu de données incluant les trafics réseau correspondant aux activités des développeurs, des secrétaires, des chefs de projet, etc de cette entreprise. Le réseau d'une université va donner un autre type de jeu de données avec les trafics des étudiants, des chercheurs, des secrétaires, des enseignants, etc. Le deuxième problème est lié au coût de l'étiquetage des ensembles de données.

Un grand nombre de jeux de données d'images existe car l'étiquetage des images ne nécessite pas de compétences spécifiques. Dans le cas de la sécurité réseau, le contexte est complètement différent. L'étiquetage du trafic de sécurité réseau ne peut être effectué que par des experts en sécurité, ce qui augmente le coût de cette tâche de manière exponentielle. Au contraire, les méthodes non supervisées ne nécessitent pas de jeu de données pour l'entraînement, ce qui résout les deux inconvénients mentionnés précédemment. Cependant, il est plus difficile d'obtenir les mêmes performances que les méthodes supervisées.

En conséquence, notre objectif est de proposer une approche d'analyse comportementale basée sur des techniques d'apprentissage automatique non supervisé, en particulier le Density-Based Spatial Clustering of Applications with Noise (DBSCAN) pour la détection de technique de tunnelisation DNS. Nous nous sommes particulièrement intéressés à l'algorithme de détection d'anomalies car nous considérons que les attaquants vont tenter de rester sous les seuils de détection et si le trafic malicieux devient important les outils de détection classiques seront capables de le déceler. Cependant, DBSCAN nécessite de trouver expérimentalement les valeurs de deux hyperparamètres. Nous proposons dans cette thèse un algorithme nommé AutoRoC-DBSCAN, qui calcule automatiquement ces hyperparamètres afin d'automatiser la détection des tunnels DNS malveillants. Nous avons comparé la performance d'AutoRoC-DBSCAN à celles de cinq autres algorithmes d'apprentissage automatique non supervisés : K-means, Gaussian Mixture Model (GMM), Isolation Forest, One-class Support Vector Machine (One-class SVM) et Local Outlier Factor (LOF). Ce travail a été validé avec deux jeux de données différents. Nous avons créé un ensemble de données pour évaluer les connexions de tunnel DNS en raison du manque d'ensembles de données de qualité pour évaluer les connexions de tunnel DNS. Le deuxième ensemble de données est fourni par le projet de l'Institut canadien pour la cybersécurité, CIRA-CIC-DoHBrw-2020 pour les connexions par tunnel DNS sur HTTPS.

3 Structure de la thèse

Cette thèse s'organise en 6 chapitres, chacun correspondant à une étape de l'argumentaire.

Le chapitre 1 présente la situation actuelle de la cybersécurité. Les attaques complexes comme des APTs aiment utiliser des canaux C&C pour envoyer les flux autorisés mais illégitimes. Nous rappelons les protocoles DNS et DNS-over-HTTPS (DoH) et illustrons comment cacher ces canaux commande et contrôle (C&C) sur ces protocoles. Pour protéger le réseau contre les attaques, nous introduisons la directive NIS2 et quelques outils de protection et de défense. De plus, nous présentons une architecture multi-services simple composée d'un SI interne, d'une passerelle Internet

sécurisée et d'un SI administration. À partir de cette architecture, chacun peut l'adapter à son réseau.

Les attaques complexes cachent souvent les canaux C&C dans les protocoles autorisés et ainsi ces flux illégitimes ressemblent à un flux authentique. Ainsi, une analyse approfondie avec un grand nombre de critères est nécessaire. De plus, les attaques complexes sont normalement contrôlées par des attaquants hautement qualifiés afin de rester sous les seuils de détection des outils. Pour détecter ces flux illégitimes, dans le chapitre 2, nous présentons les algorithmes de détection des valeurs aberrantes tels que K-means, Gaussian Mixture Model (GMM), Isolation Forest, One-class SVM, Local Outlier Factor (LOF), Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Auto-Encodeur, et Self Organizing Map (SOM).

Le chapitre 3 présente les travaux antérieurs relatifs à l'utilisation d'approches d'apprentissage automatique pour la détection d'attaques réseau.

Le chapitre 4 présente notre approche basée sur l'algorithme DBSCAN. Cependant, DBSCAN demande de prédéfinir deux hyperparamètres qui impactent fortement la performance de l'algorithme. Nous proposons un algorithme qui optimise les valeurs de ces deux hyperparamètres.

Le Chapitre 5 présente la comparaison de performance de notre approche basée sur DBSCAN avec 5 autres algorithmes sur deux jeux de données : notre jeu de données de tunnels DNS et le jeu de données CIRA-CIC-DoHBrw-2020.

Le Chapitre 6 présente la conclusion de cette thèse et les perspectives de futurs travaux d'utilisation et d'extensions de l'approche proposée.

Chapitre 1. Situation actuelle de la cybersécurité

1 Introduction

Aujourd'hui, l'Internet se développe de plus en plus vite. Les services en ligne sont de plus en plus diversifiés pour répondre à tous les besoins de la vie tels que le télétravail, l'apprentissage, le shopping, ou le divertissement. Ainsi, la confidentialité, en particulier la protection de la vie privée, est un grand défi. Pour augmenter la confidentialité, les trafics sont chiffrés et le chiffrement est de bout en bout. Seules les extrémités ont accès au trafic en clair. Comme le trafic est chiffré, un observateur ne peut pas comprendre l'objet de l'échange. C'est ce que cherchent à faire des attaquants hautement qualifiés en créant des attaques complexes comme des APTs qui utilisent souvent des canaux C&C pour envoyer des ordres ou exfiltrer des données. Ces canaux C&C sont souvent cachés dans un trafic autorisé tel que le protocole DNS ou encore plus dur lorsque le protocole DNS est encapsulé dans un protocole HTTPS chiffré. La question est donc de comment détecter ces flux autorisés illégitimes.

Dans ce chapitre, nous rappelons les protocoles DNS et DNS-over-HTTPS (DoH) et illustrons comment créer les flux autorisés illégitimes sur ces protocoles. Ensuite, nous décrivons le cycle de vie des APTs en nous focalisant sur l'établissement d'un canal C&C pour les trafics autorisés illégitimes. Pour contrer des attaques et donner des mécanismes de protection et des moyens pour détecter des attaques au plus tôt, la directive NIS2 (Network and Information Security 2 – NIS2) est proposée pour réduire les surfaces d'attaque et définir les exigences en matière de sécurité informatique envers les organismes critiques – OSE ou Opérateur de services essentiels. Le problème que nous traitons dans cette thèse est la défense contre les attaques, plus spécifiquement la détection de commande et contrôle (C&C).

2 Illustration du trafic autorisé illégitime

Nous allons expliquer et illustrer ce qu'est un trafic C&C autorisé illégitime et comment nous pouvons mettre en place un canal C&C à l'aide des protocoles DNS et DoH.

C&C [10] est une infrastructure qui contient l'ensemble des outils et techniques utilisés par les attaquants pour maintenir la communication avec des appareils compromis. Le C&C comprend un ou plusieurs canaux de communication entre les appareils d'un réseau victime et la plateforme contrôlée par l'attaquant. Ces canaux de communication sont utilisés pour envoyer des ordres aux appareils compromis et exfiltrer les données vers l'adversaire. Une stratégie courante est d'utiliser les types de

trafic légitime qui sont autorisés dans l'entreprise ciblée, comme les trafics DNS ou HTTPS.

2.1 Tunnel DNS malveillant

Le système de noms de domaine (DNS) est une partie importante d'Internet, fournissant un moyen de faire correspondre les noms des machines à leurs adresses IP (Internet Protocol). DNS est un protocole de la couche application encapsulé dans le protocole UDP (User Datagram Protocol).

Les attaquants utilisent des techniques de tunnel DNS pour éviter d'être détectés par les systèmes de sécurité (Figure 1). Le trafic du tunnel DNS est très similaire au trafic DNS authentique car il utilise le même port 53 et respecte la structure des messages DNS (Figure 2 et Figure 3). Le tunnel DNS chiffre les messages dans les requêtes ou réponses DNS. Pour transférer des informations de la machine infectée vers le serveur C&C, le malware sur la machine infectée n'a qu'à faire une requête DNS vers un nom de domaine spécifique contrôlé par les attaquants (par exemple, XYZ@attaquant.fr) et envoyer le message chiffré dans la requête DNS. Le résolveur DNS de l'organisation attaquée obtient l'adresse IP du serveur DNS gérant le nom de domaine en utilisant la fonction de protocole DNS. En fait, ce serveur DNS est le serveur C&C. Pour transférer les commandes du serveur C&C, le faux serveur DNS les chiffre et les envoie dans la réponse. Par conséquent, exfiltrer des données hors du réseau infecté consiste à envoyer une ou plusieurs requêtes DNS. Le contrôle ou la mise à jour à distance des malwares se fait en masquant les commandes C&C dans les réponses DNS. Dans ce cas, les machines infectées doivent envoyer des messages balises dans les requêtes DNS pour rendre ces réponses DNS non suspectes. En utilisant cette approche, les attaquants peuvent créer leur propre protocole malveillant à l'aide des échanges DNS.

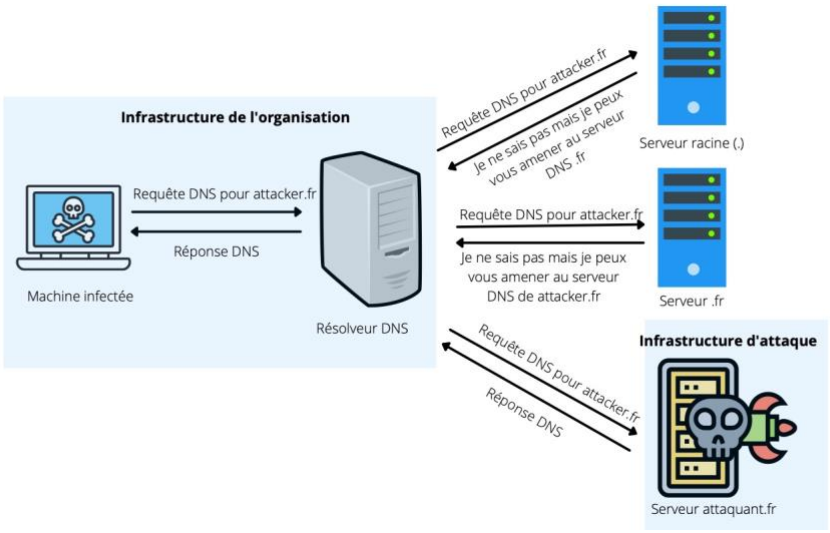


Figure 1. Processus de tunnel DNS malveillant

```
> Frame 9: 90 bytes on wire (720 bits), 90 bytes captured (720 bits)
> Ethernet II, Src: PcsCompu_0a:37:61 (08:00:27:0a:37:61), Dst: PcsCompu_2b:1e:ba (08:00:27:2b:1e:ba)
> Internet Protocol Version 4, Src: 192.168.2.10, Dst: 192.168.1.30
> User Datagram Protocol, Src Port: 48662, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0xac78
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  v Queries
    v accounts.google.com: type A, class IN
        Name: accounts.google.com
        [Name Length: 19]
        [Label Count: 3]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
```

Figure 2. Requête DNS normale

```
> Frame 9: 134 bytes on wire (1072 bits), 134 bytes captured (1072 bits)
> Ethernet II, Src: PcsCompu_8a:92:fe (08:00:27:8a:92:fe), Dst: PcsCompu_03:a7:79 (08:00:27:03:a7:79)
> Internet Protocol Version 4, Src: 192.168.2.20, Dst: 192.168.1.30
> User Datagram Protocol, Src Port: 38036, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0xabcc
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v bfbc019aa7a4b67151e29a0003558f307edd88ded38f4340266c729970b2.dnscat2.local: type TXT, class IN
        Name: bfbc019aa7a4b67151e29a0003558f307edd88ded38f4340266c729970b2.dnscat2.local
        [Name Length: 74]
        [Label Count: 3]
        Type: TXT (Text strings) (16)
        Class: IN (0x0001)
```

Figure 3. Tunnel DNS malveillant généré par DNScat2

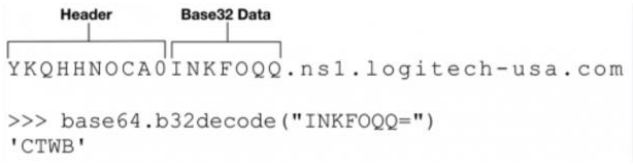


Figure 4. Requête DNS Pisloader : envoi d'une balise [11]



Figure 5. Réponse DNS Pisloader : réponse TXT par le serveur C&C [11]

La Figure 4 est un exemple de message balise envoyé par le malware Pisloader [11]. Il est composé d'une chaîne aléatoire en majuscule de 4 octets qui est utilisée comme charge utile. Le serveur C&C répond par un ordre à la machine infectée dans un enregistrement TXT. Dans le cas de la Figure 5, l'ordre est *sifo*, ce qui signifie collecter

des informations sur le système infecté. Le serveur C&C peut envoyer différents types de commandes, telles que : *drive* qui signifie lister les disques sur la machine infectée, *list* pour afficher les informations des fichiers dans le répertoire fourni, *upload* pour télécharger un fichier sur la machine infectée, etc. Après avoir reçu la commande, la machine infectée renverra les informations demandées par le serveur C&C.

2.2 Tunnel DNS malveillant sur HTTPS

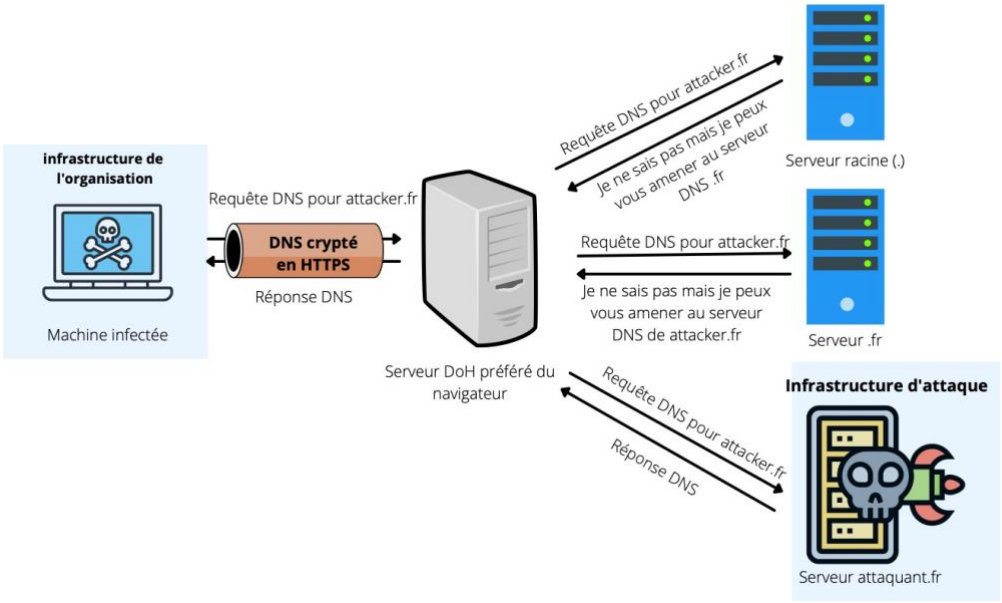


Figure 6. Processus de tunnel DNS malveillant sur HTTPS

DNS over HTTPS (DoH) [8] chiffre le trafic DNS en l'encapsulant dans le protocole HTTPS. Il a été développé pour augmenter la confidentialité et la sécurité des utilisateurs en empêchant les écoutes clandestines et les attaques pendant le transit des données. En effet, une requête DoH est envoyée à un serveur DoH (tel que Google, Cloudflare, etc.) qui est comme un résolveur DNS en fournissant un service de résolution de nom avec une politique de confidentialité pour l'utilisateur (Figure 6). La connexion entre le client DoH et le serveur est chiffrée par Transport Layer Security (TLS). De plus, comme la requête DNS est effectuée par un serveur DoH, aucune information concernant l'utilisateur qui a effectué la requête DoH n'est divulguée. Mozilla Firefox est le premier navigateur à l'implémenter et maintenant, la plupart des navigateurs populaires utilisent DoH comme Google Chrome, Microsoft Edge, etc. Bien que DoH soit un service basé sur le chiffrement pour protéger la confidentialité des utilisateurs, il présente également plusieurs problèmes de sécurité parce qu'il est impossible de vérifier le trafic DNS, ce qui rend difficile la détection de menaces.

Bien que DoH augmente la protection de la confidentialité des utilisateurs, il facilite également les communications C&C en fournissant un canal crypté gratuit et ouvert. De plus, les serveurs publics DoH étant bien connus, il n'est pas opportun de les mettre sur liste noire pour arrêter les attaques.

2.3 Outils de tunnel DNS

Plusieurs utilitaires de tunneling DNS ont été implémentés tels que DNScat2, DNS2tcp, Iodine, etc.

- DNScat2 [12] aide à créer un canal de C&C chiffré sur le protocole DNS. Il peut envoyer n'importe quelle donnée sur ce canal : charger et télécharger des fichiers, exécuter un shell. DNScat2 contient deux parties : le client et le serveur. Le client s'exécute sur une machine compromise et peut fonctionner sur la plupart des systèmes d'exploitation sans aucune installation spéciale. Le serveur peut s'exécuter sur un serveur DNS faisant autorité. Cependant, il nécessite un domaine où il écoutera les connexions des clients compromis. Lorsqu'il reçoit du trafic pour ce domaine, il tente d'établir une connexion stable.
- DNS2tcp [13] est un outil pour relayer les connexions TCP sur DNS. Aucune configuration spécifique n'est nécessaire car l'encapsulation s'effectue via TCP. DNS2tcp contient deux parties : le client et le serveur. Le serveur a une liste de ressources spécifiées dans un fichier de configuration. Le client écoute sur un port TCP prédéfini et relaie chaque connexion entrante via DNS au service final. Lorsque les connexions sont reçues sur un port spécifique, tous les trafics TCP sont envoyés au serveur DNS2tcp et transmis à un hôte et un port spécifique.
- Iodine [14] est un outil pour tunneliser le trafic de la version 4 du protocole Internet (IP) sur le protocole DNS pour passer le pare-feu et la sécurité du réseau en évitant la détection. Il a un composant serveur et client, et il est recommandé d'exécuter la même version aux deux extrémités.

3 APT/C&C

Une menace persistante avancée ou APT [4] est une cyber-attaque prolongée et ciblée par laquelle une organisation criminelle accède frauduleusement au réseau et système d'information d'une organisation privée ou publique sur une longue période. Une attaque APT est généralement destinée à surveiller l'activité de l'organisation ciblée afin de voler des informations à des adversaires ou encore voler de l'argent. Le cycle de vie d'une APT est plus long et plus complexe que les types d'attaques classiques. Ce cycle de vie peut être modélisé en 12 étapes [15] comme le montre la Figure 7. Dans l'étape 7 (intrusion initiale) et l'étape 8 (initialiser la connexion sortante), après toute la préparation et une fois l'accès à la cible établi, un logiciel malveillant est installé permettant de créer un tunnel C&C pour les futures communications entre les appareils du réseau victime et le serveur C&C de l'attaquant. Ensuite, l'attaquant exploite d'autres vulnérabilités pour renforcer sa position dans le réseau et commence à recueillir les données et les transférer vers le système de l'attaquant via le tunnel C&C. Il est très difficile de détecter ces trafics C&C parce que les attaques APTs sont

souvent contrôlées pour rester sous les radars et sous les seuils de détection des outils traditionnels.

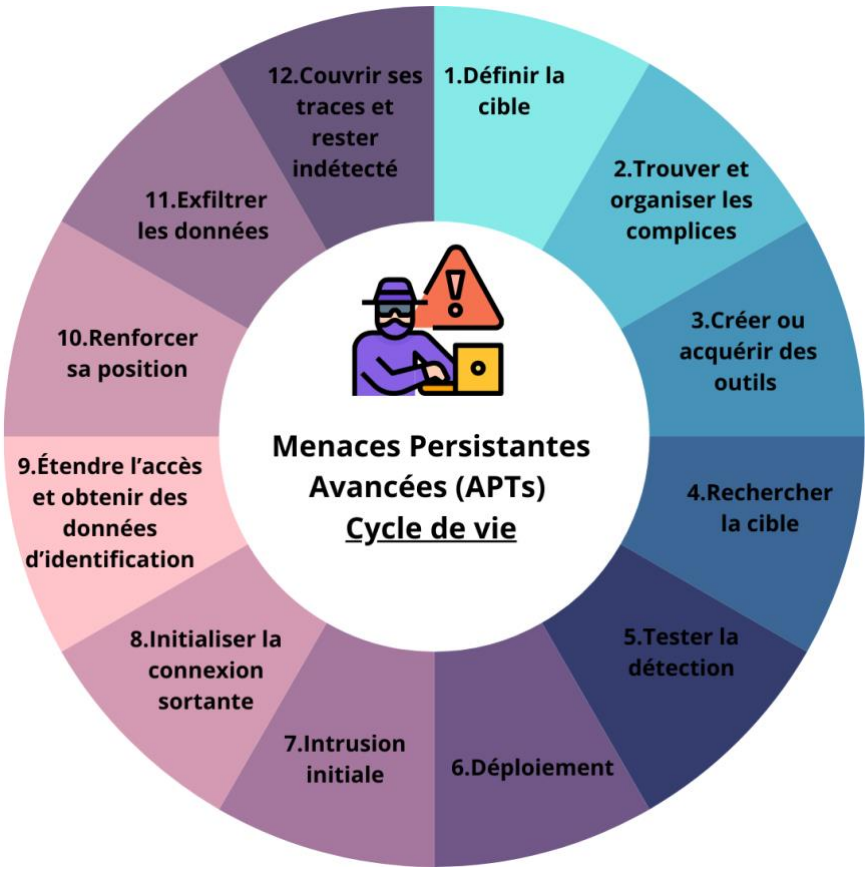


Figure 7. Cycle de la vie d'une APT [4]

Compte tenu des temps, des efforts et des ressources pour mener des attaques APTs, ces attaques sont exécutées par des groupes d'attaquants hautement qualifiés combinant plusieurs compétences (connaissance des différents systèmes d'exploitation, compétences en cryptographie, compétences en programmation, compétences en base de données, compétences en réseaux informatiques, etc.) pour pouvoir atteindre leurs objectifs. Souvent à la solde d'organisations criminelles voire de certains états, ils ciblent des entreprises et des gouvernements [16]. Par exemple, APT27 [17] du groupe GOBLIN PANDA (Chine) cible les secteurs de la défense, de l'énergie, et des pouvoirs publics en Asie du Sud-Est, et plus particulièrement au Vietnam. APT28 [18] du groupe FANCY BEAR (Russie) s'en est pris à des instances politiques américaines, à des organisations militaires européennes et à des acteurs de différents secteurs à travers le monde. APT34 [19] du groupe HELIX KITTEN (Iran) cible les entreprises des secteurs de l'aérospatiale, de l'énergie, de la finance, des services publics, de l'hôtellerie et des télécommunications. Le groupe Lockbit a revendiqué la cyberattaque contre l'entreprise française « Voyageurs du monde », le 16 mai 2023. Dans le rapport Q3 de 2022 de Kaspersky [20], les campagnes APTs sont étendues et déployées dans de nombreux pays tels que la Corée, l'Europe, les

États-Unis, l'Afrique du Sud, le Brésil, le Moyen-Orient et dans diverses régions d'Asie. Leurs cibles sont aussi diverses : des organismes gouvernementaux et diplomatiques, des entrepreneurs de la défense, le secteur financier, le secteur du matériel technologique et des semi-conducteurs, des organisations de recrutement informatique et des équipes de développement de plateformes de jeu en ligne.

4 Directive NIS 2

Pour pouvoir réduire les surfaces d'attaque et répondre à la problématique de cybersécurité, l'Union européenne (UE) a proposé la directive sur la sécurité des réseaux et systèmes d'information NIS2 [21] pour renforcer la cybersécurité des États membres de l'UE. La Commission Européenne définit un ensemble de critères simplifiés permettant de déterminer à qui s'adresse la directive : les opérateurs de services essentiels (OSE) ou fournisseurs de service numérique (FSN) [22]. La directive définit également les exigences de sécurité qui s'appliqueront aux OSE et FSN.

NIS 2 favorise l'échange d'informations entre les États membres. Elle encourage la « divulgation coordonnée des vulnérabilités ». Les pirates éthiques peuvent signaler les vulnérabilités, afin qu'elles soient diagnostiquées et corrigées. À cette fin, une base de données des vulnérabilités connues doit être tenue par l'Agence européenne pour la cybersécurité (ENISA).



Figure 8. Les règles de la directive NIS [22]

NIS2 contient 23 règles [23] qui sont réparties en 4 grands axes (Figure 8) : la gouvernance, la protection, la défense, et la résilience.

4.1 Gouvernance

La gouvernance ou le pilotage sert à comprendre ce qui se passe au niveau d'un système d'information comme qui fait quoi, quand et comment. Pour minimiser les risques, les équipes de gouvernance prennent des mesures pour bien comprendre le système et les risques auxquels il est exposé en réalisant des activités d'analyse de risque, d'audit de la sécurité, de définition de politique de sécurité ou encore de mise en place d'indicateurs de sécurité. Nous présentons la méthode EBIOS Risk Manager pour la gouvernance ci-après. Cette méthode permet de réduire la surface d'attaque et de rendre le système plus sûr.

Exemples d'outils pour la gouvernance : La méthode EBIOS RM

L'Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI) et le Club EBIOS ont proposé une méthode d'évaluation et de traitement des risques numériques (gouvernance) : EBIOS Risk Manager (EBIOS RM) [24]. EBIOS RM permet d'apprécier les risques numériques et d'identifier les mesures de sécurité à mettre en œuvre pour les maitriser. Elle permet aussi de valider le niveau de risque acceptable et de s'inscrire à plus long terme dans une démarche d'amélioration continue.

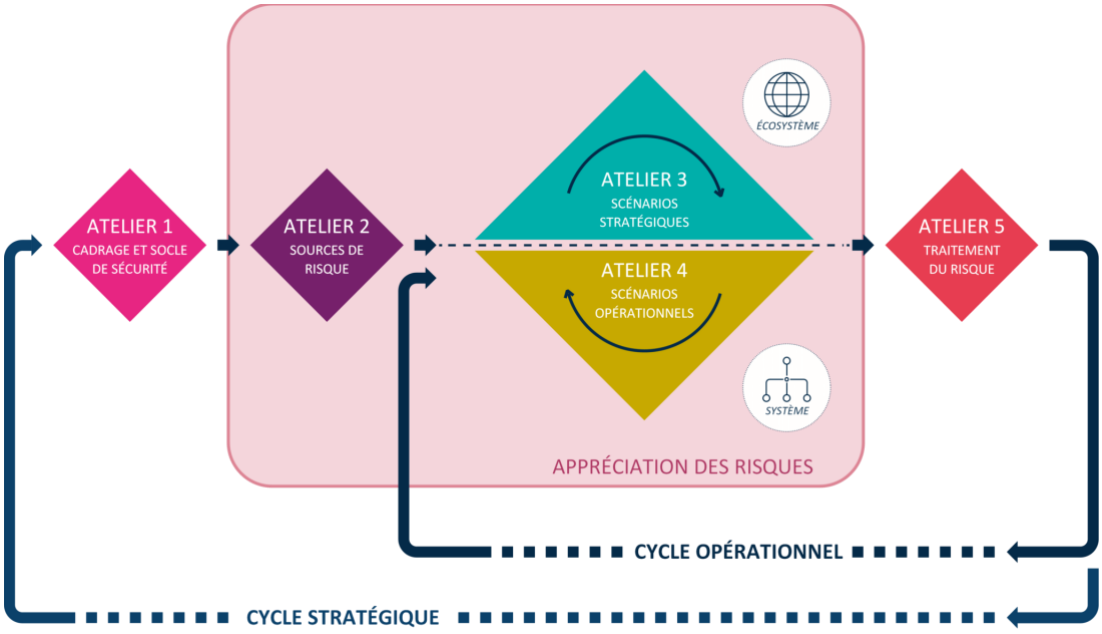


Figure 9. Une démarche itérative en 5 ateliers [24]

EBIOS RM est une méthode en 5 ateliers (Figure 9).

- Atelier 1. Cadrage et socle de sécurité

Ce premier atelier a pour but de créer les bases de l'analyse. Au début, on identifie les objectifs, le cadre et les participants associés. Ensuite, on décrit les missions de l'objet d'étude, ses biens supports et ses valeurs métiers. Finalement, on peut préciser les référentiels de sécurité appliqués et les justificatifs d'écarts potentiels connus.

- Atelier 2 : Sources de risque

Dans le deuxième atelier, on réalise une étude sur les sources de risque (SR) et les objectifs visés (OV), afin de trouver les couples SR/OV jugés les plus pertinents qui sont retenus au terme de cet atelier. Ils seront utiles à la construction des scénarios des ateliers 3 et 4.

Cette évaluation des risques s'appuie sur plusieurs critères :

  - La motivation des sources de risque.
  - Les ressources et l'activité des sources de risque.
  - L'historique des sources de risque (modes opératoires, faits d'armes…).

- Atelier 3 : Scénarios stratégiques

L'objectif de l'atelier 3 est de disposer d'une vision claire de l'écosystème, afin d'identifier les parties prenantes les plus vulnérables. L'écosystème comprend l'ensemble des parties prenantes qui sont autour de l'objet d'étude et concourent à la réalisation de ses missions (partenaires, sous-traitants, filiales, etc.)

À partir des résultats de l'atelier 2, on peut établir une cartographie des sources de risque. Ceci va permettre de bâtir des scénarios stratégiques. Ils décrivent les chemins d'attaque qu'une source de risque est susceptible d'emprunter pour atteindre son objectif (i.e., un des couples SR/OV sélectionnés lors de l'atelier 2).

- Atelier 4 : Scénarios opérationnels

Dans l'atelier 4, on établit des scénarios techniques reprenant les modes opératoires susceptibles d'être utilisés par les sources de risque pour réaliser les scénarios stratégiques et on les représente graphiquement sous la forme de scénarios opérationnels.

Une fois ces scénarios conçus, on va évaluer la vraisemblance. La méthode propose différents modes pour évaluer la vraisemblance : de la plus directe en cotant le scénario globalement, à la plus détaillée en prenant en compte la probabilité de succès et la difficulté de réalisation de chaque action élémentaire.

Enfin, une fois la vraisemblance affectée à chaque scénario opérationnel, une vision générale du risque initial est proposée à travers une cartographie globale.

- Atelier 5 : Traitement du risque

  Le but de cet atelier est de réaliser une synthèse des scénarios de risque identifiés et de définir une stratégie de traitement du risque.

  Pour chaque scénario de risque, on identifie des seuils d'acceptation du risque et un niveau de sécurité à atteindre en cas de non-acceptation. Cette décision se formalise dans la stratégie de traitement du risque.

  Une fois la stratégie de traitement validée pour chaque scénario, on définit les mesures de sécurité associées pour le traiter et on les recense dans un plan d'amélioration continue de la sécurité (PACS). À chaque mesure sont associés le responsable, les principaux freins et difficultés de mise en œuvre, le coût et l'échéance.

La méthode EBIOS RM permet d'analyser les risques et de définir les mesures associées pour les traiter mais ce n'est pas suffisant parce qu'elles ne permettent pas de lutter contre les trafics autorisés mais illégitimes. Nous avons besoin d'autres outils de protection.

4.2 Protection

La gouvernance nous permet de comprendre notre système et analyser les risques auxquels il est exposé. Il est alors nécessaire de mettre en place des mesures pour réduire la surface d'attaque et protéger notre système d'information. Ainsi, nous avons besoin de mettre en place des mesures de sécurité périmétrique comme des solutions de gestion des identités et des accès etc. L'objectif est de protéger les données et les informations des clients, d'assurer la sécurité des données partagées et garantir la fiabilité de l'accès, des performances réseau et de la protection contre les cybermenaces. Nous présentons quelques outils pour la protection ci-dessous.

Exemples d'outils pour la protection :

4.2.1 Zero Trust

Zero Trust [25] [26] est un modèle de sécurité basé sur le principe que personne ne bénéficie d'une confiance par défaut et n'est autorisé à accéder aux actifs de l'entreprise tant que les accès n'ont pas été validés comme légitimes et autorisés. L'accès doit être authentifié en permanence.

Zero Trust contient trois principes fondamentaux :

- Accorder le moins de privilèges et de droits d'accès possibles sans affecter la capacité d'un individu à effectuer ses tâches (principe de moindre privilège).
- Vérifier l'identité de l'utilisateur pour chaque nouvelle entrée dans le système ou demande d'accès à de nouvelles données.

- Toujours surveiller et évaluer toutes les actions entreprises au sein de l'infrastructure de l'organisation : le comportement des utilisateurs, les mouvements de données, et les changements de réseau.

Basées sur le modèle Zero Trust, des applications sont proposées :

- Zero Trust Network Access (ZTNA) [27] est un nouveau cadre de sécurité pour connecter les utilisateurs aux ressources de l'entreprise. Lorsqu'un utilisateur se connecte, le ZTNA basé sur le cloud vérifie l'identité et la sécurité de l'utilisateur avant de le connecter aux applications autorisées.
- Zero Trust Data Protection (ZTDP) [28] est créé par Netskope pour protéger des données. L'accès aux données est accordé sur une base contextuelle et dont le niveau de privilège est continuellement évalué et adapté de manière dynamique. ZTDP est une première ligne de défense contre l'accès non autorisé aux données et l'exfiltration.

4.2.2 Defense-in-depth

Defense-in-depth (DID) [29] est une approche de cybersécurité qui utilise plusieurs couches de sécurité pour une protection globale. Une défense en couches aide les organisations de sécurité à réduire les vulnérabilités, à contenir les menaces et à réduire les risques. Au cas où un acteur malveillant franchit une couche de défense, il peut être arrêté par la couche de défense suivante.

L'architecture de DID est conçue en plusieurs couches pour protéger les aspects physiques, techniques et administratifs du réseau.

- Les contrôles physiques comprennent des mesures de sécurité qui empêchent l'accès physique aux systèmes informatiques, telles que des gardes de sécurité ou des portes verrouillées.
- Les contrôles techniques comprennent des mesures de sécurité qui protègent les systèmes ou les ressources du réseau à l'aide de matériels ou de logiciels spécialisés, tels qu'un pare-feu ou un programme antivirus.
- Les contrôles administratifs sont des mesures de sécurité consistant en des politiques ou des procédures destinées aux employés d'une organisation, par exemple, en demandant aux utilisateurs d'étiqueter les informations sensibles comme « confidentielles » ou « secrètes ».

4.2.3 Pare-feu

Pare-feu [30] [31] est un appareil de protection du réseau qui surveille le trafic entrant et sortant sur un réseau privé afin de réaliser du contrôle d'accès sur les flux. Il fonctionne comme une barrière de protection entre le réseau interne sécurisé et le réseau externe.

Au début, le pare-feu était un simple analyseur qui autorisait ou bloquait le trafic selon un ensemble simple de règles de sécurité prédéfini. Aujourd'hui, les règles établies selon les caractéristiques des trafics permettent également la prévention des intrusions.

Un pare-feu se présente sous deux formes distinctes : le pare-feu logiciel et le pare-feu matériel.

- Le pare-feu logiciel est un programme qui peut être téléchargé et installé directement sur une machine d'extrémité. Il est facile à installer et à gérer. Il convient aux ordinateurs individuels ou aux petites entreprises.
- Le pare-feu matériel est une machine dédiée, normalement installée à l'entrée et à la sortie du réseau local. Il joue un rôle similaire au pare-feu logiciel mais de façon globale et complexe en contrôlant l'ensemble des flux en provenance ou à destination des machines d'un réseau. L'ensemble des règles à mettre en place peut s'avérer important et nécessite une maintenance permanente. Son paramétrage ou sa configuration nécessite un niveau de compétences avéré ainsi qu'une bonne connaissance des trafics devant le traverser.

Les technologies de pare-feu mettent en œuvre différents types de filtrage, ayant chacun un rôle différent :

- Le pare-feu sans état ou pare-feu de filtrage de paquets : ce type de pare-feu est plus ancien et plus simple. Il analyse les paquets et décide de les autoriser ou de les bloquer selon un ensemble de règles prédéfini : bloquer les paquets provenant d'une certaine adresse IP, etc. Ce type de pare-feu est facile à déjouer. Il ne sauvegarde pas les journaux et n'effectue pas d'analyses approfondies.
- Le pare-feu à états : il est plus complexe que le pare-feu sans état. Il analyse chaque paquet et vérifie s'il est conforme à une connexion en cours. C'est-à-dire, il vérifie si chaque paquet est bien la suite d'un paquet précédent. De plus, il continue à surveiller les paquets lorsqu'ils traversent le réseau. Il sauvegarde également des sessions et des connexions pour réagir en cas d'anomalie.
- Le pare-feu identifiant : il réalise des liens entre les adresses IP et les utilisateurs. Ainsi, les règles de filtrage sont définies par des identifications (utilisateur + IP).
- Le pare-feu applicatif ou proxy : ce type de pare-feu appartient à la dernière génération du pare-feu. Il contrôle la conformité complète du paquet à un protocole attendu.

4.2.4 Pare-feu d'application web (Website Application Firewall - WAF)

Le pare-feu d'application web (Web Application Firewall - WAF) [32] est un type de pare-feu applicatif qui aide à protéger les applications web contre les attaques. Il surveille, filtre et analyse les trafics HTTP et HTTPS entre les applications Web et l'Internet pour bloquer tout trafic malveillant. Le WAF est un proxy inversé ou reverse

proxy et permet de protéger un serveur web du trafic en provenance de l'Internet. Si le proxy vérifie la conformité du protocole en analysant le trafic en réponse, le WAF quant à lui analyse en particulier le trafic entrant et donc les requêtes des clients. Les caractéristiques d'ordre applicatif peuvent concerner l'utilisateur, la session ou l'application. Le filtrage ainsi déporté au niveau applicatif devient plus complexe et nécessite la connaissance fine des caractéristiques de développement de l'application web. C'est ainsi que le WAF peut être un point d'extraction, de transformation, d'enrichissement et d'injection d'information dans les messages à destination et/ou en provenance du serveur.

WAF peut être implémenté de trois manières :

- WAF basé sur le réseau ou basé sur le matériel : similaire à un pare-feu matériel, ce type de WAF nécessite le stockage et l'entretien de l'équipement physique. Ainsi, le coût de mise en œuvre et de maintenance est assez élevé.
- WAF basé sur l'hôte : ce type de WAF peut être déployé comme une application. Le coût est moins cher que WAF basé sur le réseau mais il demande une grande quantité de ressources de l'hôte. Ainsi, il faut avoir une machine puissante pour installer ce WAF.
- WAF basé sur le cloud : ce type de WAF est déployé et fourni par des organisations professionnelles. Il est mis à jour régulièrement par des experts et sans frais supplémentaires. De plus, les utilisateurs peuvent louer ce type de WAF pour la durée souhaitée à petit prix.

4.2.5 Virtual Private Network

Virtual Private Network (VPN) sécurisé [33] est une application destinée à renforcer la protection de la confidentialité en ligne des utilisateurs. Quand un utilisateur utilise un VPN pour sa machine, le VPN dirige tous les trafics de cette machine vers l'un de ses propres serveurs pour chiffrer toutes les données telles que la localisation, l'activité de navigation, etc. Ainsi, les attaquants ne peuvent plus identifier l'utilisateur.

Chaque jour, toutes les activités en ligne de chaque personne, telles que la localisation, le commentaire, la recherche, etc., sont collectées par les tiers : Google, Facebook, ou le fournisseur d'Internet, etc. Ils utilisent ces données à des fins non nuisibles, telles que la publicité ou les statistiques. Ces données peuvent être collectées par des attaquants qui peuvent les utiliser à des fins malveillantes. Pour augmenter la confidentialité et la sécurité, un VPN est une bonne solution. Cependant, la sécurité d'un VPN est optionnelle car un VPN avec un tunnel commutation multiprotocole par étiquette (MPLS) offre bien un partage de l'adressage IP sans sécurité. Un VPN offre des moyens importants :

- Un VPN chiffre toutes les données que nous envoyons sur l'internet. Personne ne peut voir ce que nous faisons en ligne, pas même notre fournisseur d'accès à internet.

- Un VPN protège également notre confidentialité. Les sites et les services ne peuvent plus accéder à notre IP car lorsque nous utilisons un VPN ils ne voient que l'adresse IP du VPN.
- Certains VPN bloquent les sites malveillants, les pubs et les trackers.

Les types de VPN [34] :

- VPN d'accès à distance : ce type de VPN crée des connexions entre des utilisateurs individuels et un réseau privé du VPN permettant aux utilisateurs d'utiliser les services et les ressources au sein de ce réseau privé. Ce type de VPN est idéal pour une utilisation personnelle et il est également utile pour les utilisateurs professionnels en cas de travail à distance.
- VPN de site à site ou de routeur à routeur : Il est normalement utilisé dans les entreprises qui ont des bureaux dans différentes zones géographiques. Ce type de VPN crée des connexions entre des réseaux privés où chaque réseau privé correspond à un bureau de l'entreprise.

4.2.6 Gestion des identités et des accès (Identity and Access Management - IAM)

La gestion des identités et des accès (Identity and Access Management - IAM) [35] est un système qui gère les identités numériques des utilisateurs et leurs privilèges d'accès aux ressources de l'organisation. En utilisant des solutions IAM, les administrateurs peuvent non seulement identifier les utilisateurs et modifier leurs privilèges, mais ils peuvent également surveiller l'activité des utilisateurs et fournir des rapports. De plus, les utilisateurs peuvent appliquer les politiques de conformité réglementaire et de l'entreprise pour protéger la sécurité et la confidentialité des données.

Les services en ligne sont de plus en plus diversifiés. Pour y accéder, vous devez disposer d'informations d'authentification pour vérifier l'identité de l'utilisateur. Les solution IAM font correspondre ces informations d'authentification aux utilisateurs. Les solutions IAM sont proposées sous forme de services en nuage ou déployées sur le site, ou hybride à la fois sur le site et dans le nuage.

Les facteurs d'authentification les plus couramment utilisés pour IAM sont les suivants :

- Mot de passe : c'est une chaîne de caractères générés par l'utilisateur. Ainsi, l'utilisateur connaît bien le mot de passe.
- Téléphone intelligent : c'est une chose que l'utilisateur possède. L'utilisateur peut choisir les différentes méthodes d'authentification par téléphone telles que le SMS, l'appel, l'application d'authentification.
- Une propriété physique comme une empreinte de doigt : c'est une chose qui appartient à l'utilisateur.

Normalement, les entreprises utilisent l'authentification à plusieurs facteurs pour renforcer la fiabilité et la sécurité.

Les cas d'utilisation IAM sont par exemple :

- Les solutions IAM permettent aux organisations d'autoriser ou de bloquer l'accès aux données et applications protégées.
- Les organisations peuvent utiliser IAM pour limiter le nombre d'utilisateurs pouvant accéder à une plateforme. De plus, IAM permet de définir le rôle de chaque utilisateur : développer, préparer, ou tester des produits et des services dans la plateforme.
- Les solutions IAM sont utilisées pour définir des règles strictes d'utilisation des données sensibles : qui peut consulter, créer, modifier, supprimer, ou transmettre quelles données.
- Les systèmes IAM fournissent des rapports de sécurité. Ces rapports permettent non seulement à une organisation de démontrer sa conformité aux réglementations en matière de sécurité et de confidentialité des données mais également de comprendre et d'améliorer son système de sécurité.

Les outils ci-dessus de ce bloque « protection » permettent de prévenir des attaques et de protéger le système mais ce n'est pas suffisant parce que nous ne savons pas s'ils peuvent détecter le type d'attaque contenant du trafic autorisé mais illégitime. Nous avons besoin de mesures pour détecter et empêcher les attaques.

4.3 Défense

Le pilotage du système d'information (SI) permet de mieux connaître les applications métiers, les données échangées, collectées, stockées, ainsi que les utilisateurs qui les produisent ou les consomment. L'audit du SI comme l'analyse de risque permettent la mise en place de mesures de protection. Mais nous ne pouvons pas garantir que notre système pourrait prévenir toutes les attaques, en particulier le type d'attaque complexe contenant du trafic autorisé mais illégitime. Ce type d'attaque est souvent créé et géré par des attaquants hautement qualifiés qui le contrôlent pour qu'il reste longtemps indétectable dans le réseau. C'est pourquoi, notre système doit disposer de mesures capables de détecter les attaques en essayant d'analyser des journaux afin de les détecter le plus tôt possible. Nous présentons quelques outils pour la défense ci-dessous.

Exemples d'outils pour la défense :

4.3.1 Système de détection d'intrusion (Intrusion Detection System - IDS)

Un système de détection d'intrusion (Intrusion Detection System - IDS) [36] [37] est une application logicielle qui surveille et analyse le réseau pour rechercher des activités malveillantes ou des signes de comportement anormal. Les administrateurs peuvent définir des règles pour détecter des attaques. Un IDS compare l'activité du

réseau avec ces règles pour identifier toute activité pouvant indiquer une attaque ou une intrusion. Lorsqu'une activité malveillante est détectée, un IDS envoie une alerte à l'administrateur afin qu'il puisse analyser et prendre des mesures pour prévenir les dommages ainsi que d'autres intrusions.

Il existe deux types d'IDS :

- Un IDS basé sur l'hôte (Host-based Intrusion Detection System - HIDS) est une application qui surveille un ordinateur ou un serveur pour rechercher des activités suspectes. Les activités surveillées peuvent inclure des intrusions créées par des acteurs externes mais également par une mauvaise utilisation de ressources ou de données en interne. Un HIDS enregistre des activités suspectes et les signale aux administrateurs gérant les appareils. Un HIDS surveille les fichiers journaux générés par les applications et crée un historique des activités et des fonctions, nous permettant ainsi d'identifier rapidement les anomalies et les signes d'une intrusion qui auraient pu se produire.
- Un IDS basé sur le réseau (Network Intrusion Detection System - NIDS) est souvent un appareil matériel autonome qui a des capacités de détection de réseau. Il se compose de capteurs matériels situés en divers points du réseau et de logiciels installés sur divers ordinateurs connectés au réseau. Le NIDS analyse les paquets de données entrants et sortants et offre une détection en temps réel.

Les méthodes de détection de IDS :

- Méthode basée sur la signature : Un IDS enregistre les caractéristiques d'identification spéciales des attaques connues dans sa signature. Mais il est difficile de détecter de nouvelles attaques qui ne soient pas dans sa signature.
- Méthode basée sur les anomalies : les attaques s'améliorent et changent en permanence. Un IDS basé sur les anomalies est proposé en utilisant des algorithmes d'apprentissage automatique. Il crée un modèle d'activité de confiance et tout ce qui arrive est comparé à ce modèle et déclaré suspect s'il n'est pas trouvé dans le modèle. Ce type d'IDS peut détecter des attaques non connues.

4.3.2 Analyse du comportement des utilisateurs et des entités (User and Entity Behavior Analytics - UEBA)

L'analyse du comportement des utilisateurs et des entités (User Entity Behavior Analytics - UEBA) [38] [39] est un processus de collecte d'informations sur les événements réseaux que les utilisateurs génèrent chaque jour. Une solution UEBA utilise ces informations pour modéliser les comportements typiques et atypiques des humains et des machines au sein d'un réseau en surveillant le comportement des utilisateurs et des entités d'une organisation. Basée sur des modèles prédéfinis, une solution UEBA traite les informations observées et peut décider si une activité ou un

comportement particulier peut entraîner une cyberattaque que des outils comme les antivirus traditionnels ne peuvent pas détecter. Une solution UEBA est capable de savoir ce qu'est une menace ou une attaque et ce qui est une utilisation normale, parce que quand un pirate infiltre le réseau en utilisant un mot de passe volé par exemple, ce dernier ne pourra pas imiter le comportement "normal", ce qui sera alors détecté.

Les méthodes de sécurité traditionnelles telles que les pare-feux, les IDS, etc ne peuvent protéger complètement une organisation contre les intrusions. Ainsi, les solutions UEBA sont de plus en plus développées afin de détecter même les plus petits comportements inhabituels et de les prévenir au plus tôt. Une solution UEBA fonctionne plus efficacement lorsqu'elle est installée non seulement sur chaque appareil de l'organisation, mais également sur les appareils personnels des employés afin d'en apprendre le plus possible sur leurs comportements.

4.3.3 HoneyPot

Honeypot [40] est un mécanisme de sécurité qui crée un piège virtuel pour attirer les attaquants. C'est une machine qui si elle participe à un échange, implique que le trafic est forcément anormal et donc l'objet d'une attaque en cours. Le « pot de miel » permet aux attaquants d'exploiter des vulnérabilités connues. Les équipes de sécurité peuvent alors étudier les comportements des attaquants pour améliorer leurs politiques de sécurité. Les « pots de miel » peuvent être installés sur tout type de ressource informatique, des logiciels aux serveurs de fichiers et aux routeurs pour attirer les cybercriminels.

Un honeypot peut être décomposé en fonction du type d'activité à détecter :

- Piège à e-mail ou piège à spam : une adresse e-mail fictive est ajoutée dans un champ caché qui ne peut être détecté que par un collecteur d'adresses automatisé ou un robot d'exploration de site. Les utilisateurs légitimes ne peuvent pas voir cette adresse. L'organisation peut bloquer tous les expéditeurs qui envoient des emails à cette adresse.
- Base de données leurre : c'est un ensemble de données fictives intentionnellement vulnérables qui aide les organisations à surveiller les vulnérabilités logicielles, les trous de sécurité de l'architecture ou même les acteurs internes néfastes. Grâce à cette base de données leurre, les organisations recueilleront des informations sur les techniques d'injection, le piratage d'informations d'identification ou l'abus de privilèges utilisés par un attaquant. Ces informations sont utilisées dans les systèmes de défense et les politiques de sécurité.
- Honeypot malveillant : il imite une application logicielle ou une interface de programmation d'application (Application Programming Interface - API). Ensuite, dans un environnement contrôlé, les organisations essaient d'éliminer les attaques de logiciels malveillants. Ce faisant, les organisations peuvent

analyser les techniques d'attaque et améliorer les solutions anti-malware pour répondre à ces vulnérabilités et menaces.

- Honeypot araignée : les organisations conçoivent un honeypot araignée avec des liens accessibles uniquement aux robots d'exploration automatisés pour piéger les robots d'exploration Web. Les organisations peuvent comprendre comment bloquer les robots d'exploration des réseaux publicitaires par exemple.

4.3.4 Système de gestion des informations et des événements de sécurité (Security Information and Event Management - SIEM)

Un SIEM [41] [42] est un logiciel qui permet à une organisation de prendre des mesures préventives contre les cyberattaques grâce à la collecte et à l'analyse de données de sécurité provenant de diverses sources au sein de l'infrastructure réseau.

De nombreuses entreprises utilisent un SIEM pour gérer les journaux d'événements et protéger leur réseau contre les attaques. D'une part, le SIEM enregistre des journaux et les analyse à des fins de surveillance en temps réel de la conformité des événements informatiques avec des règles établies. D'autre part, l'outil assure la collecte des données de sécurité et les analyses de conformité requises. En utilisant des bases de connaissance sur des attaques complexes comme MITRE ATT&CK [5], les experts en cybersécurité créent des règles SIEM pour détecter et prévenir les cyber-attaques. Un SIEM collecte toutes les données du réseau pour les analyser et établir des corrélations, telles que les données des logiciels antivirus, des pare-feux, des serveurs, des applications, etc. À partir des règles et des journaux, le SIEM va donner des alertes s'il y a des activités anormales.

Un SIEM collecte les données de nombreuses sources telles que :

- Les périphériques réseau sont des appareils utilisés pour communiquer et échanger entre les matériels d'un réseau informatique : routeurs, commutateurs, concentrateurs, points d'accès, etc.
- Les serveurs informatiques offrent des services accessibles via le réseau. Ils répondent aux demandes des machines clientes : serveur web, serveur de messagerie, etc.
- Les dispositifs de sécurité offrent des mesures pour protéger les données, les applications, les appareils et les systèmes contre les attaques : pare-feu, IDS, logiciels antivirus, serveur proxy etc.
- Applications : tous les logiciels installés sur les dispositifs ci-dessus.

4.3.5 Équipe d'intervention en cas d'urgence informatique (Computer Emergency Response Team - CERT)

L'équipe d'intervention en cas d'incident de sécurité informatique (Computer Security Incident Response Team - CSIRT) ou l'équipe d'intervention en cas d'urgence informatique (Computer Emergency Response Team - CERT) [43] est une équipe de

sécurité opérationnelle, composée d'experts de différents domaines (malwares, tests d'intrusion, veille, lutte contre la cybercriminalité, etc.) pour prévenir et réagir en cas d'incident informatique.

Après la création et la propagation du premier "ver" informatique, un groupe composé d'experts a été formée pour trouver un moyen d'éliminer ce logiciel malveillant. Ensuite, la DARPA (Defense Advanced Research Projects Agency) a pris la décision de mettre en œuvre une structure permanente, le CERT. Lorsqu'un incident de sécurité se produit, le CERT joue un rôle de coordination avec les administrateurs informatiques des réseaux touchés pour répondre à cet incident.

Ces équipes ont 5 grandes missions :

- Premièrement, elles centralisent les demandes d'assistance à la suite d'incidents de sécurité sur les réseaux et les systèmes d'information.
- Deuxièmement, elles échangent des informations d'analyse technique avec d'autres CSIRT pour contribuer à des études techniques spécifiques.
- Troisièmement, elles établissent et maintiennent une base de données des vulnérabilités.
- Quatrièmement, elles diffusent les informations sur les précautions à prendre pour minimiser les risques d'incident.
- Enfin, elles échangent et restent en contact régulier avec les autres entités comme les centres de compétence réseaux, les opérateurs et fournisseurs d'accès à Internet, et les CSIRT nationaux et internationaux.

Aujourd'hui, il y a des centaines de CSIRT/CERT dans les diverses organisations telles que des États, des universités, ou des entreprises. Nous pouvons les appeler des CSIRT/CERT internes. Nous pouvons trouver également des CSIRT/CERT commerciaux. Ce sont des groupes d'experts appartenant à des prestataires de services. Elles proposent des services de veille, de réponse à incidents, etc. à leurs clients. De plus, il existe des plateformes d'échange qui partagent gratuitement des informations sur les menaces ou les indicateurs de cybersécurité dans le monde entier tels que MISP[1], STIX[2], TAXII[3]. Elles comptent une communauté active de développeurs et d'analystes.

4.3.6 Centre d'opérations et de sécurité (Security Operation Center - SOC)

Centre d'opérations et de sécurité (Security Operations Center - SOC) [44] est une plateforme où les systèmes d'information de l'entreprise (sites Web, applications,

[1] https://misp-project.org/

[2] https://oasis-open.github.io/cti-documentation/stix/intro

[3] https://oasis-open.github.io/cti-documentation/taxii/intro

bases de données, centres de données, serveurs, réseaux et postes de travail et autres terminaux) sont surveillés, évalués et défendus.

Un SOC inclut des personnes, des processus et des technologies pour surveiller et améliorer en permanence la posture de sécurité d'une organisation tout en prévenant, détectant, analysant et répondant aux incidents de cybersécurité. D'abord, un SOC surveille le réseau 24 heures sur 24 afin de détecter les activités malveillantes et les prévenir avant qu'elles ne causent des dommages. Lorsque l'analyste SOC voit quelque chose de suspect, il recueille autant d'informations que possible pour une enquête plus approfondie. L'analyste SOC essaie de considérer le réseau et les opérations de l'organisation du point de vue d'un attaquant pour rechercher des indicateurs de compromission et des zones d'exposition avant qu'ils ne soient exploités. Dès qu'un incident est confirmé, le SOC effectue des actions telles que l'isolement des terminaux, l'arrêt des processus nuisibles, l'empêchement de s'exécuter, la suppression de fichiers, etc.

En fonction de la localisation, il existe plusieurs types de SOC :

- SOC interne, installé dans les locaux de l'organisation. Il est géré par des équipes informatiques internes.
- SOC externe, loué chez un prestataire de sécurité informatique. Il est contrôlé par des équipes informatiques de ce prestataire.

Un SOC va pouvoir permettre à une organisation d'administrer son réseau informatique à distance et ainsi de réduire les risques.

4.3.7 MITRE ATT&CK

Les attaques complexes comme les APTs sont très compliquées et difficiles à détecter. Pour relever un tel défi, MITRE a proposé ATT&CK une base de connaissances sur des comportements de cyber-adversaires et des différentes phases du cycle de vie des attaques. Avec MITRE ATT&CK [5], nous pouvons cartographier et essayer de comprendre le détail des actions d'un attaquant. Avec cet enchaînement d'actions, les traces qu'elles laissent, et les erreurs commises vont constituer un profil, une empreinte de techniques qui permettront la détection précoce d'attaques similaires dans le futur.

MITRE ATT&CK fournit une matrice permettant de cartographier les tactiques et techniques d'une attaque. Une tactique représente un des objectifs que l'attaquant cherche à atteindre dans le système d'information compromis. Une technique est un comportement spécifique visant à atteindre un objectif. Elle constitue une étape dans une série d'activités d'une attaque. Pour réussir une tactique, les attaquants utilisent une ou plusieurs techniques. Pour les entreprises, aujourd'hui MITRE ATT&CK distingue 14 tactiques (Reconnaissance, Développement des ressources, Accès Initial, Exécution, Persistance, Escalade de privilèges, Évasion défensive, Accès aux

identifiants, Exploration, Mouvements latéraux, Collection de données, Commande & Contrôle, Exfiltration, et Impact) et 222 techniques[4].
La matrice fournit aux équipes de cybersécurité une présentation concise et complète du cycle de vie d'une attaque, les tactiques et les techniques. Grâce à ces schémas, les équipes de cybersécurité ont les moyens de comprendre les attaques les plus complexes. Éventuellement, elle leur permet de créer des règles de surveillance ou de mettre en place des contrôles proactifs, dans le but de détecter des menaces présentes sur le système. Par exemple, APT32 [6] dont on a parlé et qui a ciblé plusieurs industries du secteur privé dans des pays d'Asie du Sud-Est comme le Vietnam, le Laos et le Cambodge, a utilisé 76 techniques de menace différentes pour attaquer ses cibles. La matrice[5] donne une vue globale sur les tactiques et les techniques. Avec cette matrice, nous pouvons voir exactement quelles techniques sont utilisées pour atteindre quels objectifs dans quelles tactiques. Par exemple, pour réussir la tactique « Reconnaissance », APT32 utilise deux techniques : une technique collecte les adresses e-mail d'activistes et de blogueurs afin d'envoyer et installer un Spyware, une autre technique utilise les liens malicieux pour collecter les identités. Ainsi, avec la matrice fournie par MITRE ATT&CK, nous pouvons comprendre clairement et analyser les attaques complexes.

4.4 Résilience

Bien que nous utilisions de nombreuses mesures pour protéger le système et détecter les attaques, il faut se préparer contre des attaques de type 0-day. En cas d'incident de sécurité sur le système, cela entraîne des impacts importants pour l'organisation, tels que la perte de données importantes, l'arrêt d'une partie du système ou même l'arrêt complet du système. De plus, la survenance d'une cyberattaque peut nuire à la réputation d'une entreprise. Cependant, si l'organisation peut réagir et prendre rapidement des mesures de gestion de crise comme l'envoi d'alerte, cela lui permettrait de gagner en crédibilité auprès des collaborateurs et des clients. Ainsi, savoir anticiper la fulgurance des crises et préparer son organisation est nécessaire pour limiter l'impact de l'attaque et assurer la reprise d'activité, en un minimum de temps et avec une perte minimale des données.

Pour gérer efficacement des crises cyber, il y a plusieurs démarches à suivre [45] :

- Définir la procédure et des ressources lorsque la cyberattaque est identifiée : des membres de la cellule de crise, un canal unique de communication pendant la crise pour éviter les pertes d'informations, un répertoire partagé et sécurisé qui contient des documents triés selon leurs objectifs et accessibles en toutes circonstances, etc.

[4] https://attack.mitre.org/

[5] https://attack.mitre.org/groups/G0050/

- Analyser les différents types de crises de cybersécurité pour planifier les différents scénarios adaptés aux circonstances. À partir de là, nous pouvons simuler les crises pour former les équipes à la gestion de crise. Ces simulations permettent d'améliorer les procédures, les techniques, mais aussi d'identifier les vulnérabilités du système pour éviter les crises majeures.
- Diffuser les informations du risque cyber en interne, renforcer la surveillance pour détecter les signes avant-coureurs des cyberattaques.
- Si toutes les tentatives pour empêcher la crise de se produire ont échoué, alors nous devons appliquer la procédure que nous avons définie. Par exemple, tous les membres de la cellule de crise sont prévenus en premier. Ils utilisent les ressources définies pour communiquer et effectuer des recherches afin de reconnaître la survenue de la crise et identifier le type de crise. Ils mobilisent toutes les techniques et tous les moyens pour contrer les attaques. En même temps, ils déploient les différentes équipes pour limiter l'impact de l'attaque et assurer la reprise d'activité : la relation client qui répond aux questions des utilisateurs, la communication interne qui communique avec les collaborateurs, le plan de communication externe, l'équipe d'audit qui analyse l'état du réseau informatique et l'ampleur des dégâts, les développeurs, qui assurent la continuité des activités en déployant un SI alternatif.
- À la fin de la crise, recueillir les retours d'expérience pour améliorer la procédure de gestion de crise.

5 Architecture détaillée

L'Internet se développant rapidement, il y a de plus en plus de menaces. Ainsi, pour protéger efficacement le réseau d'une organisation, il faut avoir une architecture technique pour l'interconnexion de son système d'information (SI) avec l'Internet. L'Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI) a proposé un guide « Recommandations relatives à l'interconnexion d'un système d'information à Internet v3 » [46] pour que les organisations puissent construire et sécuriser leur architecture technique d'interconnexion. Pour protéger le réseau de l'organisation, il faut avoir une passerelle d'interconnexion sécurisée entre le SI interne de l'organisation et l'Internet pour éviter tout accès direct. Le guide de l'ANSSI met en œuvre le concept de zone démilitarisée (Demilitarized Zone - DMZ) qui est un sous réseau entre le SI interne et l'Internet. Les points de filtrage et d'analyse du trafic y sont installés. Une passerelle d'interconnexion sécurisée contient une ou plusieurs DMZ. La Figure 10 présente un exemple d'architecture multi-services détaillée d'un SI interne, d'une passerelle Internet sécurisée en distinguant cinq types de zones (regroupements de ressources logicielles ou matérielles) et d'un SI administration.

Le SI interne contient :

- Zone de services sans interconnexion à l'Internet.

- Zone de services avec interconnexion à l'Internet.
- Zone de postes sans accès à l'Internet.
- Zone de services d'infrastructures non exposés sur l'Internet.
- Zone de postes avec accès à l'Internet.

La passerelle Internet sécurisée contient :

- Zone d'accès interne pour le filtrage entre le SI interne et la passerelle Internet sécurisée. Nous pourrons installer les outils de protection tels que le pare-feu interne, un VPN pour filtrer les flux entre le SI interne et la passerelle Internet sécurisée.
- Zone de services internes pour les ressources dédiées au fonctionnement de la passerelle Internet sécurisée telles que le serveur de résolution de noms DNS publics.
- Zone de services exposés pour l'hébergement éventuel de serveurs métiers tels que le serveur Web ou le serveur de transfert de fichiers.
- Zone de services relais pour la rupture protocolaire et l'analyse des flux. Nous pourrons installer les outils de défense tel que IDS pour analyser les flux et bloquer tout flux malveillant.
- Zone d'accès externe pour le filtrage entre la passerelle Internet sécurisée et Internet. Nous pourrons installer les outils de protection comme dans la zone d'accès interne mais pour filtrer les flux entre la passerelle Internet sécurisée et Internet.

Le SI administratif permet à l'administrateur d'observer et de gérer l'interconnexion de réseaux, y compris le SI Internet et la passerelle d'interconnexion sécurisée. Nous pourrons installer les outils de supervision et de gestion tels que SIEM ou SOC. Après avoir recherché et analysé les données recueillies par des outils, les administrateurs peuvent envoyer des ordres pour protéger le réseau.

Dans l'architecture de la Figure 10 , il y a plusieurs types de flux :

- Tout flux (en vert) venant du SI interne et à destination d'Internet est initié par le client du SI vers la passerelle Internet sécurisée puis de la passerelle Internet sécurisée vers Internet. De même, tout flux (vert) venant d'Internet et à destination du SI est initié par le client sur Internet vers la passerelle Internet sécurisée puis de la passerelle Internet sécurisée verts le SI. Il n'y a pas de flux initié depuis la passerelle Internet sécurisée.
- Les flux de synchronisation d'annuaire (en orange) sont à l'initiative de l'annuaire du SI interne vers l'annuaire dédié de la passerelle Internet sécurisée.
- La passerelle Internet sécurisée et le SI interne sont gérés par un système d'information d'administration sécurisé via les flux d'administration (rouge).

- Il est recommandé de ne pas initialiser de flux depuis la passerelle Internet sécurisée vers le SI interne. Mais cette recommandation n'est pas applicable à l'ensemble des flux, par exemple le cas d'un relais de réception de messagerie. Ainsi, une alternative, d'un niveau de sécurité moindre, des flux sont tolérés (en bleu).
- Les flux annexes (en gris) sont des flux de communication entre sous-groupes du SI interne ou de la passerelle Internet sécurisée.
- Tous les flux collectés et envoyés aux outils de supervision et de gestion (en jaune) sont également représentés.

6 Conclusion

Notre objectif est de détecter les trafics chiffrés autorisés mais illégitimes. Ces derniers sont souvent envoyés par des canaux C&C. Les attaques complexes comme les APTs utilisent souvent ce type de trafic pour envoyer les ordres ou exfiltrer des données sensibles. Cela peut sérieusement affecter l'organisation et la vie personnelle des utilisateurs. Malgré tous les outils actuels, il est encore très difficile de protéger les systèmes d'information contre ce type d'attaque. La détection est difficile parce que des attaquants hautement qualifiés contrôlent ce type d'attaque qui reste en dessous du seuil de détection des outils. Dans ce cas, les techniques d'apprentissage automatique peuvent aider à identifier les comportements anormaux par analyse comportementale dans les réseaux. Dans le chapitre 2, nous présentons quelques algorithmes d'apprentissage automatique non supervisés qui peuvent détecter des anomalies.

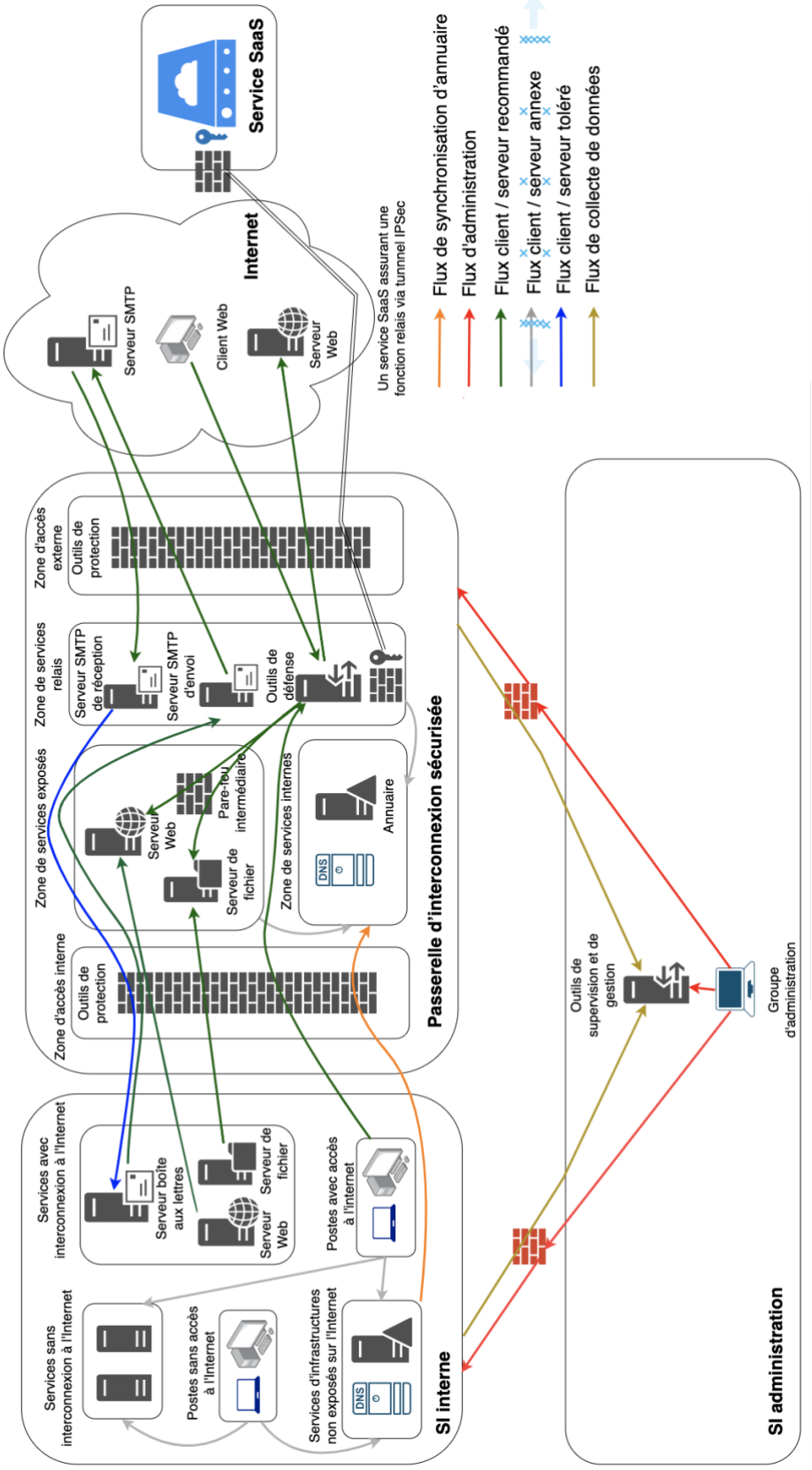


Figure 10. Architecture de la passerelle Internet sécurisée [46]

Chapitre 2. Techniques d'apprentissage automatique non supervisées

1 Introduction

Les attaques complexes comme les APTs sont normalement organisées par des attaquants hautement qualifiés qui contrôlent le taux d'attaque pour qu'il reste en dessous du seuil de détection de pare-feu et autres outils de détection d'attaque comme les SIEMs, HIDSs, etc. C'est pourquoi ces attaques sont très rares et les outils de détection d'attaques traditionnels ne peuvent pas les détecter. De plus, les APTs aiment créer des canaux de communication C&C pour cacher leurs demandes et faire ressembler un trafic d'attaque à un trafic normal : même porte, même structure de message. Donc pour détecter ces APTs, une analyse plus approfondie avec un grand nombre de critères est nécessaire. Cependant, l'écriture des règles de détection avec autant de critères dans les outils de détection traditionnels est une tâche difficile. Dans ce cas, les techniques d'apprentissage automatique peuvent aider, en particulier les algorithmes de détection des valeurs aberrantes (des anomalies). Nous partons d'algorithmes simples et bien connus comme K-means, Gaussian Mixture Model (GMM), Isolation Forest à des algorithmes plus complexes comme One-class SVM, Local Outlier Factor (LOF), Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Auto-Encoder, et Self Organizing Map (SOM) pour évaluer leurs performances dans la détection des anomalies.

2 K-means

K-means [47] est un algorithme basé sur la distance. Il essaie de regrouper les points les plus proches en plusieurs groupes dont le nombre K est prédéterminé (Figure 11). L'algorithme K-means contient 3 étapes :

- Étape 1. Initialisation : déterminer la valeur K qui représente le nombre de clusters. Choisir aléatoirement K points qui sont les K-centres de K clusters de données (nommé centroïd).
- Étape 2. Affectation de cluster : calculer la distance (distance euclidienne) entre chaque point et le centroïd. Attribuer chaque point au cluster le plus proche.
- Étape 3. Déplacer le centroïd : calculer la moyenne de chaque cluster en tant que nouveau centroïd. Répéter étape 2 et étape 3 jusqu'à ce que les centroïds s'arrêtent de changer.

K-means peut s'exécuter rapidement et efficacement sur n'importe quel ensemble de données. Cependant, K-means n'utilise que des formes circulaires pour diviser des données, il manque donc de flexibilité pour les groupes de données avec d'autres formes telles qu'un losange ou un demi-cercle.

Avantages	Inconvénients
- Facile à comprendre et implémenter. - Temps de calcul court.	- Le nombre de clusters doit être défini à l'avance. - Utilise uniquement des cercles pour regrouper les données. - Donne des résultats médiocres pour les données qui ne sont pas linéairement séparables. - N'est pas adapté aux données non numériques.

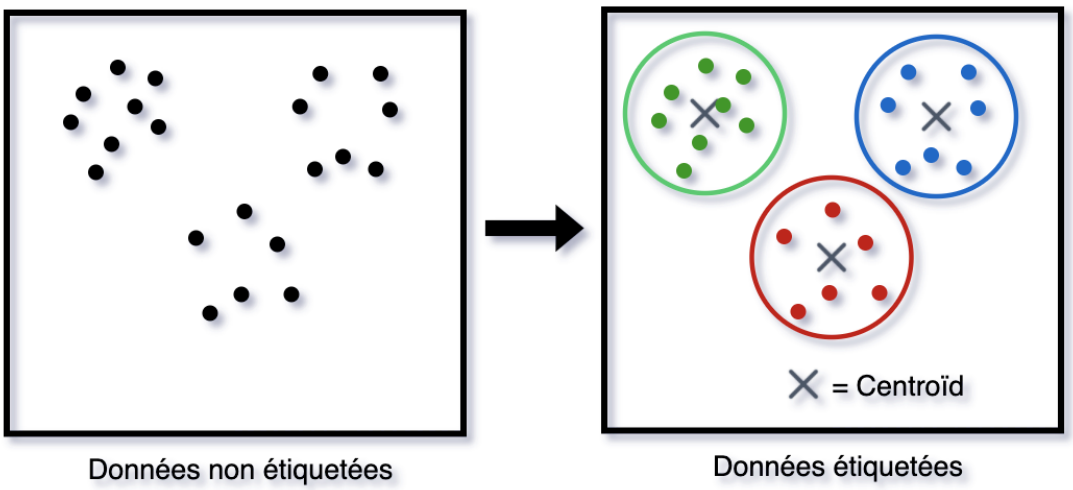


Figure 11. K-means

3 Gaussian Mixture Models (GMM)

Gaussian mixture models (GMM) [48] est un algorithme basé sur la distribution. GMM suppose qu'il y a M densités composantes gaussiennes et il essaie de regrouper les points de données appartenant au même composant pour former M groupes.

GMM est une fonction de densité de probabilité paramétrique représentée comme une somme pondérée de M densités de composantes gaussiennes par l'équation suivante :

$$p(x|\lambda) = \sum_{i=1}^{M} w_i \, g(x|\mu_i, \Sigma_i)$$

Où x sont des caractéristiques à valeurs continues de D-dimensionnelle, w_i sont les poids de mélange qui satisfont à la contrainte $\sum_{i=1}^{M} w_i = 1$, et $g(x|\mu_i, \Sigma_i)$ sont les densités composantes gaussiennes.

Pour un vecteur x de D-dimensions, la distribution gaussienne multivariée prend la forme suivante :

$$(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} e^{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)}$$

Avec μ_i est le vecteur moyen de D-dimension, et $\Sigma_i$est la matrice de covariance *DxD.*

Tous les paramètres de GMM, y compris les vecteurs moyens, les matrices de covariance et les poids de mélange des M densités composantes sont représentés par cette notation :

$$\lambda = \{\mu_i, \Sigma_i, w_i\} \quad i = 1, \dots, M$$

Il existe plusieurs approches pour estimer les paramètres de GMM et la plus populaire est l'algorithme Expectation-Maximization (EM) [49], qui optimise de manière itérative le modèle à l'aide d'estimations de vraisemblance maximale. EM comprend 4 étapes :

- *Étape 1. Initialisation* : La stratégie naïve attribue des valeurs aléatoires aux moyennes, aux variances et aux poids de mélange. Une stratégie plus élaborée consiste à utiliser une certaine forme d'estimation de quantification vectorielle binaire ou les résultats obtenus par une exécution précédente de K-means pour définir ces valeurs.
- *Étape 2. E-étape* : pour chaque point, calculer la probabilité qu'il appartienne à chaque groupe/composante. Cette probabilité sera plus élevée lorsque le point est attribué au bon cluster et plus faible dans le cas contraire.
- *Étape 3. M-étape* : mettre à jour les paramètres gaussiens (λ) pour chaque composant pour s'adapter aux points qui leur sont attribués.
- *Étape 4. Évaluer la convergence* : vérifier la convergence des paramètres. Si un certain seuil de convergence est atteint, arrêter l'algorithme. Sinon, retourner à E-étape (étape 2). Le nouveau modèle devient alors le modèle initial pour la prochaine itération.

Alors que les clusters créés par K-means ont une forme de cercle, GMM peut gérer des clusters elliptiques. K- means est une classification dure tandis que GMM est une classification douce. GMM fournit les probabilités qu'un point de données appartienne à chacun des clusters possibles.

Avantages	Inconvénients
- Fournit des estimations de la probabilité que chaque point de données appartienne à chaque cluster ce qui permet de trouver un groupe pour les points de données ambigus qui se situent à la frontière de deux clusters. - Crée non seulement des clusters circulaires mais aussi des clusters elliptiques.	- Le nombre de clusters doit être défini à l'avance. - Le temps de calcul est plus long que K- means. - Ne fonctionne pas bien dans les cas où les clusters sont de forme très irrégulière. - Peut être difficile à interpréter.

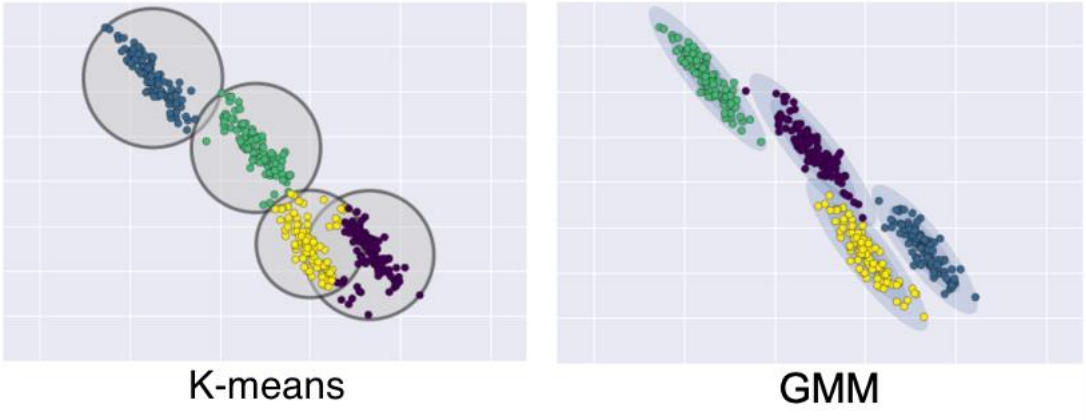


Figure 12. Classification par K-means et GMM [50]

4 Isolation Forest

Isolation Forest [51] est un algorithme basé sur des arbres, construit sur la théorie des arbres de décision et des forêts aléatoires. Après que l'algorithme soit exécuté sur toutes les données, il filtre les points de données qui ont pris le moins d'étapes (moins de temps) pour trouver d'autres points. Si le point est une valeur aberrante, il sera seul et pourra être trouvé facilement. D'un autre côté, si le point n'est pas une valeur aberrante, il y aura beaucoup de points qui seront autour, et il sera difficile à isoler. Isolation Forest a un paramètre qui doit être défini : *n_estimators*, qui est le nombre d'estimateurs de base dans l'ensemble.

Isolation Forest contient 4 étapes :

- Étape 1 : Sélectionner le point à observer.
- Étape 2 : Pour chaque caractéristique, définir la plage à isoler entre le minimum et le maximum.
- Étape 3 : Choisir aléatoirement une caractéristique et sélectionner au hasard une valeur pour cette caractéristique dans sa plage :

- Si la valeur de la caractéristique du point est supérieure à la valeur sélectionnée, cette valeur devient le nouveau minimum de la plage de cette caractéristique.
- Si la valeur de la caractéristique du point est inférieure à la valeur sélectionnée, cette valeur devient le nouveau maximum de la plage de cette caractéristique.

- Étape 4 : Répéter les étapes 3 et 4 jusqu'à ce que le point soit isolé. Le nombre de fois que ces étapes sont exécutées est le nombre d'isolement. Une valeur faible pour le nombre d'isolement reflète une anomalie du point.

Avantages	Inconvénients
- Fonctionne bien dans les petits ensembles. - Moins d'effort de calcul et faible besoin en mémoire. - Peut être appliqué avec des variables continues et catégorielles	- Le nombre d'estimateurs de base dans l'ensemble doit être défini. - Il faut avoir à l'avance une idée du pourcentage des données anormales pour obtenir une meilleure prédiction. - Ne fonctionne pas bien lorsque des relations non linéaires existent dans les données.

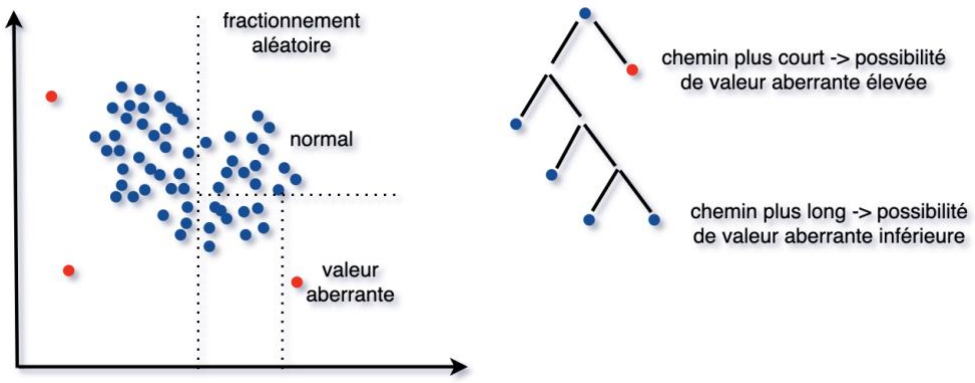


Figure 13. Isolation Forest

5 One-class Support Vector Machine (One-class SVM)

One-class SVM est utilisé lorsqu'une classe est connue et le problème est de détecter tout ce qui se trouve en dehors de cette classe. Schölkopf et al. [52] ont modifié l'algorithme Support Vector Machine (SVM) pour séparer tous les points de données de l'origine et maximiser la distance entre cet hyperplan et l'origine.

SVM représente les données dans un espace de dimension supérieure *F* en projetant les données à travers une fonction non linéaire ϕ. Dans cet espace de dimension supérieure, les points de données peuvent être séparés par une ligne droite, ce qui n'est pas possible dans leur espace d'origine.

One-class SVM a un paramètre, appelé *nu*, qui doit être défini. *nu* définit une limite supérieure sur les valeurs aberrantes et la limite inférieure sur le nombre d'exemples d'apprentissage utilisés comme vecteurs de support.

Considérons un ensemble de données $\mathcal{X} = \{x_1, x_2, \dots, x_\ell\}$, où $\ell \in N$ est le nombre d'observations. Pour simplifier, $\mathcal{X}$ est un sous-ensemble compact de $\mathbb{R}^N$.

One-class SVM contient 4 étapes:

- Étape 1 : ϕ est une cartographie des caractéristiques $X \to F$, c'est-à-dire ϕ ramène nos vecteurs dans *X* à un espace de caractéristiques *F*. Deux vecteurs d'entrées x et y dans X sont convertis en $\phi(x)$ et $\phi(y)$ dans l'espace F . Fonction noyau est une fonction *k* qui correspond à :

$$k(x, y) = \phi(x)^T \phi(y)$$

- Étape 2 : Pour séparer l'ensemble de données de l'origine, il faut résoudre le programme quadratique suivant :

$$\min_{w \in F, \xi \in \mathbb{R}^\ell, \rho \in \mathbb{R}} \frac{1}{2} \| w \|^2 + \frac{1}{nu * \ell} \sum_{i=1}^{\ell} \xi_i - \rho$$

sujet à:

$\left(w \cdot \phi(x_i)\right) \geq \rho - \xi_i$ for all $i = 1, \dots, \ell$

$\xi_i \geq 0$ for all $i = 1, \dots, \ell$

Avec :

  - nu qui définit une limite supérieure pour la fraction des valeurs aberrantes (des données qui n'appartiennent pas à une classe).
  - nu qui est une limite inférieure du nombre de données utilisées comme vecteur de support.

- Étape 3 : Utiliser les multiplicateurs de Lagrange pour résoudre ce problème de minimisation (avec programmation quadratique). La règle de la fonction de décision (classification) pour un point de données x devient :

$$f(x) = sign(\left(w \cdot \phi(x)\right) - \rho)$$

- Étape 4 : Utiliser la fonction du noyau à l'étape 1, la fonction devient :

$$f(x) = sign(\sum_{i=1}^{\ell} \alpha_i k(x_i, x) - \rho)$$

Le résultat de cette fonction f égale +1 pour une "petite" région qui contient la plupart des points de données (point normal), et -1 ailleurs (point d'anomalie). Pour un nouveau point x, la valeur $f(x)$ est déterminée en évaluant de quel côté de l'hyperplan il tombe, dans l'espace des caractéristiques.

Avantages	Inconvénients
- Fonctionne bien lorsqu'il existe une marge de séparation claire entre les classes. - Est efficace dans les espaces de grande dimension.	- La valeur de *nu* doit être définie - N'est pas adapté aux grands ensembles de données. - Ne fonctionne pas très bien lorsque l'ensemble de données a beaucoup d'outliers.

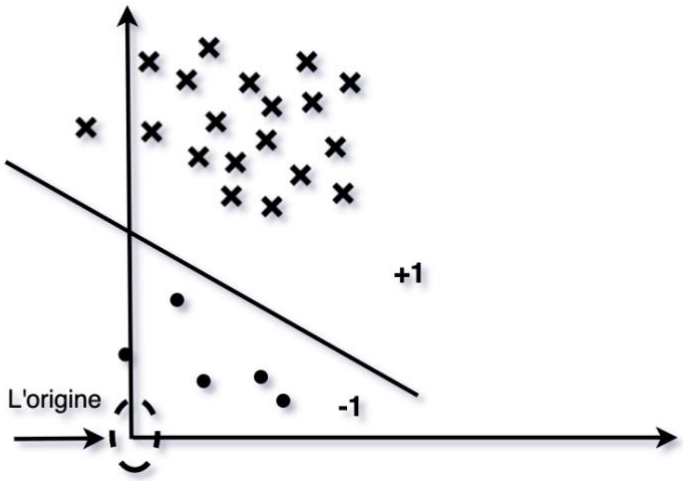


Figure 14. One-class SVM

6 Local Outlier Factor (LOF)

Local Outlier Factor (LOF) [53] est un algorithme basé sur la densité pour identifier les valeurs aberrantes locales. Un point est une valeur aberrante si la densité autour de ce point a une grande différence avec la densité autour de ses voisins. LOF s'intéresse plus à une distribution locale des données qu'à une distribution globale de celles-ci.

LOF dépend d'un seul paramètre d'entrée, nommé MinPts, qui est le nombre des plus proches voisins utilisés pour définir le voisinage local. Il se compose de 7 étapes :

- Étape 1 : Choisir MinPts comme paramètre d'entrée. K = MinPts.
- Étape 2 : Pour chaque point de données p dans l'ensemble de données D, calculer la distance entre p et tous les autres points. Définir *d(p,q)* comme la distance entre 2 points p et q. Nous pouvons choisir une métrique pour le calcul de la distance telle que Euclidienne ou Manhattan.
- Étape 3 : Trouvez le $K^{ème}$ plus proche voisin de p. Ensuite, calculer la distance de p à ce point et l'appeler *K_distance(p)*. *K_distance(p)* donne une mesure de la densité autour du point p. Lorsque *K_distance(p)* est petit, la zone autour de p est dense et inversement.
- Étape 4 : Trouver le voisinage *K_distance* de p, qui contient tous les points dont la distance à p est inférieure ou égale à *K_distance(p)*, i.e. $N_K(p) = \{q \in D \mid d(p,q) \leq K_distance(p)\}$. Notez que $|N_K(p)|$ où le nombre de points dans $N_K(p)$ peut être supérieur à K parce qu'il peut y avoir plusieurs points avec la même distance à p.
- Étape 5 : Calculer la distance d'accessibilité de p par rapport à chacun des autres points avec cette formule :

$$reach_dist\ K(p,o) = \max\{K_distance(o), d(p,o)\}$$

Pour simplifier, si p est éloigné de o, la distance d'accessibilité entre eux est leur distance réelle *d(p,o)*. Sinon, ils sont suffisamment proches, la distance réelle est remplacée par la *K-distance* de o.

- Étape 6 : Calculer la densité d'accessibilité locale (Local Reachability Density - LRD) de p, qui est une estimation de la densité au point p, définie comme l'inverse de la distance d'accessibilité moyenne des K plus proches voisins de p :

$$LRD_K(p) = \frac{|N_K(p)|}{\sum_{o \in N_K(p)} reach_dist\ K(p,o)}$$

Une faible valeur de $LRD_K(p)$ indique que p est loin de son cluster le plus proche.

- Étape 7 : Calculer LOF de p, qui détermine si un point est ou non une valeur aberrante par rapport à son voisinage. Fondamentalement, LOF(p) est la moyenne de la densité d'accessibilité locale de p et de ceux des K plus proches voisins de p :

$$LOF_K(p) = \frac{\sum_{o \in N_K(p)} \frac{LRD_K(o)}{LRD_K(p)}}{|N_K(p)|}$$

Une valeur élevée de $LOF_K(p)$ implique que la densité autour de p est très différente de la densité autour de ses K plus proches voisins. Par conséquent, p est potentiellement une valeur aberrante.

LOF fonctionne bien dans la détection des valeurs aberrantes mais le nombre de voisins *MinPts* (ou *K*) doit être choisi manuellement. Augmenter la valeur de *MinPts* ou la réduire trop pourrait conduire à une mauvaise classification.

Avantages	Inconvénients
- Résout le problème de la détection des outliers dans la densité locale. - Peut être utilisé dans une approche non supervisée et semi-supervisée. - Est viable pour les grands ensembles de données.	- La sélection de *MinPts* est une question difficile. - Est coûteux en temps de calcul et en mémoire. - Ne fonctionne pas bien lorsque l'ensemble de données a un grand nombre de dimensions mais une petite quantité de données.

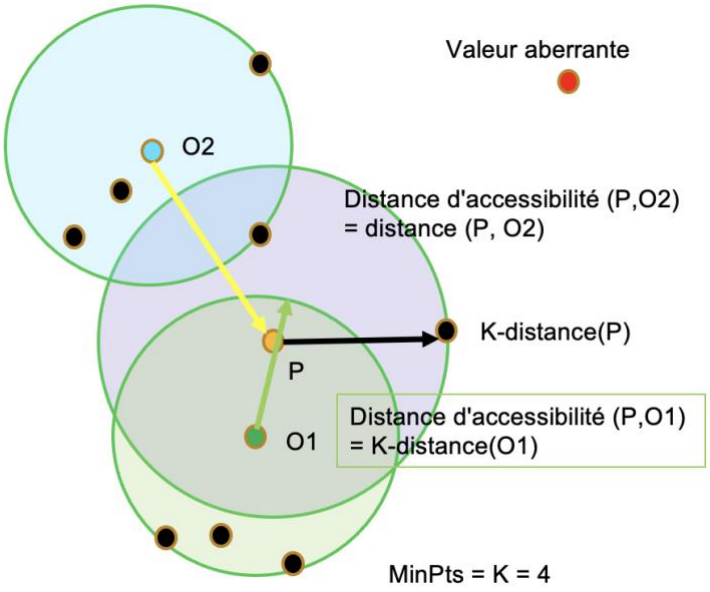


Figure 15. Local Outlier Factor

7 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [54] regroupe les points en clusters et identifie tous les points n'appartenant pas à un cluster comme des valeurs aberrantes. DBSCAN a deux paramètres qui doivent être définis :

- *minPts* : le nombre minimum de points requis pour former une région dense (un cluster).

- *eps* : une mesure de distance pour localiser les points au voisinage de n'importe quel point. Si la distance entre deux points est inférieure ou égale à cette valeur (*eps*), ces points sont considérés comme voisins.

DBSCAN étiquette les points de données comme points centraux, points frontières ou valeurs aberrantes (Figure 16). Un point donné *p* est un point central s'il a au moins des points *minPts* dans sa distance *eps* (points en rouge). Un point *q* est un point frontière s'il n'est pas un point central et se situe à l'intérieur de la distance *eps* d'un point central (point en jaune). Les points aberrants sont ceux qui ne sont ni des points centraux ni des points frontières (point en bleu).

DBSCAN contient 3 étapes :

- Étape 1 : DBSCAN choisit un point arbitrairement qui n'a pas encore été visité (jusqu'à ce que tous les points aient été visités).
- Étape 2 : Le voisinage *eps* de ce point est extrait (tous les points à l'intérieur de la distance *eps*). S'il contient suffisamment de points (*minPts*), ce point est un point central et un cluster est créé. Sinon, ce point est étiqueté comme un bruit (anomalie). Cependant, ce point peut également se trouver dans un voisinage *eps* d'un autre point, et par conséquent, ce sera un point frontière et une partie de ce cluster.
- Étape 3 : Si un point fait partie d'un cluster, son voisinage *eps* fait également partie de ce cluster. Ce processus se poursuit jusqu'à ce que tous les points soient visités.

Si la valeur de *minPts* est trop grande, DBSCAN peut détecter des anomalies mais avec un taux de faux positifs élevé. Si cette valeur est trop petite, DBSCAN ne peut pas détecter toutes les anomalies. De même, si la valeur de *eps* est trop élevée, les clusters fusionneront et DBSCAN ne pourra pas détecter toutes les anomalies. Alors que s'il est trop petit, une grande partie des données ne sera pas regroupée. Ils seront considérés comme des valeurs aberrantes car ils n'ont pas assez de points pour créer une région dense. Ainsi, DBSCAN peut détecter des anomalies mais le taux de faux positifs sera élevé. Il est donc important de choisir de bonnes valeurs pour ces paramètres.

Avantages	Inconvénients
- Il n'est pas nécessaire de connaître en avance le nombre de cluster désiré. - Permet de trouver des clusters de forme arbitraire.	- Deux paramètres doivent être définis. - Est difficile à utiliser dans les données de très grande dimension.

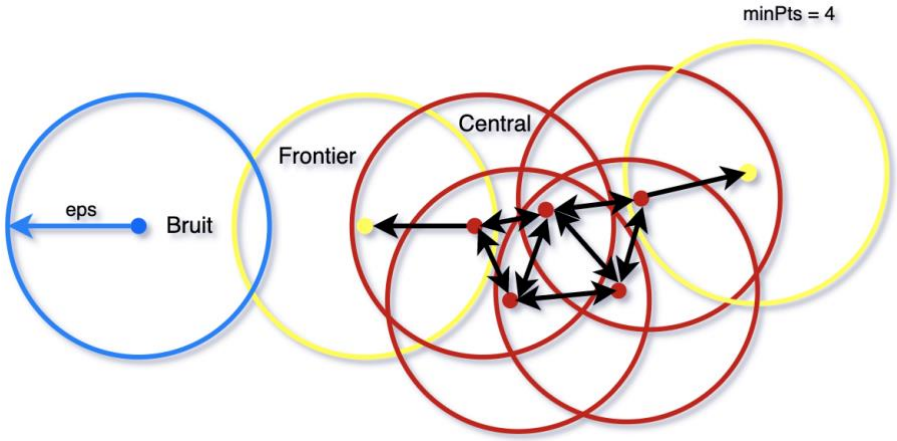


Figure 16. DBSCAN

8 Auto-Encodeur

Auto-Encodeur [55] [56] [57] est un algorithme basé sur les réseaux de neurones artificiels (Artificial Neural Network - ANN) [58] [59], qui permettent de construire une nouvelle représentation d'un jeu de données. ANN est un système informatique s'inspirant du fonctionnement du cerveau humain pour apprendre. Il se compose d'une couche d'entrée, de plusieurs couches cachées et d'une couche de sortie (Figure 17). Chaque nœud d'une couche est connecté à tous les autres nœuds de la couche suivante. Nous approfondissons le réseau en augmentant le nombre de couches cachées.

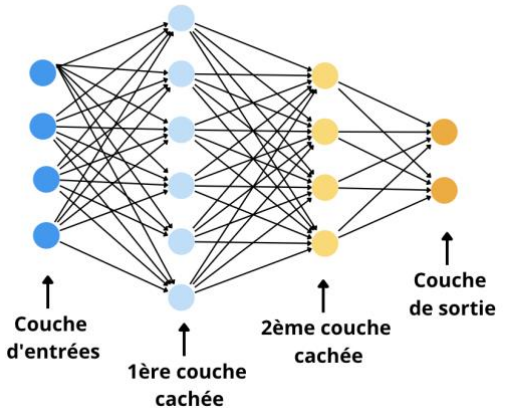


Figure 17. Réseaux de neurones artificiels (ANN) [59]

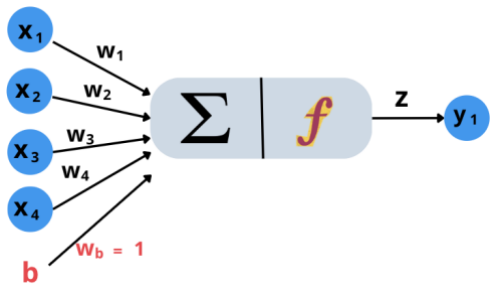


Figure 18. Créer un nouveau nœud [59]

Pour créer un nœud dans la couche cachée ou la couche de sortie, les étapes sont les suivantes (Figure 18) :

- Initialiser aléatoirement les poids pour tous les nœuds (w_1, w_2, w_3, w_4), choisir une fonction d'activation non linéaire, et bias (w_b = 1) pour aider le modèle à s'entraîner lorsque tous les poids d'entrée sont à 0.
- Toutes les entrées sont multipliées par leurs poids et calculer la somme de ces valeurs.

$$Z_1 = x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + x_4 * w_4 + w_b$$

- La fonction d'activation (f) est appliquée à l'équation linéaire Z_1. La fonction d'activation est une transformation non linéaire qui est appliquée à l'entrée avant de l'envoyer à la couche suivante.

$$Z = f(Z_1) = f(x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + x_4 * w_4 + w_b)$$

Auto-Encoder contient deux ensembles de couches de neurones : l'encodeur et le décodeur (Figure 19)

- L'encodeur est un ANN entièrement connecté pour compresser le jeu de données d'entrée en une représentation plus petite (code) dans l'espace latent. Il extrait les caractéristiques les plus importantes à partir de données initiales pour réduire la dimensionnalité du jeu de données.
- Le décodeur, qui a la structure ANN similaire, tente de décompresser le code pour reconstruire le plus fidèlement possible les données.

L'architecture du décodeur est l'image miroir de l'encodeur. La seule exigence est que la dimensionnalité de l'entrée et de la sortie doit être la même. Un bon auto-encodeur devrait être capable de compresser les données pour réduire la dimensionnalité (code), puis de décompresser ce code pour reconstruire les données sans introduire beaucoup d'erreurs. Si l'auto-encodeur introduit une erreur importante pour un point de données, ce point de données peut être une valeur aberrante.

Auto-Encoder a 4 hyperparamètres :

- Taille du code est nombre de nœuds dans l'espace latent. Une taille plus petite entraîne plus de compression.
- Nombre de couches détermine la profondeur de l'encodeur et du décodeur. Alors qu'une profondeur plus élevée augmente la complexité du modèle, une profondeur plus faible est plus rapide à traiter.
- Nombre de nœuds par couche : le nombre de nœuds par couche diminue avec chaque couche suivante de l'encodeur et augmente dans le décodeur.
- Fonction de perte (Loss function) sert à sélectionner l'auto-encodeur. Si les valeurs d'entrée sont dans la plage [0, 1], utilisez l'entropie croisée (crossentropy), sinon utilisez l'erreur quadratique moyenne (mean squared error).

Avantages	Inconvénients
- Travaille avec de grands ensembles de données. - Peut être utilisé pour la réduction de dimension.	- Quatre paramètres à définir. - Le processus de codage et décodage n'est jamais parfait. Il peut perdre des informations importantes. - Le temps de traitement est long. - Peut être difficile à interpréter.

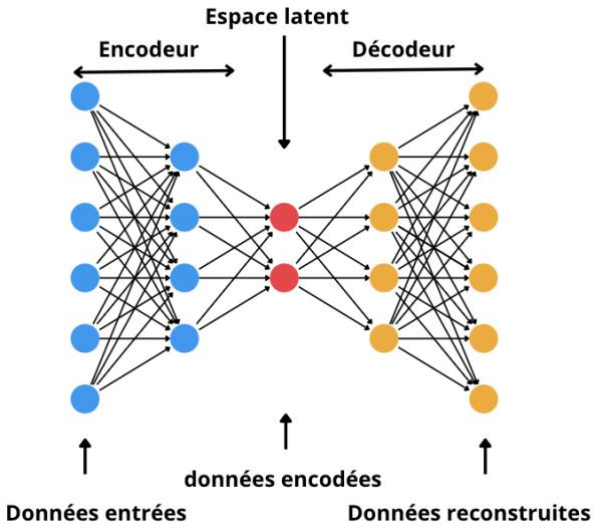


Figure 19. Auto-Encodeur [57]

9 Self Organizing Map (SOM)

Self Organizing Map (SOM) [56] [57] est une technique de visualisation de données pour comprendre les données de grande dimension en réduisant les dimensions des données à une carte. Il peut également regrouper des données similaires pour détecter des anomalies.

SOM est un type de réseau de neurones. SOM a deux couches, une couche d'entrée et une couche de sortie. Les étapes de SOM [62] pour une base de données n dimensions sont :

- Étape 1 : Initialiser aléatoirement les poids pour toutes les connections de chaque nœud (Figure 20).
- Étape 2 : Choisir un vecteur d'entrée $x = [x_1, x_2, x_3, \dots, x_n]$.
- Étape 3 : Calculer la distance entre le vecteur d'entrée et les nœuds. La distance euclidienne est donnée par :

$$Distance = \sqrt{\sum_{i=0}^{i=n} (x_i - w_i)^2}$$

- Étape 4 : Choisir le nœud qui a la plus petite distance et l'appeler Best Matching Unit (BMU).
- Étape 5 : Calculer la taille du voisinage autour de la BMU pour trouver les neurones de proximité de BMU. Cette taille diminue avec une fonction de décroissance exponentielle. Elle se rétrécit à chaque itération jusqu'à atteindre juste le BMU (Figure 21).

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\lambda}\right)$$

σ_0 : la largeur du réseau au temps zéro
t : le pas de temps courant, t = 0,1,2,3, …
λ : la constante de temps

- Étape 6 : Ajuster les poids des nœuds selon l'équation suivante :

$$Nouveaux\ poids = anciens\ poids + taux\ d'apprentissage\ (vecteur\ d'entrée - anciens\ poids)$$

Répéter à partir de l'étape 2 jusqu'à ce qu'il n'y ait aucun changement dans les poids.

Après un entraînement réussie, les vecteurs d'entrées sont visualisés d'une manière préservant la topologie, i.e. les points de données qui sont similaires les uns aux autres dans l'espace d'entrée sont appariés sur des neurones proches les uns des autres dans l'espace. Les données normales devraient former de grands et denses clusters de neurones dans la carte visualisée. Les valeurs aberrantes, d'autre part, devraient être dispersées à une grande distance des clusters denses.

Avantages	Inconvénients
- Peut visualiser des données de grande dimension pour comprendre les données. - Génère un résumé des données. - Peut être utilisé pour la réduction de dimension.	- Deux paramètres doivent être définis : la taille du voisinage de BMU au temps zéro et taux d'apprentissage. - Les neurones proches sur la carte visualisée peuvent être éloignés dans l'espace des caractéristiques. - Fonctionne mal lorsqu'il traite des données mixtes. - Le temps est extrêmement lent.

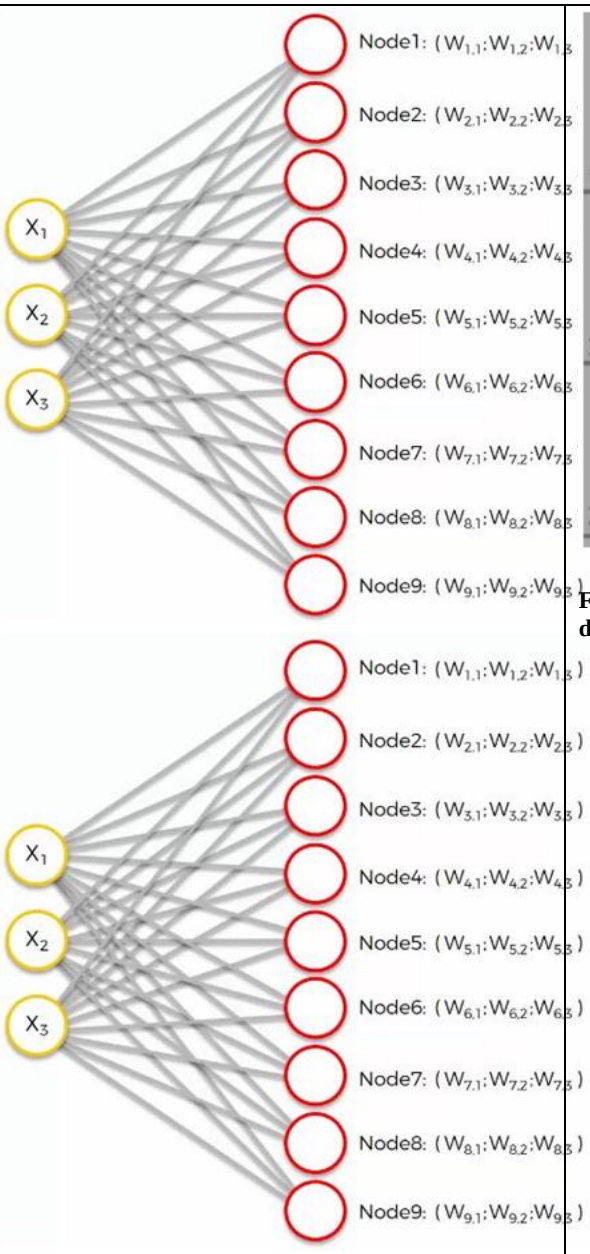


Figure 20. Calculer la distance entre le vecteur d'entrée et les nœuds [62]

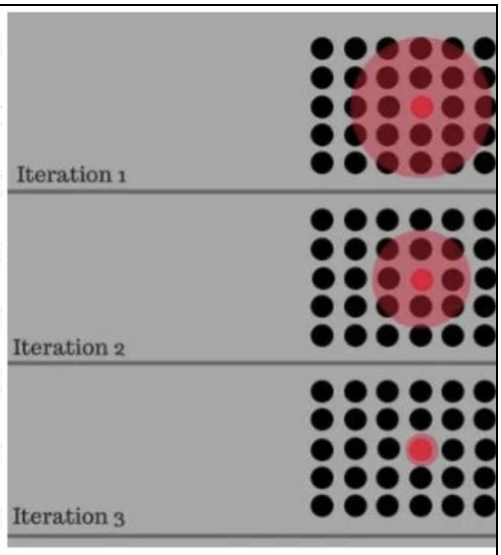


Figure 21. La taille du voisinage de la BMU diminue après chaque itération [63]

10 Conclusion

Dans ce chapitre, nous avons présenté les algorithmes d'apprentissage automatique non supervisés qui pourront détecter des anomalies : K-means, GMM, Isolation Forest, One-class SVM, LOF, DBSCAN, Auto-Encodeur, et SOM. K-means et GMM sont facile à comprendre mais ils donnent des résultats médiocres dans le cas où les données

ne sont pas linéairement séparables ou les clusters sont de forme très irrégulière. Isolation Forest et One-class SVM fonctionnent bien dans les petits ensembles. Cependant, Isolation Forest ne fonctionne pas bien lorsque des relations non linéaires existent dans les données. One-class SVM ne fonctionne pas très bien lorsque l'ensemble de données a beaucoup d'outliers. LOF et DBSCAN ne demandent pas le nombre de cluster en avance et ils sont viables pour les grands ensembles de données. Cependant, ils sont coûteux en temps de calcul et en mémoire. Auto-Encodeur et SOM peuvent travailler dans des grands ensembles de données. Ils sont normalement utilisés pour la réduction de dimension. Leurs temps de traitement sont extrêmement lents.

Dans le chapitre 3, nous présenterons une vue d'ensemble de l'utilisation des algorithmes d'apprentissage automatique pour détecter les attaques dans le réseau, en particulier les attaques utilisant la communication C&C basée sur DNS.

Chapitre 3. État de l'art des techniques d'apprentissage automatique pour la détection d'attaques réseau

1 Introduction

Devant la difficulté à détecter des attaques, de plus en plus de travaux proposent d'utiliser les algorithmes d'apprentissage automatique. Dans cette section, nous résumons les travaux antérieurs relatifs à l'utilisation d'approches d'apprentissage automatique pour la détection d'attaques réseau. Tout d'abord, nous énumérons les approches d'apprentissage automatique non supervisées pour détecter les intrusions génériques sur le réseau. Dans un deuxième temps, nous présentons des approches d'apprentissage automatique pour détecter spécifiquement le trafic DNS anormal, en particulier les tunnels DNS.

2 Approches non supervisées pour détecter des attaques dans le réseau

Il existe de nombreux travaux appliquant des algorithmes d'apprentissage automatique non supervisés pour détecter les intrusions sur le réseau (Tableau 1).

Kazato et al. [64] décrit une méthode pour estimer la malveillance des indicateurs de compromission (IoCs). Cette méthode contient trois étapes principales : recueillir des informations relationnelles et extraire des caractéristiques IoC individuelles ; construire un graphique IoC en utilisant les nœuds IoC existants dans l'IoC graphique pré-construit ; estimer la malveillance IoC basée sur les réseaux convolutifs de graphes. Les auteurs utilisent deux caractéristiques dans le réseau convolutif de graphes pour améliorer la précision de l'estimation, qui sont les fonctions IoC individuelles et les relations entre les IoC. Ils créent un ensemble de données de 20 000 noms de domaine (résolus par DNS publiés sur Internet), 10 000 noms de domaine bénins (répertoriés sur Alexa Top Sites), 10 000 noms de domaine malveillants (résolubles depuis Abuse.ch ZeuS Tracker et DNS - BH Malware Domain Blocklist) pour évaluer leur approche, et le taux de détection est de 88%.

Chen et al. [65] ont proposé le modèle de deep autoencoding Gaussian Mixture Model assisté par Federated Learning (FDAGMM) pour la détection d'anomalies de réseau. Il s'agit d'une amélioration du DAGMM (Deep Autoencoding Gaussian Mixture Model) en utilisant l'apprentissage fédéré afin de gérer les scénarios où les données de formation sont insuffisantes parce que les utilisateurs normaux ne veulent pas ou ne peuvent pas partager des informations privées. FDAGMM est une combinaison de

trois algorithmes : Auto-encodeur pour réduire la dimension, Gaussian Mixture Model (GMM) pour estimer la densité et Federated Learning pour améliorer ses performances avec plus de données tout en protégeant la confidentialité. Ils évaluent cette méthode sur la base de données KDD CUP 1999, et le taux de détection est de 98.03%.

J.K.M. Perera et al. [66] ont introduit une solution de classification du trafic basée sur l'apprentissage automatique pour la mise en reseau logicielle (Software Defined networking - SDN) qui consiste en deux processus. Dans le processus hors connexion, ils utilisent K-means pour regrouper et étiqueter l'ensemble de données, puis ils utilisent cet ensemble de données pour former plusieurs modèles de classification (SVM, Decision Tree, Random Forest, KNN). Dans le processus en ligne, le modèle de classification est utilisé pour classer le trafic réseau entrant en temps réel. Ils évaluent leur approche sur l'ensemble de données "IP Network Traffic Flows, Labeled with 75 Apps" de Kaggle. Avec SVM, la méthode proposée donne le meilleur résultat avec une précision de 96.37 %.

Aliakbarisani et al. [67] proposent une méthode d'apprentissage métrique linéaire pour les systèmes de détection d'anomalies de réseau (Network Anomaly Detection Systems - NADS) non supervisés, qui utilise des méthodes de regroupement ou de classification basés sur la distance. Cette méthode d'apprentissage métrique est le processus d'apprentissage automatique d'une fonction de distance spécifique à l'application. Elle utilise des connaissances préalables sous la forme de contraintes de distance relative ou de contraintes de similarité/dissemblance. Cependant, le NADS est basé sur le clustering non supervisé, donc il n'y a pas d'informations prédéfinies. Les auteurs tentent d'extraire directement des contraintes de similarité/dissemblance à partir des échantillons d'apprentissage, c'est-à-dire les vecteurs de caractéristiques non étiquetés des connexions réseau. NADS contient les phases de formation et de test. La phase de formation comporte cinq étapes : filtrage, apprentissage métrique, transformation, regroupement et estimation des limites. Ils utilisent un SVM à classe unique dans l'estimation des limites. Et la phase de test se compose de trois étapes : transformation, division et classification. Ils évaluent leur NADS sur les jeux de données Kyoto 2006+ [68] et NSL-KDD [69]. Avec l'apprentissage métrique, les algorithmes peuvent améliorer les performances.

Alam et al. [70] ont présenté un système de détection d'intrusions et d'anomalies dans le réseau en temps réel qui utilise un Auto-encodeur basé sur un memristor. Auto-encodeur est un algorithme non supervisé qui peut reconnaître rapidement et avec précision les données anormales. Memristor est une nouvelle classe de dispositifs semi-conducteurs à l'échelle nanométrique et peut effectuer en parallèle de nombreuses opérations de multiplication-addition dans le domaine analogique. Ainsi, l'utilisation de dispositifs memristor pour effectuer des calculs d'auto-encodeurs permet de présenter des systèmes à très faible consommation pour la détection

d'intrusions et d'anomalies en temps réel. Ils utilisent la base de données NSL-KDD pour l'évaluation, et la précision de détection globale de ce système est de 92.91%.

Xiaofei et al. [71] passent en revue les différents types de SOM pour la détection d'intrusion réseau : le SOM statique - HSOM (Hierarchical Self-Organizing Maps) peut réduire efficacement les frais de calcul et représenter efficacement la hiérarchie des données ; le SOM dynamique - GHSOM (Growing Hierarchical Self-Organizing Maps) est assez efficace pour la détection d'intrusion en ligne avec une faible latence de calcul, une auto-adaptabilité dynamique et un auto-apprentissage ; et le SOM hybride est une combinaison avec un autre algorithme (par exemple K-means). Ils comparent les méthodes sur la base de données KDD99. GHSOM donne un taux de vrais positifs de 99% mais un taux de faux positifs de 3.72% tandis que HSOM ne donne qu'un taux de détection de 90.4% et un taux de faux positifs de 1.38%.

Carrasco et al. [72] ont présenté le modèle Skip-gram, une variante de l'algorithme word2vec, qui peut modéliser le comportement légitime du réseau pour détecter les intrusions sur le réseau. Skip-gram prédit les mots de contexte à partir d'un mot cible donné. Les auteurs ont modifié skip-gram avec l'hypothèse que les systèmes et leurs connexions réseau peuvent être représentés efficacement sous forme de mots à partir d'un texte à des fins de modélisation du comportement. L'ensemble de données UNSW-NB 15 a été utilisé pour l'évaluer, avec une précision de 99.20% et un taux de faux positifs de 0.61%.

Shone et al. [73] ont introduit une nouvelle méthode NDAE (Non-symmetric Deep Auto-Encoder) pour l'apprentissage non supervisé des caractéristiques : NDAE est un encodeur automatique avec plusieurs couches cachées non symétriques, c'est-à-dire il ne contient que la phase de l'encodeur. Cette technique réduit à la fois les coûts de calcul et de temps, avec un impact minimal sur la précision et l'efficacité. En outre, ils présentent un nouveau modèle de classification construit à l'aide de NDAE empilés et de l'algorithme de classification Random forest (Random forest - Non-Symmetric Deep Auto-Encoder - S-NDAE). Les NDAE empilés permettent d'apprendre les relations complexes entre différentes caractéristiques afin de choisir les plus importantes. Ils forment le classificateur Random forest en utilisant les représentations codées apprises par les NDAE empilés pour classer le trafic réseau en données normales et en attaques connues. Ils comparent les performances de S-NDAE avec Deep Belief Networks (DBNs) sur deux bases de données, KDD cup'99 et NSL-KDD. S-NDAE améliore la précision et réduit le temps de formation avec un taux de détection de 97.85 %.

Amoli et al. [74] ont proposé un NIDS temps réel non supervisé pour les réseaux à haut débit. Ce NIDS contient deux moteurs distincts qui permettent la détection des intrusions sur le réseau. Le premier moteur surveille le comportement du réseau pour détecter les intrusions en temps réel à l'aide de DBSCAN avec un seuil automatisé qui

s'adapte à l'état actuel du réseau pour déterminer le volume normal attendu du trafic réseau à l'avenir. Le deuxième moteur a pour objectif de détecter le botnet interne. Ce moteur qui a besoin de plus de temps pour observer suffisamment d'informations afin de trouver l'éventuel botmaster. À chaque fois que le premier moteur classe une attaque comme DDOS, il envoie les adresses IP des attaquants (bots éventuels) au deuxième moteur pour analyser la possibilité de trouver le botmaster. En général, le serveur C&C communique plusieurs fois avec les bots au cours de leur vie. Ainsi, à partir des adresses IP des attaquants reçues du premier moteur, le deuxième moteur crée une liste d'adresses IP des ordinateurs qui ont communiqué aux attaquants et les ajoute dans la liste des serveurs C&C éventuels (botmaster). Ensuite, DBSCAN est utilisé pour détecter les outliers et il considère qu'il y a de fortes chances de trouver les flux entre le serveur C&C et les bots dans les outliers. Ainsi, quand le deuxième moteur trouve qu'une adresse IP de la liste des serveurs C&C éventuels correspond aux adresses IP des outliers, il considérera cette adresse IP pour le C&C et la signalera à l'administrateur. Le deuxième moteur regroupe des fonctionnalités réseau spécifiques pour détecter les communications C&C de botnet centralisées et décentralisées. Deux ensembles de données ont été utilisés pour évaluer l'approche proposée : DARPA et ISCX [75]. Le modèle proposé a atteint une précision de 98.39% et un taux de faux positifs de 3.61%.

Casas et al. [76] ont présenté UNIDS, un système de détection d'intrusion réseau non supervisé capable de détecter des attaques réseau inconnues sans utiliser aucun type de signatures, de trafic étiqueté ou de formation. UNIDS contient trois étapes consécutives. Tout d'abord, il utilise un algorithme standard de détection de changement sur trois métriques de volume simples et classiques (octets, paquets et flux par tranche de temps) pour détecter une tranche de temps anormale dans laquelle l'analyse de clustering sera effectuée. Deuxièmement, il utilise un algorithme de multi-clustering capable de détecter des outliers. Cet algorithme est basé sur une combinaison de techniques de Sub-Space Clustering (SSC), de clustering basé sur la densité (Density-based Clustering) et de Evidence Accumulation Clustering (EAC) pour identifier les flux aberrants. De plus, cet algorithme est également utilisé pour classer le degré d'anormalité de tous les flux aberrants identifiés, en construisant un classement des outliers. Enfin, un seuil prédéfini permet de déterminer les anomalies. Ils évaluent l'UNIDS sur l'ensemble de données KDD Cup 1999 où le taux de détection est supérieur à 90% et le taux de faux positifs inférieur à 1%.

Zanero et Serazzi [77] ont proposé une architecture à deux niveaux qui est capable de regrouper les charges utiles des paquets et de corréler les anomalies dans le flux de paquets pour la détection d'intrusions sur le réseau. Le premier niveau du système classe la charge utile des paquets à l'aide d'un algorithme de clustering non supervisé - SOM. Le deuxième niveau détecte les anomalies dans chaque paquet et dans une séquence de paquets avec une version modifiée de l'algorithme SmartSifter qui est un algorithme d'apprentissage d'actualisation. Afin d'ajuster automatiquement le seuil au-

delà duquel un vecteur de données doit être considéré comme une valeur aberrante, les auteurs ont modifié SmartSifter en introduisant une phase d'apprentissage au cours de laquelle la distribution des scores d'anomalie est approchée, et un quantile estimé de la distribution est également calculé. De plus, ils proposent des améliorations et des heuristiques pour augmenter le débit de SOM à un débit adapté à la détection d'intrusion en ligne. Ils utilisent la base de données DARPA 1999 [16] pour évaluer, le taux de détection n'est que de 66.7%.

Gunes Kayacik et al. [78] ont présenté une approche de Self- Organizing Feature Maps (SOM) hiérarchiques qui contient 3 couches : dans les couches 1 et 2, des SOM relativement petits sont utilisés (6 × 6); dans la dernière couche, les SOM sont plus grands (20 × 20). Ils construisent les architectures SOM hiérarchiques avec deux algorithmes d'apprentissage : le SOM est utilisé à chaque couche de la hiérarchie ; le clustering des fonctions potentielles est utilisé pour quantifier le nombre de neurones SOM "perçus" par la deuxième couche pour l'architecture à 6 caractéristiques. Pour évaluer cette méthode, ils appliquent cette hiérarchie SOM à deux couches à la base de données KDD CUP 1999 qui contient 41 caractéristiques. Dans le cas de 41 caractéristiques, la hiérarchie SOM a besoin de deux couches : la première couche repose sur six caractéristiques de base ; la deuxième couche repose sur neuf caractéristiques de base et 32 caractéristiques dérivées. La méthode proposée a atteint un taux de détection de 90% et un taux de faux positifs de 1.38%.

Wang et Battiti [79] ont proposé une nouvelle méthode d'identification d'intrusion basée sur Principal Component Analysis (PCA). Il crée deux profils : le premier représente le comportement normal appris sur la base de données de trafic authentiques et l'autre correspond au comportement d'attaque de chaque type d'attaque basé sur un type de données d'attaque associé. Pour classer une nouvelle donnée, ils comparent la distance avec le comportement normal et le comportement d'attaque pour décider si cette nouvelle donnée est une attaque ou non. La méthode a été testée sur l'ensemble de données 1998 DARPA, le taux de détection est de 98.8% et le taux de faux positifs est de 0.4%. Cette approche n'est pas une approche purement non supervisée et nécessite un ensemble de données d'apprentissage.

Zhang et Zulkernine [80] ont appliqué l'un des algorithmes d'exploration de données efficaces appelés algorithme de forêts aléatoires dans les NIDSs basés sur des anomalies. D'abord, le NIDS capture le trafic réseau et construit un ensemble de données. Ensuite, les modèles basés sur le service sont construits sur l'ensemble de données à l'aide de l'algorithme de forêts aléatoires. Avec ces modèles construits, nous pouvons trouver des valeurs aberrantes liées à chaque modèle. Enfin, le système déclenchera des alertes lorsque des valeurs aberrantes seront détectées. Ils ont développé NIDSs en utilisant l'environnement Waikato pour l'analyse des connaissances (WEKA). L'approche proposée est évaluée à l'aide de l'ensemble de

données KDD'99, le taux de détection est de 95% tandis que le taux de faux positifs est de 1%.

Leon et al. [81] ont présenté une nouvelle approche de détection d'anomalies basée sur Unsupervised Niche Clustering (UNC) et l'ont appliquée à la détection d'intrusion réseau. L'UNC est une technique de niche génétique qui peut gérer le bruit, est capable de déterminer automatiquement le nombre de clusters. Les auteurs caractérisent chaque cluster prédit à l'aide d'une fonction d'appartenance floue (fuzzy membership function) qui suit une forme gaussienne définie par les centres et rayons de cluster évolués. De plus, ils utilisent le raffinement de Maximal Density Estimator (MDE) pour améliorer la qualité de la solution et Principal Component Analysis (PCA) pour réduire la complexité du jeu de données et améliorer encore les performances de l'approche proposée. Le modèle a été testé sur le jeu de données KDD Cup 1999 [82] et montre un taux de détection de 99.2% et un taux de faux positifs de 2.2%.

Tableau 1. Approches non supervisées pour détecter des attaques dans le réseau

	Auteurs	Année	Algorithme	Jeu de données	Résultats
1	Kazato et al. [64]	2020	GCN	Propriétaire	DR : 88%
2	Chen et al. [65]	2020	Auto-encodeur, GMM, et Federated Learning	KDD CUP 1999	DR : 98.03%
3	J.K.M. Perera et al. [66]	2020	K-means, SVM, Decision tree, Random forest, KNN	IP Network Traffic Flows, Labeled with 75 Apps	SVM DR : 96.37%

4	Aliakbarisani et al. [67]	2019	SVM	Kyoto 2006+, NSL-KDD	
5	Alam et al. [70]	2019	Auto-encodeur et Memristor	NSL-KDD	DR : 92.91%
6	Xiaofei et al. [71]	2019	HSOM, GHSOM	KDD99	GHSOM DR : 99% FPR : 3.72% HSOM DR : 90.4% FPR : 1.38%
7	Carrasco et al. [72]	2018	Word2vec	UNSW-NB 15	DR : 99.2% FPR : 0.61%
8	Shone et al. [73]	2018	Auto-encodeur, Random forest, DBNs	KDD cup'99 et NSL-KDD	S-NDAE DR : 97.85%
9	Amoli et al. [74]	2016	DBSCAN	DARPA, ISCX	DR : 98.39% FPR : 3.61%
10	Casas et al. [76]	2012	SSC et EAC	KDD Cup 1999	DR : 90% FPR < 1%
11	Zanero and Serazzi [77]	2008	K-means, PDDP, SOM et SDLE	DARPA 1999	DR : 66.7%
12	Gunes Kayacik et al. [78]	2007	SOM et clustering des fonctions potentielles	KDD CUP 1999	DR : 90% FPR : 1.38%

13	Wang and Battiti [79]	2006	PCA	1998 DARPA	DR : 98.8% FPR : 0.4%
14	Zhang et Zulkernine [80]	2006	Random forest	KDD'99	DR : 95% FPR : 1%
15	Leon et al. [81]	2004	PCA et UNC	KDD Cup 1999	DR : 99.2% FPR : 2.2%

Avec l'évolution rapide et la prolifération des attaques complexes telles que les botnets, les APTs deviennent des cybermenaces de plus en plus dangereuses et sérieuses. Les technologies de sécurité basées sur le réseau telles que les systèmes de détection d'intrusion (IDS), les pare-feux sont améliorés pour protéger les systèmes informatiques et les réseaux contre les attaquants sur l'Internet. En particulier, de nombreux efforts ont été faits, comme indiqué ci-dessus, pour appliquer des techniques d'apprentissage automatique afin d'améliorer la détection des attaques dans les réseaux. Les auteurs ont utilisé la plupart des techniques non supervisées et obtiennent de bons résultats. Cependant, ils évaluent leur travail sur d'anciens bases de données comme DARPA 1999, KDD 1999, Kyoto 2006+, etc. Ces bases de données ont été construites il y a longtemps et ne peuvent pas refléter les situations actuelles du réseau et les dernières tendances en matière d'attaques. Les attaques dans ces bases de données ne sont pas représentatives de ce que nous voulons.

Notre objectif est de détecter des attaques sophistiquées et actualisées. Pour échanger facilement des données entre le serveur de l'attaquant et la machine infectée, les attaquants créent souvent des tunnels C&C et les cachent dans les protocoles autorisés. Nous essayons de détecter ces tunnels C&C.

À notre connaissance, il y a aussi des travaux pour tenter de détecter des tunnels C&C, voir dans la partie suivante. Cependant, ils utilisent des approches supervisées et ils ont besoin d'un jeu de données pour apprendre. Cela a un coût pour labelliser.

3 Détection de communications C&C encapsulées dans DNS

Nous présentons dans cette section les travaux dédiés à la détection de tunnels DNS. Il est à noter qu'ils sont tous basés sur des algorithmes supervisés (Tableau 2).

Jitesh Miglani and Christina Thorpe [83] ont proposé une approche active de détection du tunnel DNS en capturant tous les paquets d'un réseau local et en utilisant des modèles d'apprentissage automatique pour détecter les données tunnellisées. Dans la première phase, ils ont collecté des données et créé un ensemble de données. Pour

diversifier l'ensemble de données, ils utilisent différents protocoles de couche d'application sur le tunnel DNS pour collecter les requêtes DNS tunnellisées, notamment HTTP, HTTPS, FTP et UDP. Ils utilisent un jeu de données qui contient des trafics DNS capturés pendant 10 jours d'avril à mai 2016 [84] comme requêtes DNS légitimes. Ils ont évalué quatre classificateurs d'apprentissage automatique supervisé (Decision tree, Random forest, Gradient Boosting et AdaBoost). Sur ces quatre modèles, tous sauf Decision tree étaient des techniques d'apprentissage d'ensemble. Decision tree a eu les performances les plus faibles. AdaBoost donne le meilleur résultat avec un taux de détection de 99% à 100%.

Franco Palau et al. [85] ont proposé une approche de détection basée sur un réseau de neurones convolutifs (Convolutional Neural Networks - CNN) avec une complexité d'architecture minimale. Ils ont utilisé une grille de recherche traditionnelle sur un ensemble d'apprentissage pour trouver les paramètres optimaux et le meilleur modèle résultant a été évalué sur un ensemble de test. Ils ont créé un nouvel ensemble de données contenant un tunnel DNS généré avec cinq outils de tunnel DNS : iodine, dns2tcp, dnscat2, tuns et DNSexfiltrator. Le taux de détection est de 92% et le taux de faux positifs est de 0.8%.

MontazeriShatoori et al. [86] ont introduit une nouvelle approche à deux couches pour classer le trafic DoH du trafic non-DoH dans la première couche et caractériser le trafic DoH (DoH bénin et DoH malveillant) dans la deuxième couche. Ils ont généré un nouvel ensemble de données étiqueté, nommé CIRA-CIC-DoHBrw-2020, en capturant le trafic chiffré selon trois classes Benign-DoH, Malicious-DoH et non-DoH dans le réseau. Ils ont utilisé le même ensemble de caractéristiques et les mêmes classificateurs (Random forest, C4.5, SVM, NB, DNN, 2D CNN) à deux couches. Random Forest et C4.5 donnent les meilleurs résultats avec un taux de détection à 99,3%. De plus, ils ont proposé un nouvel ensemble de caractéristiques basé sur la représentation chronologique des flux de trafic en introduisant le concept d'agrégats de paquets et en démontrant l'efficacité de cet ensemble de fonctionnalités dans la caractérisation du trafic crypté en utilisant la mémoire à long court terme (Long Short-Term Memory - LSTM), une importante architecture de réseau neuronal récurrent (Recurrent Neural Network - RNN), pour créer des classificateurs binaires d'apprentissage en profondeur aux deux couches. LSTM donne une image ostensible de la façon dont la valeur de toutes les métriques d'évaluation continue d'augmenter avec le nombre d'agrégats formés à une très courte durée de flux.

Banadaki and Robert [87] ont proposé une approche systématique avec des modèles d'apprentissage automatique à deux couches : la couche 1 pour détecter le trafic DNS sur HTTPS (DoH) et la couche 2 pour distinguer le trafic DoH normal du trafic DoH malveillant. Ils ont utilisé des modèles d'apprentissage automatique dans la plate-forme IBM pour identifier le meilleur type de modèles pour les données et comparer les performances de six modèles d'apprentissage automatique (Decision tree,

Extremely randomized Trees (Extra Trees), Gradient Boosting, XGBoost (XGB), Light Gradient Boosting Machine (LGBM) et Random forest). Ils ont évalué leur approche sur l'ensemble de données CIRA-CIC-DoHBrw-2020. Les algorithmes LGBM et XGBoost donnent les meilleurs résultats que les autres algorithmes avec un taux de détection maximum à 100% dans les tâches de classification des couches 1 et 2.

Singh and Roy [88] ont utilisé de nombreux classificateurs d'apprentissage automatique tels que Naive Bayes, Logistic Regression, Random Forest, K-Nearest Neighbor, Gradient Boosting pour détecter le trafic DNS malveillant dans l'environnement DoH. Ils ont évalué ces méthodes sur l'ensemble de données CIRA-CIC-DoHBrw-2020. Les classificateurs Random Forest et Gradient Boosting donnent les meilleurs résultats avec un taux de détection à 100%.

Lin et al. [89] ont proposé une méthode de détection de tunnel de couche applicative en appliquant un filtrage de nom de domaine basé sur des règles pour l'algorithme de Domain Generation Algorithm (DGA) basé sur un modèle de trigramme et l'apprentissage automatique. Lorsque le nom de domaine adopté par les données de communication ne satisfait pas à la règle de filtrage des noms de domaine DGA, ces données sont directement bloquées. Sinon, un modèle d'apprentissage automatique sera utilisé pour les analyser plus en détail. Ils ont testé l'efficacité de la méthode proposée en menant des expériences sur des tunnels DNS, HTTP et HTTPS. Dans chaque type de tunnel, ils ont appliqué et comparé six algorithmes de classification (Gaussian Bayes, SVM, C4.5, Random Forest, GBDT, XGboost). À l'exception de Gaussian Bayes, les résultats de classification des cinq algorithmes d'apprentissage automatique étaient bons avec un taux de détection de 99.9% et une précision de 99.8%.

Berg et al. [90] ont comparé les différents modèles (réseau de neurones multinomial et Random forest) sur l'ensemble de caratéristiques de Request et Response dans le trafic DNS comme la longueur du paquet IP, la longueur du nom et l'entropie du nom. Les expériences ont montré que les performances des modèles augmentent lors de la formation des modèles sur les paires de requêtes et de réponses plutôt que d'utiliser uniquement des requêtes ou des réponses. De plus, les auteurs ont présenté un outil d'enquête réseau (Network Forensic Tool) qui peut être utilisé comme un outil post mortem afin de détecter le trafic DNS tunnelisé et est capable d'analyser les requêtes DNS et les réponses par paires. Cet outil permet d'extraire des caractéristiques à partir de fichiers pcap, d'entraîner des modèles sur ces caractéristiques et de présenter les résultats dans des tables Latex. La précision des modèles est supérieure à 83 % et le taux de détection est de 100 %.

Almusawi et Amintoosi [91] ont introduit une classification multilabel utilisant le noyau SVM qui peut différencier non seulement les trafics légitimes et les trafics de

tunneling, mais aussi les types de tunneling (HTTP, HTTPS, FTP, POP3). Ils comparent les performances de deux modèles (le noyau SVM proposé et le classificateur Bayesian multilabel) sur un ensemble de données de trafic tunnelisé DNS qui a été créé à partir des outils de tunnellisation DNS iodine et dns2tcp. Les résultats ont montré que le noyau SVM proposé fonctionnait mieux que le classificateur de Bayes avec un DR compris entre 62.7% et 97.5%.

Homem and Papapetrou [92] ont introduit une approche d'apprentissage automatique, basée sur trois caractéristiques de longueur de paquet IP, d'entropie de nom de requête DNS et de longueur de nom de requête DNS, pour identifier les protocoles (HTTP, HTTPS, FTP, POP3) transportés dans le tunnel DNS. Ils comparent les performances de quatre modèles de classification (Decision tree, Support vector machines, KNN et réseaux de neurones) sur un ensemble de données de trafic tunnelisé DNS. Le modèle Réseaux de neurones donne le meilleur résultat avec un taux de détection allant de 89% à 100%.

Do and al.[93] ont proposé d'utiliser One Class Support Vector Machine (OCSVM) et K-means dans la détection du tunneling DNS dans les réseaux mobiles. Ils ont généré un jeu de données DNS avec des caractéristiques spécifiques telles que l'heure, la source, la destination, le protocole, lengthUp, lengthDown, info, l'étiquette de la requête DNS. K-means avec un réglage initial sur k-means++ donne le meilleur résultat avec un taux de détection de 93%. OCSVM avec le noyau réglé sur polynôme donne le meilleur résultat avec un taux de détection de 95%.

Buczak et al.[94] ont introduit une méthode basée sur l'algorithme Random Forest pour détecter la présence de tunnels DNS dans le trafic réseau. Les auteurs ont utilisé la méthode Gain Ratio pour trouver les caractéristiques avec un pouvoir discriminant suffisant pour la classification. La méthode proposée est évaluée sur un ensemble de données DNS qu'ils ont créé avec quatre outils de tunnellisation DNS (Iodine, Dnscat2, Cobalt Strike, Pick Pocket). Le taux de détection est de 95.4%.

Aiello and al. [95] ont présenté une approche d'apprentissage automatique pour détecter le tunnel DNS en examinant de simples propriétés statistiques des messages de protocole comme des statistiques de temps d'inter-arrivée de paquets et de tailles de paquets. L'approche contient deux niveaux de classification. Dans le premier niveau de classification, ils ont utilisé quatre classificateurs de base basés sur l'apprentissage automatique : Bayes, KNN, Neural Networks, et SVM. Dans le deuxième niveau de classification, les auteurs ont répété les classifications sur différentes données en prenant une décision commune collective de leur part avec un mécanisme de vote à la majorité simple sur ces données. Ils ont évalué leur approche sur un ensemble de données DNS (50% tunnel, 50 % normal) qu'ils ont créé dans leur institut. Les Neural Networks donnent le meilleur résultat avec une précision entre 98.7% et 100%.

Tableau 2. Approches de détection de communications C&C encapsulées dans DNS

	Auteurs	Année	Algorithme	Jeu de données	Résultats
1	Jitesh Miglani and Christina Thorpe [83]	2021	Decision tree, Random forest, Gradient boosting et AdaBoost	Propriétaire	AdaBoost DR : 99%
2	Franco Palau et al. [85]	2021	CNN	Propriétaire	DR : 92% FPR : 0.8%
3	MontazeriShatoori et al. [86]	2020	Random forest, C4.5, SVM, NB, DNN, 2D CNN, LSTM	CIRA-CIC-DoHBrw-2020	Forêt aléatoire DR : 99.3%
4	Banadaki and Robert [87]	2020	Decision tree, Extra Trees, Gradient Boosting, XGBoost, LGBM et Random forest	CIRA-CIC-DoHBrw-2020	LGBM DR : 100%
5	Singh and Roy [88]	2020	Naive Bayes, Logistic Regression, Random Forest, K-Nearest Neighbor, Gradient Boosting	CIRA-CIC-DoHBrw-2020	Random Forest et Gradient Boosting DR : 100%
6	Lin et al. [89]	2019	DGA, Gaussian Bayes, SVM,	Propriétaire	DR : 99.9%

			C4.5, Random Forest, GBDT, XGboost		Précision : 99.8%
7	Berg et al. [90]	2019	réseau de neurones multinomial et Random forest	Propriétaire	DR : 100% Précision : 83%
8	Almusawi et Amintoosi [91]	2018	SVM, Bayesian	Propriétaire	DR : 62.7% - 97.5%
9	Homem and Papapetrou [92]	2017	Decision tree, Support vector machines, KNN et réseaux de neurones	Propriétaire	Réseaux de neurones DR : 89%-100%
10	Do and al.[93]	2017	OCSVM, K-means	Propriétaire	K-means DR : 93% OCSVM DR : 95%
11	Buczak et al.[94]	2016	Random Forest	Propriétaire	DR : 95.4%
12	Aiello and al. [95]	2015	Bayes, KNN, Neural Networks, SVM	Propriétaire	Neural Networks Précision : 98.7% - 100%

Nous pouvons constater que différents chercheurs ont proposé d'utiliser les techniques d'apprentissage automatique pour détecter le tunnel DNS. Ils ont utilisé des méthodes

d'apprentissage supervisé, sauf K-means qui est un algorithme très simple, et obtenu de bons résultats de détection avec la majorité des taux de détection supérieurs à 90 %. Cependant, ces méthodes nécessitent un jeu de données d'apprentissage qui doit être exhaustif. De plus, ils utilisent des ensembles de données non publics à l'exception de CIRA-CIC-DoHBrw-2020. Il est donc difficile de reproduire les expériences.

4 Conclusion

Dans ce chapitre, nous avons présenté des travaux connexes en trois groupes. Dans le premier groupe, des approches d'apprentissage automatique non supervisées pour détecter les intrusions dans le réseau, les auteurs ont utilisé la plupart des techniques non supervisées et obtiennent de bons résultats. En revanche, ils ont utilisé des anciens jeux de données qui ne contiennent pas d'attaques complexes comme les tunnels DNS. Dans le deuxième groupe, des approches d'apprentissage automatique pour détecter le trafic DNS anormal, les auteurs ont utilisé des techniques d'apprentissage supervisées et obtenu de bons résultats de détection. Cependant, les méthodes supervisées nécessitent un jeu de données d'apprentissage. Ils utilisent des jeux de données non publics à l'exception de CIRA-CIC-DoHBrw-2020. Il est donc difficile de reproduire les expériences. C'est pourquoi dans le chapitre 4, nous proposons une approche d'analyse comportementale basée sur la méthode DBSCAN pour détecter les tunnels DNS malveillants dans le réseau.

Chapitre 4. Automatisation de DBSCAN pour la détection de tunnels DNS

1 Introduction

La détection de tunnels DNS par des outils classiques tels que les systèmes de détection d'intrusion ou les SIEMs ne peut être envisagée. En effet, mettre en place des listes noires d'adresses IP suspectes ou des noms de domaine suspects peut très facilement être contourné. De même, définir un seuil quant à la quantité d'octets échangés est difficile surtout lorsque les flux DNS sont encapsulés dans le protocole HTTPS comme pour DoH.

Il est alors nécessaire d'étudier la combinaison d'un grand nombre de paramètres associés aux flux de données pour étudier le comportement réseau. Définir manuellement des seuils avec un grand nombre de paramètres est alors impossible. L'utilisation d'algorithme d'apprentissage automatique devient nécessaire. Comme nous l'avons montré dans le chapitre précédent, de nombreux travaux utilisant des algorithmes d'apprentissage automatisés ont été proposés pour la détection d'intrusions. Cependant, les travaux dédiés à la détection de tunnels DNS utilisent tous des algorithmes d'apprentissage supervisé. Or la pertinence de ces algorithmes dépend de l'exhaustivité du jeu de données d'apprentissage. Les algorithmes d'apprentissage non supervisés ne souffrent pas de cette contrainte. Néanmoins, il n'y a pas de travaux utilisant cette approche pour la détection de tunnels DNS.

Ce chapitre présente notre démarche nous ayant conduit à bâtir une solution autour de l'algorithme DBSCAN. Toutefois DBSCAN demande de prédéfinir deux hyperparamètres qui impactent fortement la performance de l'algorithme. Nous avons proposé un algorithme qui optimise les valeurs de ces deux hyperparamètres. Enfin, nous analysons notre algorithme d'optimisation par rapport à la littérature.

2 Approche proposée utilisant des techniques d'apprentissage automatique non supervisées

Notre approche générale consiste à détecter des tunnels DNS grâce aux méthodes de détection de valeurs aberrantes utilisant des algorithmes d'apprentissage automatique non supervisés. Nous avons aussi émis deux hypothèses qui ont guidé notre démarche :

Hypothèse 1 : *Le trafic malicieux représente un faible pourcentage du trafic authentique*. En effet, nous considérons des attaques perpétrées par des attaquants hautement qualifiés qui tentent de maintenir leur trafic malveillant sous les seuils des équipements traditionnels tels que les SIEM ou les pare-feux. Ainsi, la part de trafics correspondant aux tunnels DNS doit rester faible (par exemple, moins de 2 ou 3 % du trafic). En effet, dans le cas contraire, les systèmes de détection traditionnels tels que les SIEM pourront les détecter.

Hypothèse 2 : *la détection des valeurs aberrantes nécessite de comparer des éléments similaires*. En effet, comme nous souhaitons découvrir des valeurs aberrantes, il est nécessaire d'avoir une certaine homogénéité dans les flux de données authentiques. Contrairement aux approches généralistes de détection d'intrusion qui tentent de catégoriser tous les flux de données répertoriées dans le chapitre précédent, nous considérons que la détection d'attaques ciblées nécessite de séparer les flux par service afin de pouvoir détecter les utilisations frauduleuses des services réseau autorisés.

Par conséquence, les étapes de notre processus de détection sont les suivantes (Figure 22). Tout d'abord, nous séparons le trafic réseau par service (par exemple, DNS, HTTPS, …) dans les journaux d'événements à partir des numéros de ports des services réseau.

Dans une deuxième étape, nous préparons les données et les enrichissons avec des caractéristiques et statistiques calculées par rapport aux données des journaux. Pour cela, nous utilisons l'outils CICFlowMeter [96] qui est un générateur et un analyseur de flux de trafic réseau. Dans notre travail, l'outil n'est utilisé que pour extraire des caractéristiques du trafic réseau à partir des fichiers de capture brute au format PCAP. CICFlowMeter génère un jeu de données de flux réseau décrits par 38 caractéristiques qui sont :

- Les informations de base du protocole Internet (IP) : les adresses IP source et destination, les ports source et destination, et protocole encapsulé dans le paquet IP.
- Les informations de base des temps de paquets : l'horodatage, la durée du flux, le nombre d'octets du flux par seconde, le nombre de paquets du flux par seconde, etc.
- Le nombre total de paquets envoyés et reçus par sens.
- Les informations statistiques obtenues à partir des longueurs de paquets : le nombre d'octets moyen, la taille moyenne des paquets, l'écart type des tailles des paquets, etc.
- Le temps d'inter-arrivée des flux : le temps moyen entre deux paquets émis dans le flux, le temps total entre deux paquets émis, le temps d'écart type entre deux paquets émis, etc.

- Les statistiques des flags : le nombre de fois qu'un flag a été envoyé en paquets.
- Le temps de réponse : la durée moyenne d'activité d'un flux avant de devenir inactif, etc.

Ajouté à cela, au lieu d'utiliser la caractéristique « timestamps », nous avons extrait trois sous-caractéristiques à partir cette caractéristique : heure, minute, numéro de semaine (0-4 : jour de la semaine, 5-6 : week-end). Étant donné que tout le trafic passe par le résolveur DNS (pour les trafics DNS) ou le serveur DoH (pour les trafics DoH), l'adresse IP de destination que nous avons capturée est toujours celle du résolveur DNS ou le serveur DoH. De plus, dans la réalité, une machine infectée peut également envoyer les trafics authentiques. Ainsi, nous n'avons pas incluse les adresses IP dans notre jeu de données. Ceci donne finalement 39 caractéristiques au total (38 caractéristiques et label). La liste complète est indiquée dans le Tableau 16 en annexe.

Finalement, nous appliquons une ou plusieurs techniques d'apprentissage automatique non supervisé pour trouver les valeurs aberrantes qui seront les flux malicieux recherchés.

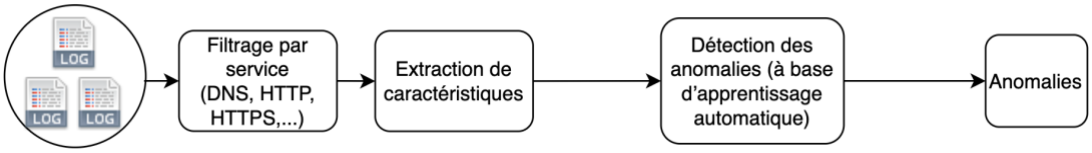


Figure 22. Phases de la méthode proposée

Comme nous l'avons montré dans le chapitre 2, plusieurs algorithmes d'apprentissage automatique non supervisés peuvent être considérés pour la détection de valeurs aberrantes. Pour effectuer ce choix, nous avons étudié dans [97] au début de ce doctorat les capacités de détection de quatre algorithmes (K-means, GMM, LOF, et DBSCAN) sur la base de données BOTS (Boss of the SOC) version 1 qui est mise à disposition par l'entreprise Splunk [6]. Cette base de données contient un scénario d'attaque comprenant une APT incluant la mise en œuvre d'un tunnel DNS pour une communication C&C. De plus, cette base de données contient deux parties : une incluant tout le trafic (malicieux et authentique) et une ne comprenant que l'attaque. Cela nous a permis de pouvoir évaluer les quatre algorithmes en terme de taux de détection, taux de faux positifs. Pour cela, nous avons généré sept sous-ensembles de données de 100 000 enregistrements contenant respectivement 0.1%, 0.5%, 1%, 2%, 3%, 4% et 5% d'attaques. Cette étude a mis en avant les capacités de l'algorithme DBSCAN par rapport aux autres. En particulier, nous avons pu isoler le trafic de C&C grâce après un post-traitement sur les anomalies trouvées par DBSCAN. Normalement, les hôtes infectés modifient leur comportement DNS, par exemple en

[6] https://github.com/splunk/botsv1

augmentant le volume de requêtes ou le volume d'octets sortants, ce qui peut identifier des signes de communication C&C ou de mouvement de données. Ainsi, nous avons classé les adresses IP du cluster anormal en fonction du nombre de requêtes et du nombre d'octets envoyés. Nous avons trouvé les adresses IP, qui ont le nombre le plus élevé de requêtes et le volume d'octets le plus élevé envoyé, correspondent exactement aux machines infectées dans l'ensemble de données. Cependant, cette étude a des limites car nous avions manuellement optimisé chaque hyperparamètre de chacun des algorithmes testés. Les valeurs optimales avaient été obtenues par expérimentations et n'étaient valides que pour un jeu de données particulier. Il n'est ainsi pas possible de garantir les mêmes performances sur un autre jeu de données.

DBSCAN ayant présenté les meilleurs taux de détection et taux de faux positifs, nous avons proposé une méthode pour déterminer automatiquement les valeurs optimales de ses deux hyperparamètres afin qu'il puisse être adapté aux différents jeux de données à analyser.

3 AutoRoC-DBSCAN

DBSCAN est un algorithme de clustering basé sur la densité qui est efficace pour trouver des régions à haute densité et des valeurs aberrantes. Il ne nécessite pas de spécifier le nombre de clusters et fonctionne bien avec des clusters de forme arbitraire. Cependant, ses performances dépendent de l'initialisation de ses deux hyperparamètres *minPts* et *eps*. Par conséquent, dans cette section, nous proposons une méthode pour trouver automatiquement les valeurs optimales pour ces deux paramètres.

Le premier paramètre, *minPts*, est le nombre de points de données requis pour former un cluster. Ce paramètre est en fait lié au taux d'attaque maximal attendu dans le réseau. Étant donné que le nombre d'attaques correspond également au nombre de valeurs aberrantes attendues dans DBSCAN, nous en déduisons que tous les clusters « normaux » devraient contenir plus de points que le nombre *minPts*. Dans le cadre de cette thèse, nous avons pris comme hypothèse que le pourcentage d'attaques maximal est 3% du trafic. Par conséquent, la valeur de *minPts* choisie est 3% du nombre de points de données dans l'ensemble de données. Cette valeur peut être raffinée selon l'analyse de risque effectuée pour une organisation donnée.

Le deuxième paramètre à déterminer *eps* est la distance maximale entre 2 points dans le même cluster. Contrairement à *minPts*, il n'est pas possible de déterminer directement cette valeur par rapport à des hypothèses de sécurité. Nous proposons de résoudre ce problème à l'aide de l'algorithme des K-plus proches voisins (KNN) [98]. L'idée de base contient deux étapes : d'abord, calculer la distance moyenne entre chaque point et ses K plus proches voisins (étape 1), puis calculer le taux de changement sur ces distances (étape 2) pour trouver la valeur optimale de *eps*. Le point

correspondant à un changement soudain et significatif des valeurs de distance est alors utilisé pour la valeur de *eps*.

Il y a 2 paramètres qui doivent être prédéterminés dans notre algorithme :

- K : est le nombre de voisins dans l'algorithme KNN utilisé à l'étape 1. Puisque le nombre minimum de points requis pour former un cluster est *minPts*, on peut facilement voir que le nombre minimum de voisins de chaque point du cluster est *minPts-1*. Pour un grand ensemble de données, il peut être considéré comme identique à *minPts*. Par conséquent, la valeur de K peut être identique à *minPts*, soit 3% de l'ensemble de données dans notre cas.
- Delta : est le nombre de points dans chaque plage de valeurs qui sera utilisé pour calculer le taux de changement à l'étape 2. Une valeur trop faible de Delta réduira la capacité à détecter les points de changement puisque la valeur des points proches de l'autre ne change pas beaucoup. Inversement, si la valeur de Delta est trop élevée, nous pourrions manquer le point de changement optimal s'il se situe à l'intérieur de la dernière large plage de valeurs. Une bonne valeur d'équilibre pour Delta est d'environ 10% de *minPts*, soit 0.3% de l'ensemble de données. Cette valeur a été obtenue expérimentalement sur un jeu de données spécifique différent de celui que nous avons utilisé pour l'évaluation des performances.

Notre algorithme pour trouver la meilleure valeur d'*eps* se compose en 7 étapes, divisées en 3 parties principales :

Partie 1 : Calculer la courbe des distances basée sur l'algorithme KNN (Figure 23).

1. Pour chaque point x du jeu de données, exécutez l'algorithme KNN avec K = 3% du jeu de données, puis calculez la distance moyenne entre x et ses K voisins les plus proches.
$$mean_x^{KNN} = mean(KNN(x, K = 3\%))$$
2. Triez l'ensemble de données par rapport à $mean_x^{KNN}$ pour produire *sortedDistanceList.*

Algorithm 1: AutoRoc-DBSCAN

Part 1: Calculate the curve of the distances based on KNN algorithm;
Data: n: number of data in dataset X;
K: number of neighbors;
$K = 0,03 * n$;
$KNNdistances = \emptyset$;
for x *in* X **do**
  $KNNdistances = KNNdistances \cup (x, calculateKNN(x, X$ with number of neighbors = K)) ; /* Step 1 */
end
$meanDistanceList = \emptyset$; /* Step 2 */
for *(x, distances) in KNNdistances* **do**
  $meanDistanceList = meanDistanceList \cup (calculateMean(distances))$;
end
$sortedDistanceList = sort(meanDistanceList)$;

Figure 23. AutoRoC-DBSCAN partie 1

Partie 2 : Trouver les points avec de grands changements de valeur (Figure 24).

3. Calculer la pente d'un intervalle de données. Nous ne calculons pas la pente entre les points adjacents car la différence entre eux sera trop petite pour détecter un changement important. Ainsi, nous définissons l'intervalle de données comme Delta = 0.3% de l'ensemble de données. Pour chaque point *y* dans *sortedDistanceList*, calculer la pente entre ce point et un point qui en est éloigné et ajoutez-le à *slopeList*.
   $slope_y = sortedDistanceList_{y+\Delta} - sortedDistanceList_y$
   $slopeList = \{slope_y\}$
4. Calculer les pentes moyennes par bloc de taille Delta. Pour l'évolutivité, nous devons réduire le nombre de points à considérer. Ainsi, nous regroupons les pentes par bloc de taille Delta et calculons la moyenne de chaque groupe et les ajoutons à la *meanSlopeList*.
   $meanSlope_a = \frac{1}{\Delta}\sum_{i=a}^{a+\Delta} slopeList_i$
   $meanSlopeList = \{meanSlope_a\}$
5. Calculer le pourcentage de différence entre la moyenne d'un bloc et la moyenne du bloc précédent et ajoutez-le à *diffPercentageList*.
   $diffPercentage_a = \frac{(meanSlopeList_a - meanSlopeList_{a-1}) * 100}{meanSlopeList_{a-1}}$
   $diffPercentageList = \{diffPercentage_a\}$
6. Calculer la différence entre le pourcentage de changement entre deux blocs adjacents et la moyenne de tous les pourcentages de changements précédents. Si cette différence est supérieure à 3 fois, il y a un grand changement de valeur à ce bloc. Nous ajoutons le premier point de ce bloc à *largeChangeValuePointDict* avec la clé étant le point de données correspondant.

$$avgDiffPercentageList_a = \frac{1}{(a-1)} \sum_{i=1}^{a-1} diffPercentageList_i \; ; a \geq 2$$

$$difference_a = \frac{diffPercentage_a}{avgDiffPercentageList_a}$$

$$pointNumber = a * Delta$$

If $difference_a \geq 3$ ***Then*:**

$$largeChangeValuePointDict[pointNumber] = difference_a$$

Algorithm 1: AutoRoc-DBSCAN

Part 2: Find the points with large changes in value;
Data: n: number of data in dataset X;
$Delta$: number of points in each range of values in the rate of change;
$Delta = minPts * 0.1 * n = 0.003 * n$;
$slopeList = \emptyset$;
for *index* $i = 0$ *to (length(sortedDistanceList) - delta), delta* **do**
  $delta_i = i + delta$; `/* Step 3 */`
  $slopeList = slopeList \cup (sortedDistanceList[delta_i] - sortedDistanceList[i])$
end
$meanSlopeList = \emptyset$;
for *index* $j = 0$ *to length(slopeList-delta), delta* **do**
  $delta_j = j + delta$; `/* Step 4 */`
  $meanSlopeList =$
  $meanSlopeList \cup (calculateMean(slopeList[j], ..., slopeList[j + delta]))$;
end
$diffPercentageList = \emptyset$;
$avgDiffPercentageList = \emptyset$;
$largeChangeValuePointDict = \emptyset$; `/* Step 5 and 6 */`
for *index* $k = 1$ *to length(meanSlopeList), 1* **do**
  $avgDiffPercentageList_k = calculateMean(diffPercentageList)$;
  $differentValue_k = meanSlopeList[k]–meanSlopeList[k-1]$;
  $differentPercentage_k = absolute(differentValue_k * 100/meanSlopeList[k-1])$;
  $diffPercentageList = diffPercentageList \cup differentPercentage_k$;
  $difference_k = differentPercentage_k/avgDiffPercentageList_k$;
  if $difference_k >= 3$ **then**
    $largeChangeValuePointDict =$
    $largeChangeValuePointDict \cup (k * Delta, difference_k)$;
  end
end

Figure 24. AutoRoC-DBSCAN Partie 2

Partie 3 : Trouver le premier point de changement de valeur maximale parmi les points répertoriés dans largeChangeValuePointDict (Figure 25).

7. Appliquer la méthode *find_peaks* [99] pour trouver une liste de pics et choisir la valeur du premier pic comme valeur de *eps* pour DBSCAN. *find_peaks* est une méthode de bibliothèque *scipy* pour rechercher des pics (maxima locaux)

basée sur une simple comparaison de valeurs d'échantillons voisins et renvoie ces pics. Si le *largeChangeValuePointDict* est une liste ascendante, nous ne pouvons pas trouver de pic. Dans ce cas, nous calculons la différence entre chaque paire de valeurs adjacentes. Ensuite, nous choisissons le point qui a la plus grande différence entre deux points adjacents.

Algorithm 1: AutoRoc-DBSCAN

```
Part 3: Find the first maximum value change point among the listed points in Part 2;
peakList = ∅;
pointListValue = largeChangeValuePointDict.values()
 peaksList = findPeaks(pointListValue) ;                    /* Step 7 */
if peaksList ≠ ∅ then
    pointValue = pointListValue[peaksList[0]];
    for key in largeChangeValuePointDict do
        if largeChangeValuePointDict[key] = pointValue then
            point = key ;
            epsilon = sortedDistanceList[point];
        end
    end
end
if peaksList = ∅ then
    dif = 0;
    for index i = 0 to length(pointListValue)-1 do
        j = i + 1;
        difNew = pointListValue[j]/pointListValue[i];
        if difNew > dif then
            dif = difNew;
            point = largeChangeValuePointDict.keys()[j];
            epsilon = sortedDistanceList[point];
        end
    end
end
```

Figure 25. AutoRoC-DBSCAN Partie 3

4 Méthode de détermination automatique des paramètres de DBSCAN

Starczewski et al. [100] ont proposé une nouvelle méthode qui détermine les paramètres de DBSCAN pour différents types de clusters : spirales, yeux, carré, triangle, formes d'onde, etc. Cette méthode utilise une fonction kdist pour calculer la distance entre chaque élément d'un ensemble de données et son k-th voisin le plus proche. Ensuite, ils ont trié les distances et déterminé la taille du genou en fonction du nombre de distances générées par la fonction kdist et de la taille du jeu de données. Le paramètre MinPts est calculé par la taille de ce genou. Ils ont défini le point qui correspond aux fortes augmentations des distances. Sur la base de ce point et de la

taille du genou, la valeur correcte du paramètre Eps est calculée. Ils ont évalué leur méthode sur plusieurs ensembles de données en 2 et 3 dimensions avec la différence de nombre de clusters, de tailles et de formes. Tous les résultats confirment une très grande efficacité de l'approche nouvellement proposée.

Falahiazar et al. [101] ont introduit un algorithme hybride qui utilise l'algorithme génétique multi-objectif (Multiobjective Genetic Algorithm - MOGA) pour déterminer automatiquement les paramètres de l'algorithme DBSCAN. Ils ont utilisé l'algorithme de triangulation de Delaunay pour déterminer les bornes initiales des paramètres de DBSCAN. Ensuite, ils ont utilisé les indices internes (indices Silhouette, Dunn) comme fonctions objectifs pour choisir un ensemble de valeurs pour les paramètres Eps et MinPts. Un nouvel index interne nommé Outlier-index est proposé pour apporter plus de diversité et de qualité dans les solutions produites. Ils ont comparé les résultats de MOGA-DBSCAN avec les meilleurs résultats obtenus à partir de DBSCAN et ceux de SOGA-DBSCAN sur cinq jeux de données (aggregation, spiral, flame, compound, path-based) par six indices externes qui sont Jaccard, Rand, Minkowski, Précision, Rappel et F1. Les résultats de MOGA-DBSCAN, qui détermine automatiquement les paramètres, sont proches des meilleurs résultats du clustering DBSCAN.

Karami and Johansson [102] ont présenté une méthode de clustering hybride, nommée BDE-DBSCAN, qui combine Binary Differential Evolution (BDE) et l'algorithme DBSCAN pour spécifier simultanément rapidement et automatiquement les valeurs de paramètre appropriées pour Eps et MinPts. Cette méthode a exécuté plusieurs fois l'algorithme Differential Evolution pour trouver un ensemble de valeurs pour les paramètres Eps et MinPts. Ensuite, la méthode de Tournament Selection est également utilisée pour trouver la meilleure valeur de Eps et de MinPts. L'algorithme proposé est évalué sur neuf ensembles de données artificielles 2D comme Cluster-inside-Cluster (K=3 et 5), Corners, Crescent Full Moon, Half Kernel, Pinwheel, Semi Circular (K=4 and 8), Outlier et Aggregation, Compound et Pathbased. Les résultats de l'algorithme proposé fournissent une précision optimale par la pureté comprise entre 99.4% et 100%.

DBSCAN est très populaire et le problème du choix automatique des paramètres d'entrée de l'algorithme DBSCAN a été un grand défi. Les chercheurs ont proposé des méthodes pour trouver automatiquement ces hyper paramètres pour de petits ensembles de données et quelques dimensions seulement. Par exemple, dans le jeu de données en spirale, il n'y a que 3 clusters et 2 dimensions. Ce n'est pas évolutif et donc valable pour détecter le trafic DNS anormal. Notre méthode vise à identifier les valeurs aberrantes dans de grands ensembles de données à 40 dimensions.

5 Conclusion

Dans ce chapitre, nous avons présenté une méthode pour détecter des valeurs aberrantes basée sur l'algorithme DBSCAN. Cependant, cela nécessite de creuser profondément dans l'ensemble de données et d'effectuer de nombreuses expériences intéressantes pour déterminer les 2 paramètres. Par conséquence, nous avons proposé une méthode pour déterminer automatiquement ces 2 paramètres. Nous allons montrer les résultats dans le chapitre 5.

Chapitre 5. Évaluation des performances

1 Introduction

Les algorithmes d'apprentissage automatique ont démontré leur potentiel à détecter des attaques dans le réseau (voir chapitre 3), en particulier les communications C&C encapsulées dans DNS. Cependant, les algorithmes d'apprentissage automatique ont des paramètres qu'il faut définir à l'avance. Dans le chapitre précédent, nous avons proposé une approche utilisant l'algorithme DBSCAN pour détecter des tunnels DNS incluant une méthode pour trouver automatiquement les valeurs optimales des hyperparamètres de DBSCAN.

Bien que les algorithmes d'apprentissage automatique non supervisées n'aient pas besoin d'un jeu de données pour la phase d'apprentissage, un jeu de données étiqueté est nécessaire afin d'évaluer leur performance. De plus, un jeu de données étiqueté peut donner plus d'assurance sur l'évaluation puisque le résultat final est connu. C'est pourquoi nous avons besoin d'un jeu de données contenant des trafics DNS normaux (respectivement des trafics DoH normaux) et des tunnels DNS (respectivement des tunnels DoH) pour l'étape d'évaluation.

Comme nous avons montré dans le chapitre précédent, nous avons étudié dans [97] les capacités de détection de quatre algorithmes (K-means, GMM, LOF, et DBSCAN) sur la base de données BOTS version 1. Bien qu'elle contienne un scénario d'attaque APT qui utilise un tunnel DNS pour la communication C&C, cette base de données a des limites. Tout d'abord, elle ne contient pas les données brutes initiales au format PCAP par exemple. De plus, le processus de calcul des caractéristiques (e.g., statistiques) n'est pas décrit. L'architecture et les outils utilisés pour générer le trafic font également défaut. Enfin, le nombre de caractéristiques est limité. Par conséquent, il n'est pas possible de reproduire ou d'améliorer ce jeu de données pour comparer les performances des algorithmes de détection. Nous avons donc décidé de créer notre propre jeu de données qui contient des tunnels DNS car il n'existe pas d'autre jeu de données pour tester la détection de tunnels DNS. Par contre, nous pourrons réutiliser les jeux de données CIRA-CIC-DoHBrw-2020 du projet de l'Institut Canadien pour la Cybersécurité (CIC) qui contient des trafics HTTPS normaux, des trafics DoH normaux, et des trafics DoH malveillants. Ce jeu de données contient les données brutes et son processus de création est transparent. Ainsi dans cette thèse, nous utilisons deux jeux de données pour évaluer notre méthode : notre jeu de données pour la détection de trafics DNS malveillants ; le jeu de données CIRA-CIC-DoHBrw-2020 pour la détection de trafics DoH malveillants.

Dans notre approche, nous désirons détecter des tunnels DNS (respectivement des tunnels DoH) qui sont utilisés souvent par des attaques complexes comme des APTs. Ce genre d'attaque est contrôlé pour rester sous les seuils de détection. Ainsi, les attaquants tenteront de garder à un niveau très faible le pourcentage de ce type d'attaque dans le réseau. Nous le considérons donc comme étant des anomalies par rapport au trafic réseau authentique. Pour détecter ce type d'attaque, nous avons étudié dans le chapitre 2 quelques algorithmes qui permettent de détecter des anomalies. Auto-Encodeur et SOM sont deux algorithmes d'apprentissage en profondeur. Auto-Encodeur est bien dans le cas de reconstruction des images et SOM peut aider à visualiser des données de grande dimension pour comprendre les données. Ils fonctionnent bien pour la réduction de dimension mais ils ne sont pas adaptés pour détecter des anomalies avec un grand nombre de caractéristiques. De plus, leurs temps d'exécution sont extrêmement lents.

Nous comparons donc dans ce chapitre les performances de notre algorithme basé sur DBSCAN avec 5 autres algorithmes : K-means, GMM, Isolation Forest, One-class SVM et LOF (voir chapitre 2) en utilisant 72 sous-jeux de données extraits de notre jeu de données de tunnel DNS et de CIRA-CIC-DoHBrw-2020. Nous avons configuré K-means et GMM pour classer les jeux de données en deux clusters (K = 2) afin de séparer les trafics authentiques et anormaux. Les 4 algorithmes restants ont été réglés en fonction du pourcentage d'attaque maximal (3%), ce qui signifie que nous avons défini le nombre d'estimateurs *n_estimators* de Isolation Forest, le *nu* lié à One-class SVM, le *minPts* de DBSCAN et le nombre de voisins *MinPts* de LOF sont égaux au nombre maximum d'attaques. Enfin, La valeur de *eps* est réglée par notre algorithme AutoRoC-DBSCAN (voir chapitre 4).

Nous utilisons le taux de détection (DR – Detection Rate) et le taux de faux positifs (FPR – False Positive Rate) pour évaluer les performances des algorithmes. Le taux de détection est le nombre d'attaques détectés par le système divisé par le nombre d'attaques dans le jeu de données. Le taux de faux positifs est le nombre de connexions authentiques classées à tort comme des attaques divisé par le nombre de connexions authentiques dans le jeu de données. Par conséquent, un bon algorithme doit atteindre une valeur de DR élevée tout en maintenant le FPR bas. Comme les attaques complexes sont difficiles à détecter, nous ciblons un DR supérieur à 80 % et un FPR inférieur à 2 %. Ainsi, nous faisons le choix de favoriser la pertinence des alertes par rapport à la capacité de détection.

2 Détection des tunnels DNS

2.1 Description du jeu de données

Comme il n'existe pas de jeu de données pour tester des algorithmes de détection des tunnels DNS, ainsi nous avons créé un jeu de données qui contient à la fois des trafics

DNS authentiques et des trafics tunnels DNS. Pour cela, nous avons créé une infrastructure de machines virtuelles qui contient tous les composants requis pour effectuer une connexion de tunnel DNS. La topologie du réseau utilisée pour capturer le trafic est présentée dans le Figure 26.

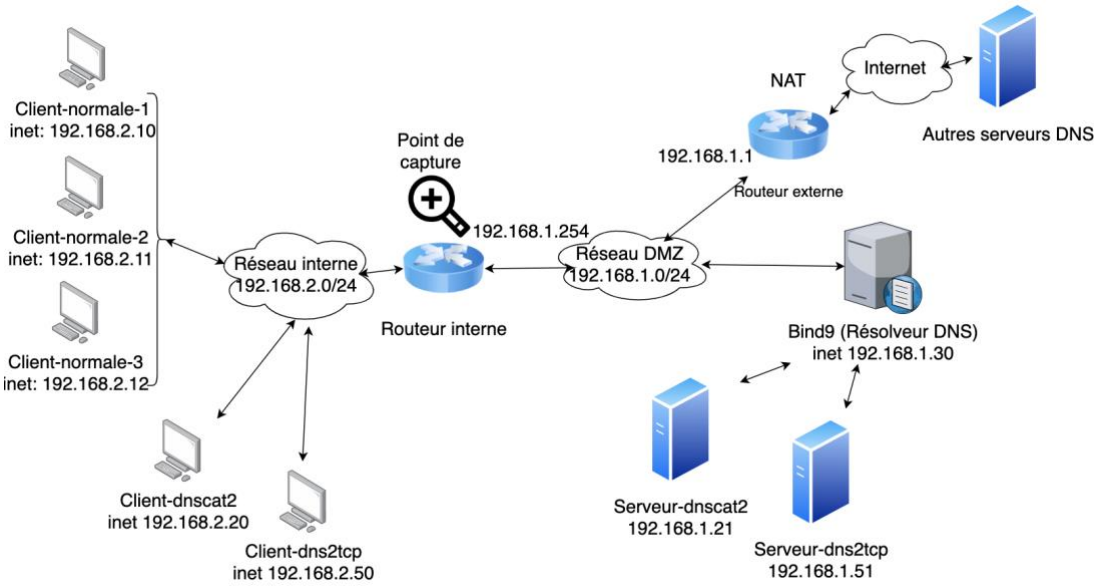


Figure 26. Topologie du réseau utilisé pour capturer le trafic

Tout d'abord, le trafic authentique est généré par une activité de navigation Web normale. Nous utilisons Chrome et Firefox pour accéder de manière aléatoire à une liste de 100 sites Web différents. Ensuite, le trafic DNS malveillant est généré par une combinaison d'outils utilisés pour créer des tunnels DNS, tels que Dnscat2 et Dns2tcp. La quantité de chaque type de trafic est disponible dans le Tableau 4. Ce jeu de données est disponible sur Github[7].

Nous avons mis en place un réseau local interne qui contient tous les postes clients (machines normales et machines compromises) et un réseau DMZ qui contient un serveur résolveur DNS Bind9 qui relaye les requêtes soit vers Internet, soit vers les serveurs C&C selon le nom de domaine demandé. En effet, dans le cas des attaques par tunnels DNS, un attaquant doit enregistrer un nom domaine réel et son serveur C&C qui est vu comme un serveur DNS. Dans notre cas, tous les trafics sont générés dans l'infrastructure de la machine virtuelle ; ainsi, nous avons configuré le résolveur DNS pour transmettre les requêtes DNS concernant les domaines des attaquants aux serveurs C&C associés (c'est-à-dire, serveur Dnscat2, serveur Dns2tcp). Les requêtes vers d'autres noms de domaine sont redirigées vers Internet.

[7] https://github.com/quynnhnguyen/DNS-dataset/tree/main

Pour plus de simplicité, les serveurs Dnscat2 et Dns2tcp C&C sont également installés dans la DMZ. Cela n'affecte pas la qualité du jeu de données, car le trafic réseau est capturé au niveau du routeur interne et le nom de domaine n'est pas utilisé dans les caractéristiques de détection.

Cette approche permet à quiconque de créer facilement son propre jeu de données en manipulant les fichiers de données brutes en fonction de ses propres objectifs.

2.2 Préparation

Nous utilisons CICFlowMeter [96] pour préparer notre jeu de données. CICFlowMeter nous permet de regrouper les paquets IP en « flux » selon les adresses IP source et destination, les ports source et destination et de calculer des statistiques qui seront utilisées dans l'analyse. Au vu de ces flux, on pourrait penser qu'il est facile d'identifier les tunnels DNS en observant le nombre d'octets échangés. Dans le Tableau 3, le nombre d'octets du flux DNS normal varie de 70 à 790, tandis que le nombre d'octets du trafic d'attaque Dnscat2 est énorme, allant de 241 à 126479. La différence entre les flux DNS normaux et ceux de Dns2tcp n'est pas aussi clair que pour Dnscat2, cependant le nombre maximum d'octets semble toujours un critère différenciant.

Cependant, ce résultat s'explique par le fait que les clients Dnscat2 et Dns2tcp utilisent le même port source pour différentes requêtes DNS. Ainsi, CICFlowMeter regroupe les différentes requêtes/réponses dans le même flux. Ce comportement peut être facilement changé par un attaquant en modifiant le code des clients Dnscat2 et Dns2tcp pour utiliser un port source différent pour chaque requête DNS, rendant inutile une détection de tunnel DNS uniquement basée sur le nombre d'octets des flux calculés par CICFlowMeter.

Tableau 3. Nombre d'octets échangés des trafics normaux, Dnscat2 et Dns2tcp

	Original CICFlowMeter		Modified CICFlowMeter	
	Nombre d'enregistrements	Nombre d'octets échangés	Nombre d'enregistrements	Nombre d'octets échangés
Chrome (normale)	124158	Min : 81 Max : 663	124158	Min : 81 Max : 663
Firefox (normale)	223858	Min : 70 Max : 790	223858	Min : 70 Max : 790
Dnscat2 (attaque)	1967	Min : 241 Max :126479	353604	Min : 108 Max : 1830
Dns2tcp (attaque)	406352	Min : 94 Max : 2941	445097	Min : 94 Max : 618

Tableau 4. Quantité de chaque type de trafic – notre base de données

	Chrome	Firefox	Dnscat2	Dns2tcp
Nombre d'enregistrements	124158	223858	353604	445097

Nous avons donc modifié CICFlowMeter pour construire des flux en regroupant les paquets selon le champ DNS « TransactionID ». Les attaquants ne peuvent pas usurper cette valeur car il est simple d'implémenter une règle sur le pare-feu qui supprimerait les paquets avec un « TransactionID » incompatible. En analysant le résultat après modification de CICFlowMeter dans le Tableau 3, nous voyons qu'il n'y a pas de différence significative entre une requête/réponse DNS normale et une requête/réponse de tunnel DNS en termes de nombre d'octets. Cela confirme la nécessité de mettre en œuvre une analyse comportementale du réseau avec des algorithmes d'apprentissage automatique afin de combiner de nombreux critères.

En utilisant l'outils CICFlowMeter modifié, nous avons créé un jeu de données qui contient 39 caractéristiques (voir le Tableau 16 en annexe). Afin d'évaluer comment les algorithmes d'apprentissage automatique peuvent détecter un trafic DNS anormal avec différents pourcentages d'attaque, nous avons généré 12 sous-ensembles de données de 100 000 enregistrements chacun, qui contiennent respectivement 0.1 %, 0.5 %, 1 %, 2 %, 3 %, 4 %, 5%, 6%, 7%, 8%, 9%, 10% des attaques. Dans chaque sous-ensemble de données, nous avons défini les pourcentages des deux types d'attaque (Dnscat2 et Dns2tcp) pour qu'ils soient presque égaux. Par exemple, 2 % des attaques dans le sous-ensemble de données proviennent à parts égales de Dnscat2 (1%) et de Dns2tcp (1%).

2.3 Évaluation de performance

La Figure 27 et la Figure 28 montrent les DR et les FPR des six algorithmes pour les 12 sous-ensembles de données de notre jeu de données. Les résultats détaillés sont dans le Tableau 5.

Les algorithmes donnent de meilleurs résultats lorsque le taux d'attaque est faible. Lorsque le taux d'attaque augmente, leurs performances diminuent rapidement car les attaques ne sont plus aberrantes.

En supposant que les attaques sont contrôlées à un faible taux afin de rester sous les seuils classiques de détection, lors de la comparaison des performances des algorithmes, nous ne considérerons que les résultats des sous-ensembles de données avec des taux d'attaque allant jusqu'à 3%.

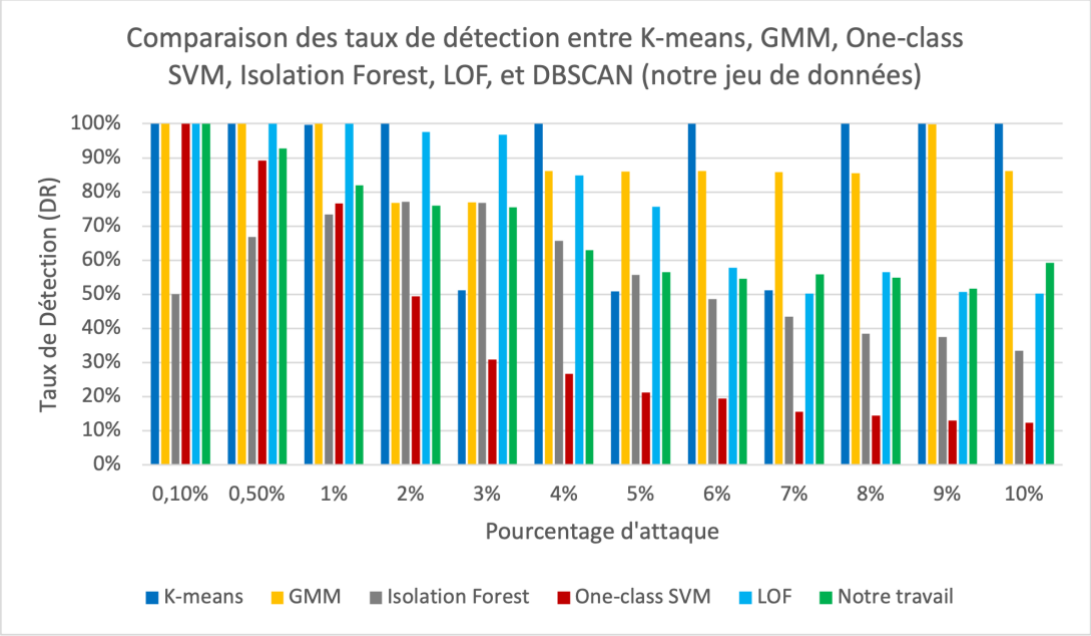


Figure 27. Comparaison des taux de détection dans notre jeu de données

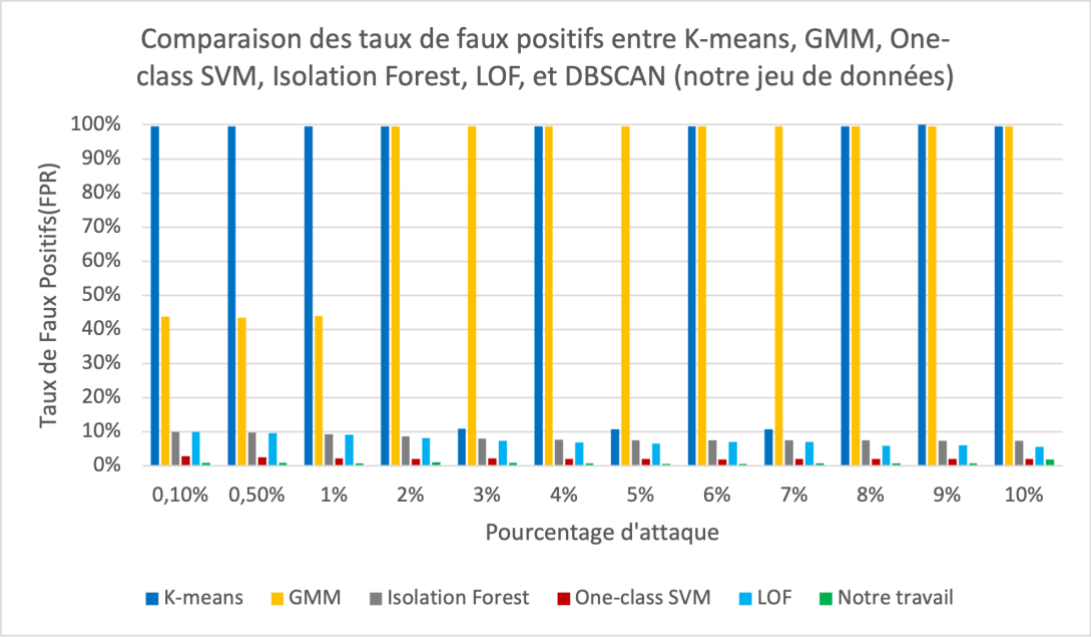


Figure 28. Comparaison des taux de faux positifs dans notre jeu de données

Tableau 5. Résultats de notre jeu de données

DIACRE-mix	K-means		GMM		Isolation Forest		One-class SVM		LOF		Notre travail	
	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)
0.1%	100	99.56	100	43.7	50	9.95	100	2.9	100	9.9	**100**	**0.96**
0.5%	100	99.52	100	43.49	66.8	9.71	89.2	2.5	100	9.54	**92.8**	**0.96**
1%	99.6	99.56	100	43.94	73.4	9.35	76.7	2.2	100	9.09	**81.9**	**0.8**
2%	100	99.57	76.75	99.56	77.05	8.63	49.35	2.05	97.55	8.21	**76.05**	**1**
3%	51.1	10.94	76.96	99.52	76.83	7.93	30.9	2.13	96.7	7.31	**75.53**	**0.95**
4%	99.95	99.55	86.07	99.53	65.67	7.68	26.72	2.01	84.91	6.87	**62.92**	**0.69**
5%	50.86	10.69	85.9	99.52	55.62	7.59	21.2	2.04	75.6	6.54	**56.56**	**0.6**
6%	100	99.57	86.1	99.54	48.63	7.53	19.43	1.95	57.78	6.95	**54.56**	**0.6**
7%	51.12	10.74	85.81	99.51	43.51	7.47	15.55	2.05	50.28	6.96	**55.84**	**0.67**
8%	99.97	99.54	85.46	99.5	38.42	7.52	14.42	2	56.52	5.95	**54.92**	**0.73**
9%	99.98	100	99.87	99.54	37.41	7.28	12.97	2.01	50.62	5.98	**51.58**	**0.74**
10%	99.93	99.57	86.15	99.43	33.38	7.4	12.29	1.96	50.2	5.53	**59.28**	**1.83**

K-means et GMM donnent de mauvais résultats. Les DR de K-means sont élevés, plus de 99% mais ses FPR sont aussi élevés, plus de 99%. Dans certains cas, si son FPR est faible de 10.94%, son DR est également faible de 51.1%. Similaire à K-means, les DR de GMM sont bons de 76.96% à 100% mais ses FPR sont élevés, plus de 43%.

Isolation Forest ont des meilleurs FPR que K-means et GMM mais reste élevé, de 7.93% à 9.95%, et ses DR sont de 50% à 77.05%.

LOF détecte bien avec DR de 96.7% à 100%, mais ses FPR sont très importants, de 7.31% à 9.9%. One-class SVM détecte bien lorsque le taux d'attaque est inférieur à 0.5% avec DR plus de 89.2% et FPR de 2.5% à 2.9%. Cependant, ses DR chutent très rapidement quand le taux d'attaque dépasse cette valeur. Notre algorithme a de bons résultats pour des valeurs de taux d'attaques plus grandes. Son DR est supérieur à 81.9% pour des taux d'attaques jusqu'à 1% et reste bon jusqu'à 3% d'attaque avec un DR supérieur à 75.53%. Il faut aussi noter que notre algorithme a d'excellents résultats pour le FPR avec des valeurs inférieures à 1% et ceux quelque soit la valeur de taux d'attaque.

Lorsque nous séparons le trafic normal de Chrome et Firefox, les résultats obtenus à partir de tous les algorithmes sont meilleurs, comme le montrent le Tableau 6 et les figures : Figure 29 – Figure 32. Les résultats détaillés sont dans les tableaux : Tableau 7 et Tableau 8.

Tableau 6. Comparaison des résultats de notre jeu de données avec des taux d'attaque allant jusqu'à 3%

	K-means		**GMM**		**Isolation Forest**		**One-class SVM**		**LOF**		**Notre travail**	
	DR (%)	FPR (%)	DR (%)	FPR (%)	DR (%)	FPR (%)	DR (%)	FPR (%)	DR (%)	FPR (%)	DR (%)	FPR (%)
Notre jeu de données												
0.1%	100	99.56	100	43.7	50	9.95	100	2.9	100	9.9	**100**	**0.96**
0.5%	100	99.52	100	43.49	66.8	9.71	89.2	2.5	100	9.54	**92.8**	**0.96**
1%	99.6	99.56	100	43.94	73.4	9.35	76.7	2.2	100	9.09	**81.9**	**0.8**
2%	100	99.57	76.75	99.56	77.05	8.63	49.35	2.05	97.55	8.21	**76.05**	**1**
3%	51.1	10.94	76.96	99.52	76.83	7.93	30.9	2.13	96.7	7.31	**75.53**	**0.95**
Notre jeu de données - Chrome												
0.1%	100	99.57	100	51.64	51	9.95	100	2.9	100	9.9	**100**	**1.09**
0.5%	100	99.56	100	51.6	71.6	9.69	89.8	2.55	100	9.54	**98.8**	**2.08**
1%	99.9	99.57	100	54.8	79.5	9.29	73.4	2.28	98.9	9.1	**96.4**	**2.57**
2%	100	99.56	100	51.58	81.65	8.53	45.8	2.12	96.65	8.23	**87.65**	**2.84**
3%	99.96	99.59	99.96	51.44	76.56	7.94	30.86	2.14	91.83	7.46	**85.63**	**3.09**
Notre jeu de données - Firefox												
0.1%	53	11.94	100	99.51	53	9.95	100	2.9	100	9.9	**100**	**0.99**
0.5%	99.8	99.51	99.8	99.51	62.2	9.73	100	2.56	100	9.54	**100**	**0.89**
1%	100	99.49	100	99.47	63.5	9.45	72.3	2.29	100	9.09	**95.6**	**0.89**
2%	99.9	99.48	99.95	99.46	66.35	8.85	44.15	2.15	100	8.16	**84.35**	**0.81**
3%	99.9	99.53	85.7	99.51	63.86	8.33	31.86	2.1	98.76	7.25	**80.56**	**1.08**

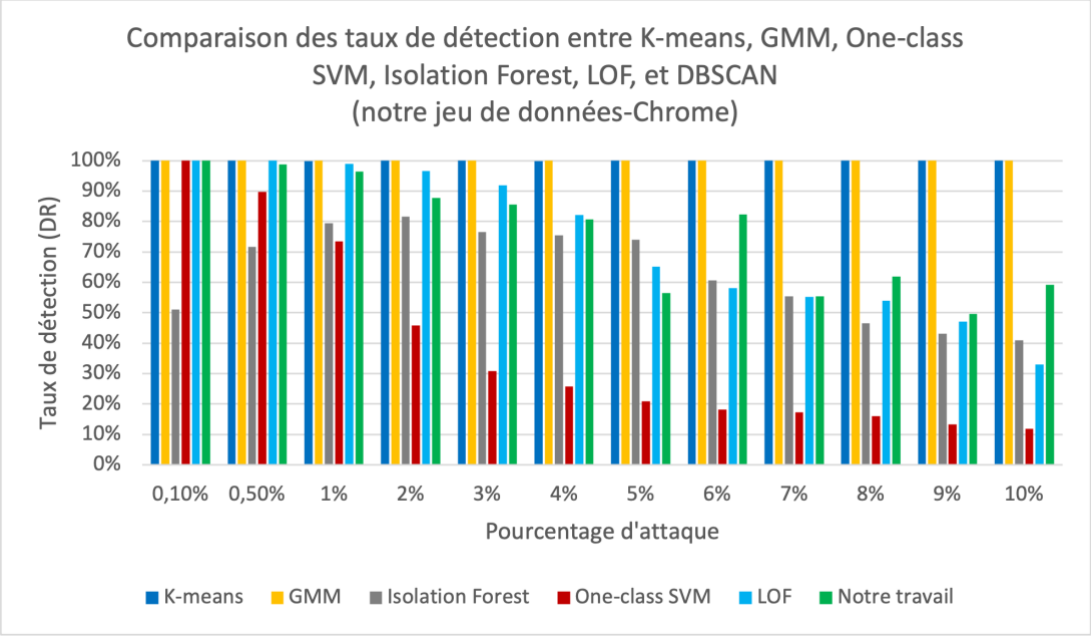


Figure 29. Comparaison des taux de détection dans notre jeu de données – Chrome

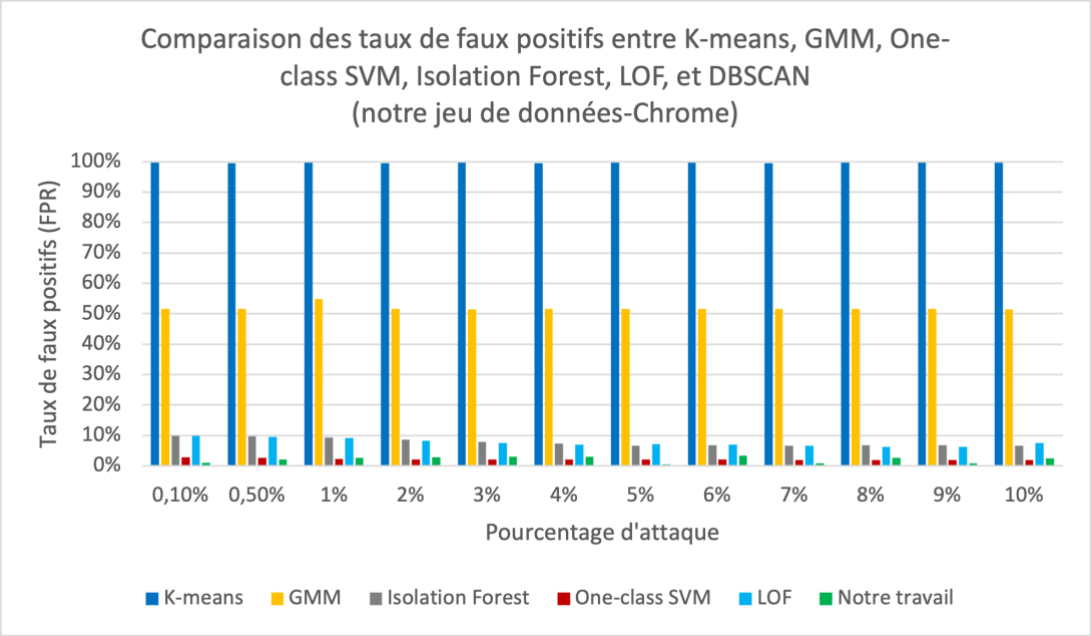


Figure 30. Comparaison des taux de faux positifs dans notre jeu de données – Chrome

Dans les jeux de données avec Chrome (Figure 29, Figure 30, et Tableau 7), K-means et GMM donnent toujours de mauvais résultats. Les DR de K-means sont élevés de 99.96% à 100% et ses FPR sont plus de 99.56%. Les DR de GMM sont plus de 99.96% et ses FPR sont élevés, plus de 51.44%. Isolation Forest a toujours des DR faibles de 51% à 81.65% et des FPR élevés de 7.94 % à 9.95 %.

LOF donne des bons DR jusqu'à 3% d'attaque (plus de 91.83%) voire même jusqu'à 4% d'attaque (DR à 82.2%) mais ses FPR restent élevés de 7.46% à 9.9%. Quand le taux d'attaque à 0.5%, One-class SVM a de bons DR, plus de 89.8% et des FPR faibles de 2.55% à 2.9%. Cependant les performances chutent très rapidement dès que le taux d'attaque dépasse les 0.5%. Notre travail a de très bons DR de 85.63% à 100% et ses FPR sont inférieurs à 3.09%. On peut voir globalement que notre travail est un peu moins bon que LOF sur le DR mais il est meilleur sur le FPR.

Tableau 7. Résultats de notre jeu de données – Chrome

DIACRE-chrome	K-means		GMM		Isolation Forest		One-class SVM		LOF		Notre travail	
	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)
0.1%	100	99.57	100	51.64	51	9.95	100	2.9	100	9.9	**100**	**1.09**
0.5%	100	99.56	100	51.6	71.6	9.69	89.8	2.55	100	9.54	**98.8**	**2.08**
1%	99.9	99.57	100	54.8	79.5	9.29	73.4	2.28	98.9	9.1	**96.4**	**2.57**
2%	100	99.56	100	51.58	81.65	8.53	45.8	2.12	96.65	8.23	**87.65**	**2.84**
3%	99.96	99.59	99.96	51.44	76.56	7.94	30.86	2.14	91.83	7.46	**85.63**	**3.09**
4%	99.9	99.55	100	51.62	75.37	7.27	25.7	2.05	82.2	6.99	**80.65**	**3.06**
5%	99.94	99.59	100	51.56	74.04	6.62	20.82	2.06	65.22	7.09	**56.46**	**0.56**
6%	99.98	99.59	99.98	51.64	60.6	6.77	18.11	2.03	58.16	6.92	**82.33**	**3.35**
7%	100	99.55	100	51.61	55.31	6.58	17.21	1.92	55.28	6.59	**55.31**	**0.8**
8%	99.96	99.57	100	51.63	46.46	6.82	15.93	1.87	54.03	6.17	**61.85**	**2.65**
9%	99.95	99.58	100	51.67	43.12	6.72	13.27	1.98	47.11	6.32	**49.6**	**0.77**
10%	99.93	99.58	99.99	51.49	40.88	6.56	11.94	2	33.06	7.43	**59.19**	**2.53**

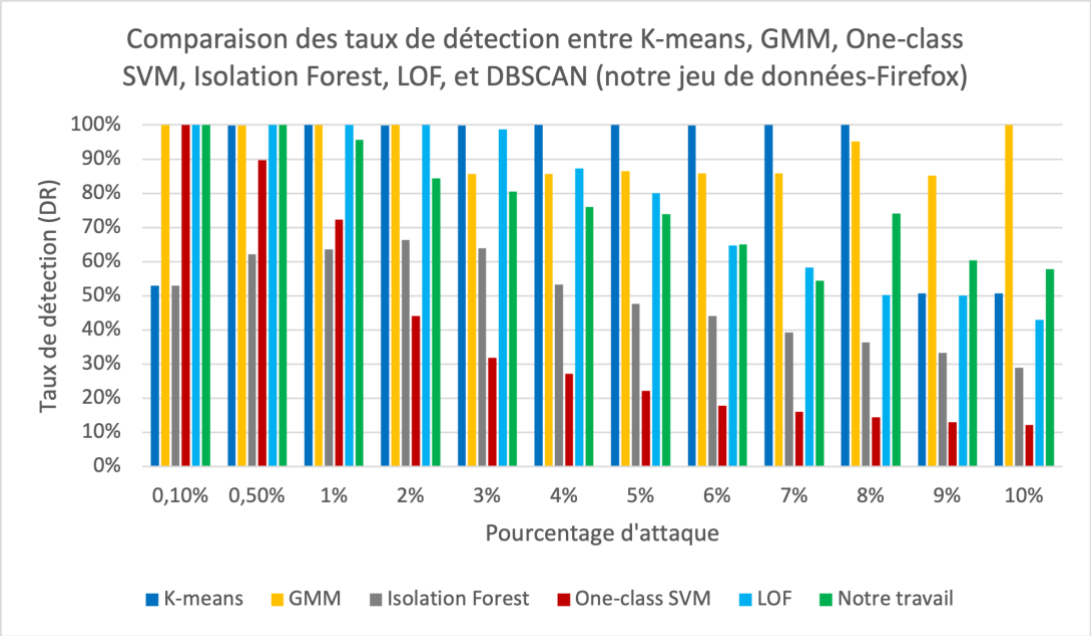


Figure 31. Comparaison des taux de détection dans notre jeu de données – Firefox

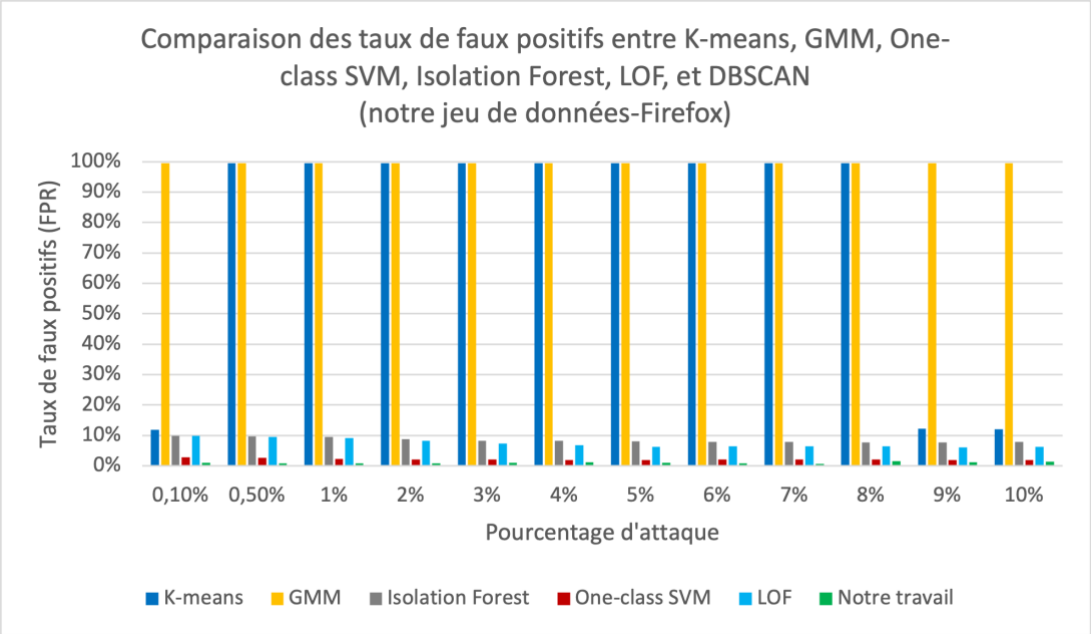


Figure 32. Comparaison des taux de faux positifs dans notre jeu de données – Firefox

Tableau 8. Résultats de notre jeu de données – Firefox

DIACRE-firefox	K-means		GMM		Isolation Forest		One-class SVM		LOF		Notre travail	
	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)
0.1%	53	11.94	100	99.51	53	9.95	100	2.9	100	9.9	**100**	**0.99**
0.5%	99.8	99.51	99.8	99.51	62.2	9.73	100	2.56	100	9.54	**100**	**0.89**
1%	100	99.49	100	99.47	63.5	9.45	72.3	2.29	100	9.09	**95.6**	**0.89**
2%	99.9	99.48	99.95	99.46	66.35	8.85	44.15	2.15	100	8.16	**84.35**	**0.81**
3%	99.9	99.53	85.7	99.51	63.86	8.33	31.86	2.1	98.76	7.25	**80.56**	**1.08**
4%	99.95	99.49	85.72	99.47	53.2	8.2	27.17	1.99	87.3	6.77	**76.05**	**1.12**
5%	99.92	99.50	86.38	99.48	47.66	8.01	22.14	1.99	80.06	6.31	**73.88**	**1.09**
6%	99.86	99.52	85.83	99.49	44.15	7.82	17.9	2.04	64.71	6.5	**65.03**	**0.88**
7%	99.97	99.48	85.78	99.45	39.31	7.79	16.02	2.02	58.22	6.36	**54.42**	**0.65**
8%	99.92	99.54	95.15	99.53	36.4	7.7	14.4	2.01	50.27	6.49	**74.1**	**1.59**
9%	50.76	12.13	85.22	99.47	33.3	7.69	13.04	2	50.04	6.03	**60.31**	**1.27**
10%	50.76	12.02	99.92	99.53	28.97	7.89	12.17	1.98	42.94	6.34	**57.84**	**1.29**

Dans les jeux de données avec Firefox (Figure 31, Figure 32, et Tableau 8), les résultats de K-means et GMM sont toujours mauvais car les FPR sont élevés. Les DR de K-means sont de 53% à 100% mais ses FPR sont très élevés de 11.94% à 99.53%. Les DR de GMM sont élevés de 85.7% à 100% et ses FPR sont aussi élevés, plus de 99.46%. Les DR de Isolation Forest sont faibles de 53% à 66.35%, tandis que ses FPR sont encore élevés de 8.33% à 9.95%.

LOF donne toujours des très bons DR de 98.76% à 100%, mais ses FPR restent élevés de 7.25% à 9.9%. Lorsque le taux d'attaque faible à 0.5%, One-class SVM a de bons DR de 100% et ses FPR sont assez bons de 2.56% à 2.9%. Quand le taux d'attaque est plus de 1%, ses FPR diminuent un peu mais ses DR diminuent très rapidement. Notre travail donne des bons DR de 80.56% à 100%, et ses FPR sont inférieurs à 1.08%.

2.4 Évaluation de temps d'exécution

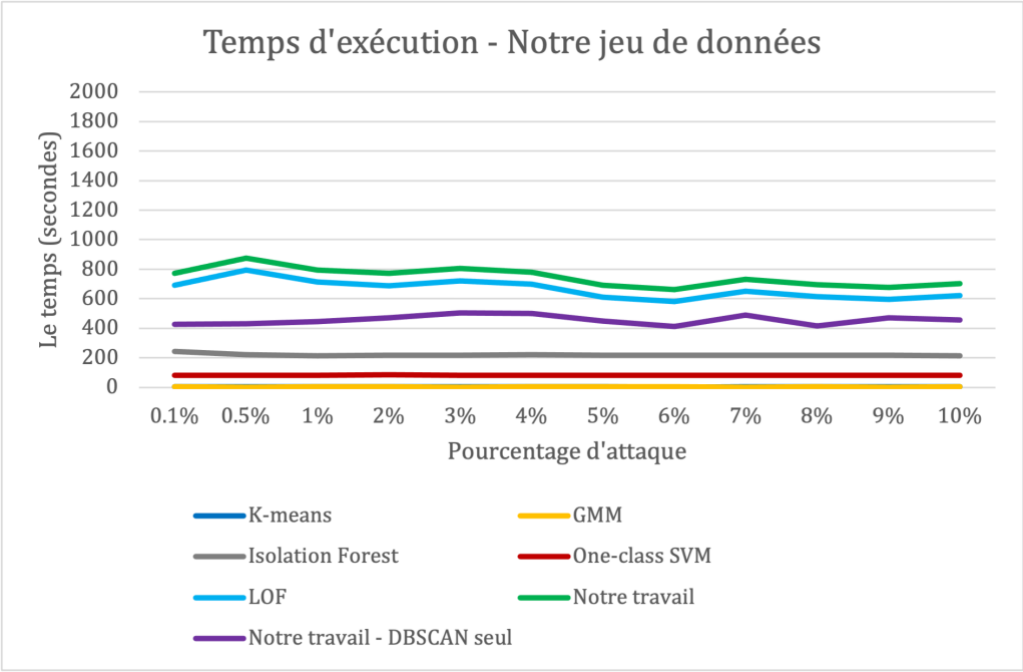


Figure 33. Temps d'exécution sur notre jeu de données

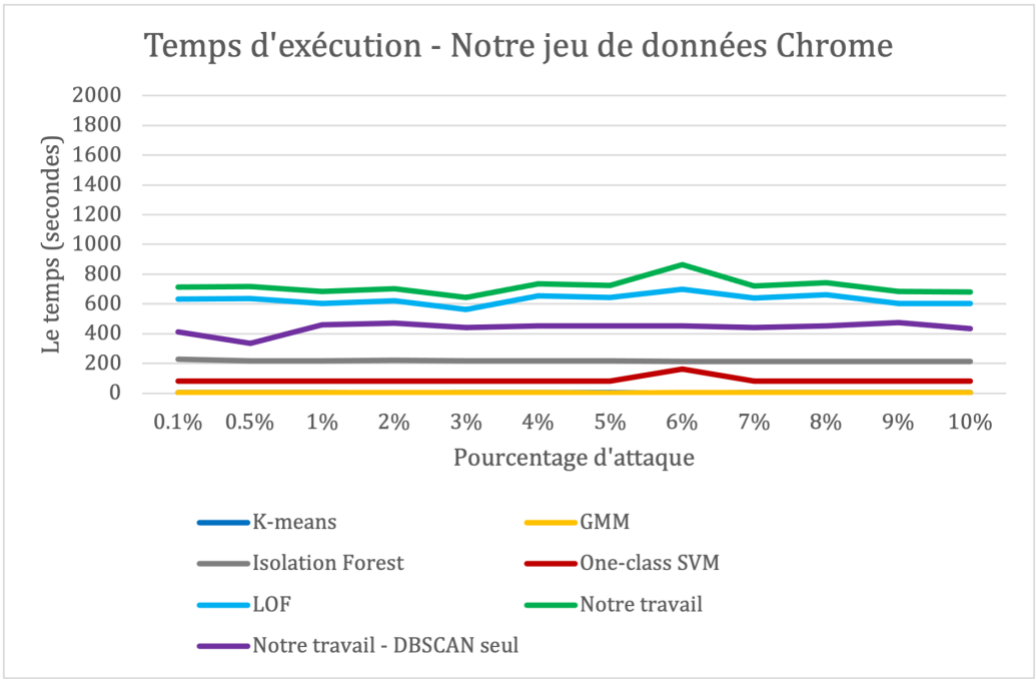


Figure 34. Temps d'exécution sur notre jeu de données Chrome

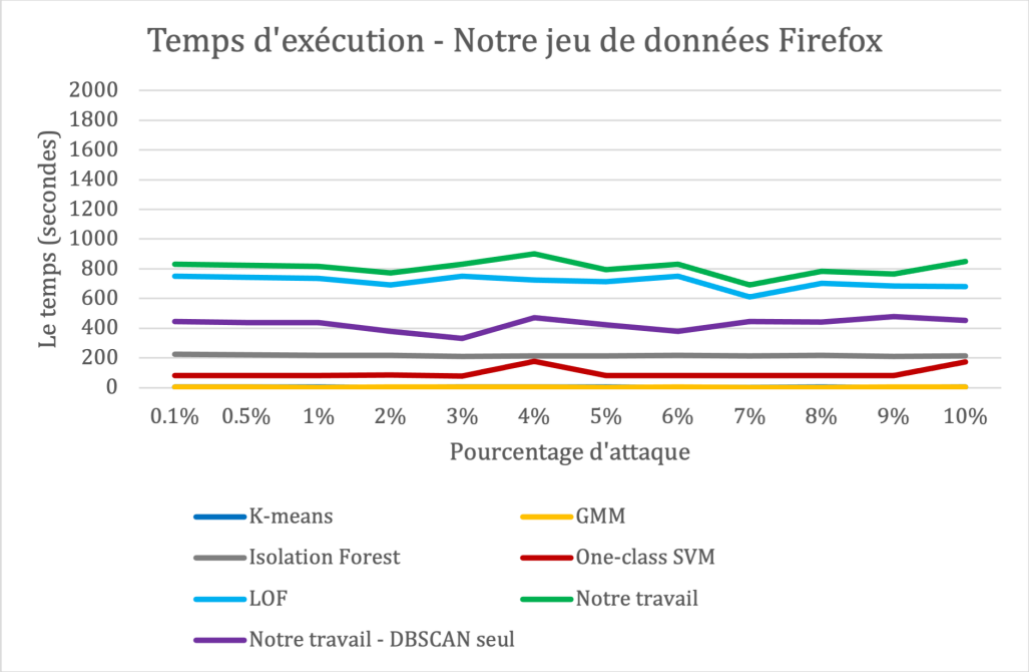


Figure 35. Temps d'exécution sur notre jeu de données Firefox

Tableau 9. Comparaison des temps d'exécution sur notre jeu de données

Algorithme	Temps moyen
K-means	0.083 minute
GMM	0.166 minute
Isolation Forest	3.7 minutes
One-Class SVM	**1.6 minutes**
LOF	12 minutes
Notre travail	**14 minutes**
Notre travail DBSCAN seul	**7.34 minutes**

Les figures (Figure 33 – Figure 35) et le Tableau 9 montrent les temps d'exécution sur notre jeu de données. L'exécution de K-means et GMM est très rapide mais leurs performances sont faibles. Le temps d'exécution d'Isolation Forest est aussi rapide, en moyenne 3.7 minutes, mais ses performances de détection sont faibles.

LOF peut détecter bien lorsque le taux d'attaque à 4% mais ses FPR sont très élevés. Les temps d'exécution de LOF sont longs, en moyenne 12 minutes. One-class SVM est plus rapide, en moyenne 1.6 minutes, mais ses performances ne sont bonnes que lorsque le taux d'attaque est extrêmement faible à 0.5%. Lorsque le taux d'attaque

augmente, ses DR diminuent rapidement donc nous pouvons l'utiliser pour les cas dont le pourcentage d'attaque est très faible.

Le temps d'exécution de notre travail est un peu plus long que LOF, en moyenne 14 minutes. Mais quand nous séparons le temps en deux parties : le premier temps est pour calculer et trouver le *minPts* de DBSCAN, en moyenne 6.66 minutes et le deuxième temps est pour lancer DBSCAN pour trouver des anomalies, en moyenne 7.34 minutes. Il est donc possible d'imaginer ne pas appliquer le calcul de l'hyperparamètre à chaque fois afin d'avoir un gain en temps d'exécution.

3 Détection des tunnels DNS-over-HTTPS (DoH)

3.1 Infrastructures

Le jeu de données CIRA-CIC-DoHBrw-2020 [86] est fourni par le projet de l'Institut Canadien pour la Cybersécurité (CIC). Il contient les trafics non-DoH, les trafics DoH normaux et les trafics DoH malveillants.

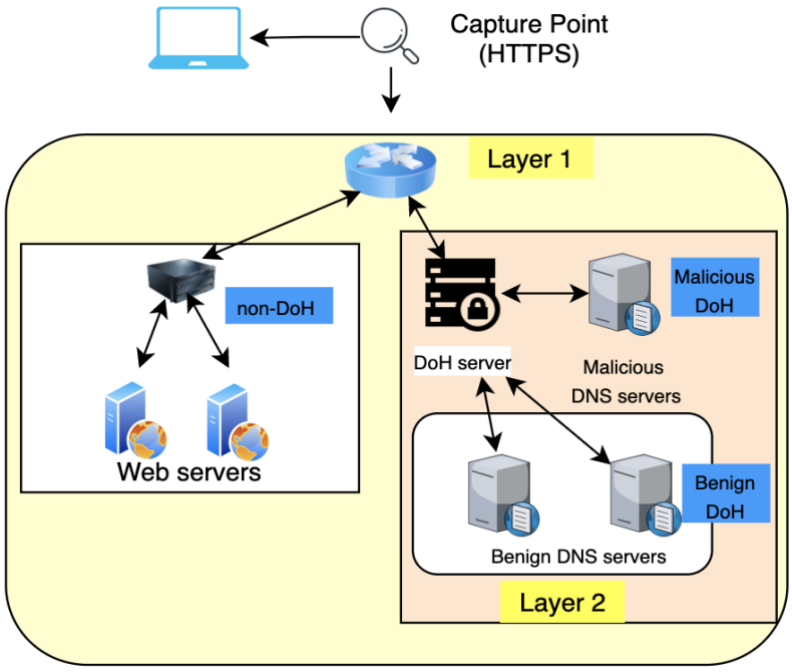


Figure 36. Topologie de réseau utilisé pour capturer le trafic pour le jeu de données CIRA-CIC-DoHBrw-2020 [86]

Le schéma de réseau utilisé pour capturer les trafics pour le jeu de données CIRA-CIC-DoHBrw-2020 est présenté à la Figure 36. Il y a des trafics DoH et non-DoH à la première couche et à la deuxième couche, les trafics DoH se distinguent en DoH normale et DoH malveillant. Les auteurs ont utilisé les navigateurs Web Mozilla Firefox et Google Chrome pour générer les trafics non-DoH et DoH normal. Un trafic non-DoH est généré en accédant à un site Web avec le protocole HTTPS. Un trafic

DoH normal est généré en utilisant la même technique que le non-DoH en naviguant sur le Web avec les navigateurs Web Mozilla Firefox et Google Chrome. Les navigateurs Web ont été configurés pour utiliser des serveurs DoH publics pour créer les trafics DoH normaux. Ensuite, ils ont utilisé une combinaison d'outils de tunneling DNS pour créer les trafics tunnels DoH malveillants. Ces outils peuvent créer des tunnels de données cryptées pour envoyer du trafic TCP encapsulé dans des requêtes DNS. Par conséquent, les requêtes DNS sont envoyées à l'aide de requêtes HTTPS cryptées par TLS à des serveurs DoH spéciaux. Les trafics générés par tous ces outils sont capturés par TCPDump au point de capture. Dans cette simulation, il y a quatre serveurs DoH : Adguard, Cloudfare, Google et Quad9 ; et trois outils de tunneling DNS : DNS2tcp, DNSCat2 et Iodine. La quantité de chaque type peut être vue dans le Tableau 10.

Tableau 10. Quantité de chaque type de trafic – CIRA-CIC-DoHBrw-2020

	Chrome	Firefox	Dnscat2	Dns2tcp	Iodine
Nombre d'ennregistrements	545464	420577	32721	132793	46598

3.2 Préparation

Dans la base de données CIRA-CIC-DoHBrw-2020, les trafics normaux sont créés par deux navigateurs : Mozilla Firefox et Google Chrome. Mozilla Firefox utilise Gecko-Driver tandis que Google Chrome utilise Chrome-Driver pour communiquer avec le navigateur. Notre analyse a montré que le trafic généré par eux est différent. Il est difficile d'identifier les valeurs aberrantes lorsque le trafic DoH normal a été généré à la fois par Firefox et Chrome. En conséquence, nous avons créé trois types de données : les trafics normaux créés par Firefox, les trafics normaux créés par Chrome, les trafics normaux créés par Firefox et Chrome. Pour chaque type de données DoH, nous avons généré 12 sous-ensembles de données de 100000 enregistrements contenant respectivement 0.1%, 0.5%, 1%, 2%, 3%, 4%, 5%, 6%. 7%. 8%, 9% et 10% des attaques. Dans chaque sous-ensemble de données, nous avons défini les pourcentages des trois types d'attaques pour qu'ils soient presque égaux. Par exemple, 3 % des attaques dans le sous-ensemble de données proviennent à parts égales de Dnscat2 (1%), de Dns2tcp (1%) et d'Iodine (1%).

3.3 Évaluation de performance

La Figure 37 et la Figure 38 montrent le DR et le FPR des six algorithmes pour les 12 sous-ensembles de données CIRA-CIC-DoHBrw-2020. Les résultats détaillés sont dans le Tableau 11.

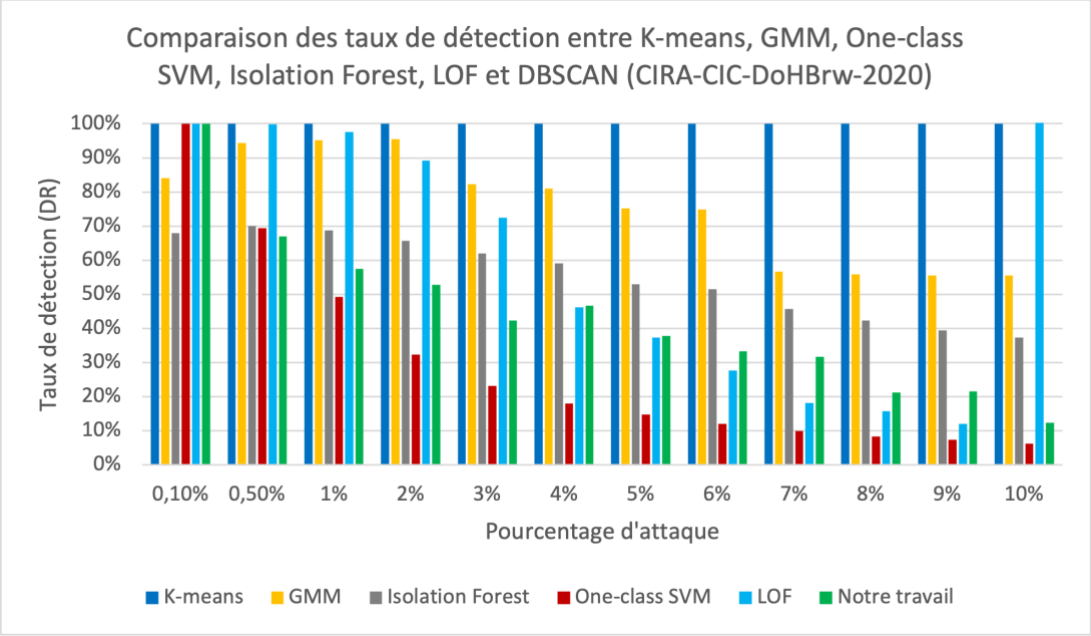


Figure 37. Comparaison des taux de détection dans CIRA-CIC-DoHBrw-2020

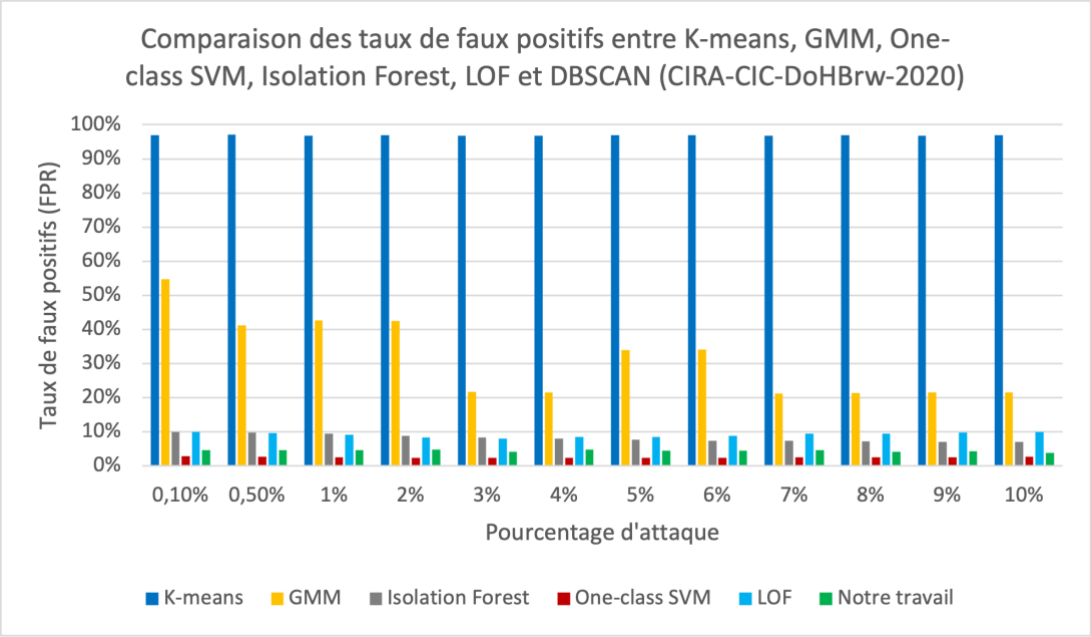


Figure 38. Comparaison des taux de faux positifs dans CIRA-CIC-DoHBrw-2020

Tableau 11. Résultats de CIRA-CIC-DoHBrw-2020

CIRA-mix	K-means		GMM		Isolation Forest		One-class SVM		LOF		Notre travail	
	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)
0.1%	100	96.92	84	54.69	68	9.94	100	2.9	100	9.9	**100**	**4.63**
0.5%	100	97.04	94.4	41.21	70	9.69	69.4	2.67	99.8	9.54	**67**	**4.61**
1%	100	96.85	95.1	42.57	68.8	9.4	49.2	2.53	97.5	9.11	**57.4**	**4.6**
2%	100	96.9	95.55	42.47	65.75	8.86	32.3	2.4	89.15	8.38	**52.8**	**4.73**
3%	100	96.81	82.2	21.69	62	8.39	23.2	2.37	72.5	8.06	**42.26**	**4.2**
4%	100	96.78	81.025	21.46	59.05	7.95	17.95	2.38	46.1	8.49	**46.67**	**4.8**
5%	99.98	96.87	75.2	33.94	52.94	7.74	14.84	2.37	37.38	8.55	**37.8**	**4.4**
6%	100	96.89	74.86	34.17	51.51	7.35	12	2.42	27.65	8.87	**33.3**	**4.41**
7%	100	96.85	56.7	21.25	45.62	7.31	9.91	2.48	18.1	9.39	**31.68**	**4.58**
8%	100	96.87	55.9	21.29	42.28	7.19	8.36	2.53	15.67	9.5	**21.16**	**4.16**
9%	99.98	96.76	55.54	21.49	39.44	7.08	7.35	2.57	12.04	9.79	**21.58**	**4.36**
10%	100	96.86	55.59	21.46	37.35	6.96	6.25	2.63	10.76	9.91	**12.29**	**3.76**

Comme nous l'avons expliqué avant, nous supposons que les attaques sont contrôlées à un faible taux pour rester sous les seuils classiques de détection, lors de la comparaison des performances des algorithmes, nous ne considérerons que les résultats des sous-ensembles de données avec des taux d'attaque allant jusqu'à 3 %.

K-means et GMM donnent de mauvais résultats. Les DR de K-means sont élevés à 100% mais ses FPR sont aussi élevés, plus de 96.81%. Les DR de GMM sont bons de 82.2% à 95.55% mais ses FPR sont élevés, plus de 21.69%. Les résultats de Isolation Forest ne sont pas bons avec des faibles DR de 62% à 70% et des FPR élevés de 8.39% à 9.94%.

LOF obtient d'excellents DR (supérieur à 89.15%) jusqu'à 2% d'attaque et plus de 72.5% à 3% d'attaque. Cependant, ses FPR sont comme toujours élevés, plus de 8.06%. One-class SVM est bon à 0.1% d'attaque mais ses performances de détection chutent ensuite. Les DR de notre travail a un peu les mêmes résultats que One-class SVM. Il détecte bien les attaques à 0.1% (DR à 100% et FPR à 4.6%). Cependant, ses performances de détection chutent ensuite.

Lorsque nous séparons le trafic normal dans Chrome et Firefox, les résultats obtenus à partir de tous les algorithmes sont bien meilleurs, comme le montrent le Tableau 12 les figures Figure 39 – Figure 42. Les résultats détaillés sont dans les tableaux : Tableau 13 et Tableau 14.

Tableau 12. Comparaison des résultats de CIRA-CIC-DoHBrw-2020 avec des taux d'attaque allant jusqu'à 3%

	K-means		GMM		Isolation Forest		One-class SVM		LOF		Notre travail	
	DR (%)	FPR (%)	DR (%)	FPR (%)	DR (%)	FPR (%)	DR (%)	FPR (%)	DR (%)	FPR (%)	DR (%)	FPR (%)
CIRA-CIC-DoHBrw-2020												
0.1%	100	96.92	84	54.69	68	9.94	100	2.9	100	9.9	**100**	**4.63**
0.5%	100	97.04	94.4	41.21	70	9.69	69.4	2.67	99.8	9.54	**67**	**4.61**
1%	100	96.85	95.1	42.57	68.8	9.4	49.2	2.53	97.5	9.11	**57.4**	**4.6**
2%	100	96.9	95.55	42.47	65.75	8.86	32.3	2.4	89.15	8.38	**52.8**	**4.73**
3%	100	96.81	82.2	21.69	62	8.39	23.2	2.37	72.5	8.06	**42.26**	**4.2**
CIRA-CIC-DoHBrw-2020 - Chrome												
0.1%	84	13.68	100	48.56	71	9.93	100	2.9	100	9.9	**100**	**1.17**
0.5%	88	13.64	92	48.16	80	9.64	94.6	2.54	100	9.54	**100**	**1.05**
1%	85.1	13.5	100	46.52	79	9.3	70.1	2.32	99.2	9.09	**100**	**1.99**
2%	84.8	13.12	85	48.4	78.6	8.6	42.05	2.2	96.45	8.23	**61.75**	**1.19**
3%	85.1	13.02	100	46.22	71.63	8.09	30.6	2.14	69.26	8.16	**66.56**	**1.53**
CIRA-CIC-DoHBrw-2020 - Firefox												
0.1%	94	94.67	62	24.26	62	9.94	100	2.91	100	9.9	**100**	**3.49**
0.5%	99.4	94.75	57.4	24.55	67.4	9.71	85.2	2.58	100	9.54	**100**	**4.54**
1%	99.9	94.67	57.1	25.6	67	9.42	58.6	2.43	100	9.09	**100**	**3.18**
2%	99.95	94.59	53.7	24.55	58.7	9.00	37.7	2.29	96.85	8.22	**43.25**	**3.62**
3%	99.96	94.63	63.23	39.45	53.53	8.65	28.4	2.21	84.46	7.69	**45.03**	**5.45**

Dans les jeux de données avec Chrome (Figure 39, Figure 40, et Tableau 13), K-means donne des bons DR, plus de 84% mais ses FPR sont élevés de 13.02% à 13.68%. GMM détecte mieux que K-means avec DR de 85% à 100% mais ses FPR sont plus élevés de 46.22% à 48.56%. Isolation Forest a de bons DR de 71% à 80% et des FPR élevés, plus de 8.09%.

LOF a d'excellents DR (plus de 96.4%) jusqu'à 2% d'attaque mais ses FPR sont toujours élevés, de 8.16% à 9.9%. One-class SVM a des DR de 30.06% à 100% et des FPR inférieurs à 3%. Le comportement de One-class SVM est toujours le même avec

de très bons résultats de détection jusqu'à 0.5% (DR supérieur à 94.6% et FPR inférieur à 2.9%) mais ses performances chutent dès ce taux d'attaque dépassé. Notre travail est excellent jusqu'à 1% d'attaque (DR à 100% et FPR inférieur à 1.99%). Au-delà, les DR tombent entre 61 et 66%. Cependant, le FPR reste extrêmement bas à moins de 1.53%.

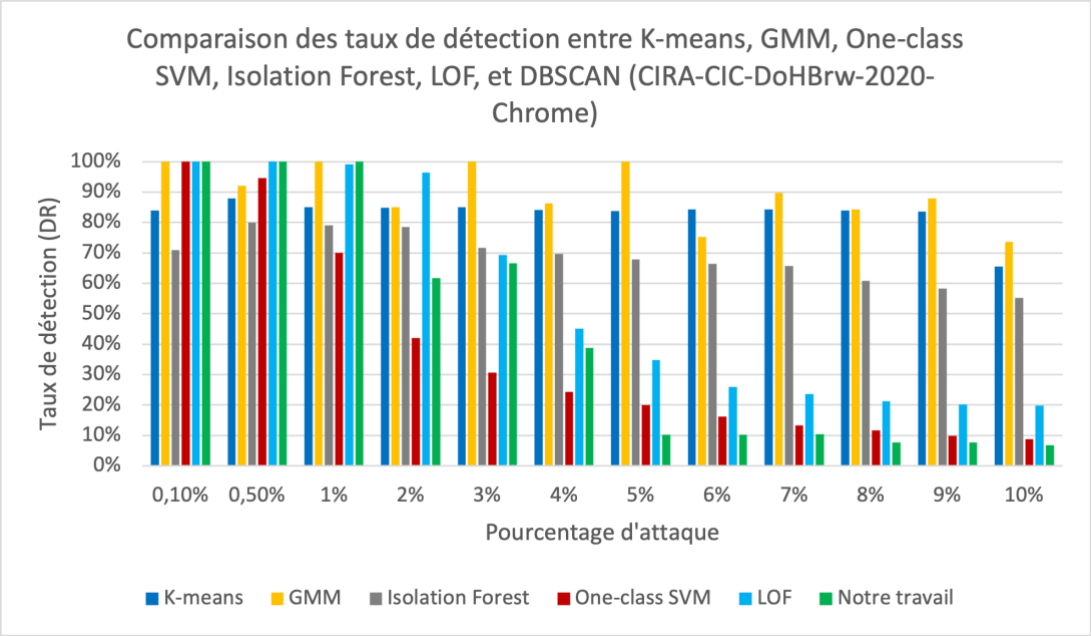


Figure 39. Comparaison des taux de détection dans CIRA-CIC-DoHBrw-2020 – Chrome

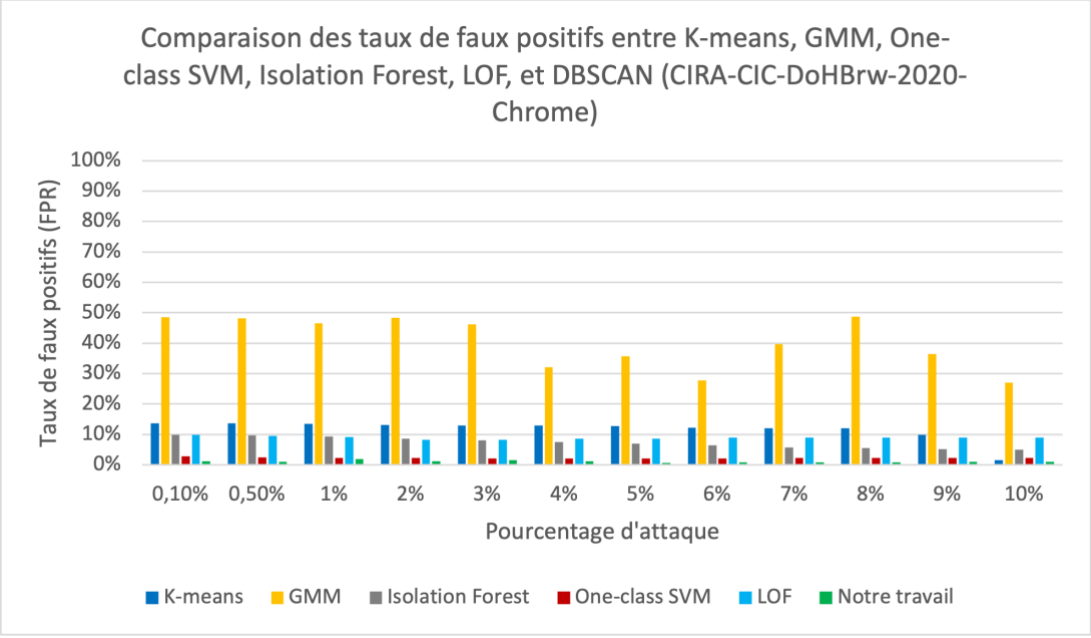


Figure 40. Comparaison des taux de faux positifs dans CIRA-CIC-DoHBrw-2020 – Chrome

Tableau 13. Résultats de CIRA-CIC-DoHBrw-2020-Chrome

CIRA-Chrome	K-means		GMM		Isolation Forest		One-class SVM		LOF		Notre travail	
	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)
0.1%	84	13.68	100	48.56	71	9.93	100	2.9	100	9.9	**100**	**1.17**
0.5%	88	13.64	92	48.16	80	9.64	94.6	2.54	100	9.54	**100**	**1.05**
1%	85.1	13.5	100	46.52	79	9.3	70.1	2.32	99.2	9.09	**100**	**1.99**
2%	84.8	13.12	85	48.4	78.6	8.6	42.05	2.2	96.45	8.23	**61.75**	**1.19**
3%	85.1	13.02	100	46.22	71.63	8.09	30.6	2.14	69.26	8.16	**66.56**	**1.53**
4%	84.05	12.98	86.27	32.04	69.67	7.51	24.35	2.1	45.12	8.53	**38.75**	**1.21**
5%	83.76	12.78	99.96	35.65	67.8	6.95	20.02	2.1	34.8	8.69	**10.16**	**0.74**
6%	84.36	12.17	75.31	27.75	66.48	6.39	16.21	2.15	25.86	8.98	**10.2**	**0.82**
7%	84.37	12.03	89.75	39.69	65.74	5.8	13.35	2.22	23.67	8.97	**10.37**	**0.88**
8%	83.92	12.04	84.32	48.72	60.73	5.58	11.63	2.25	21.16	9.02	**7.72**	**0.84**
9%	83.55	9.9	87.84	36.40	58.36	5.21	9.9	2.31	20.17	8.99	**7.,7**	**1.02**
10%	65.6	1.62	73.63	27.03	55.24	4.97	8.82	2.36	19.8	8.91	**6.87**	**0.99**

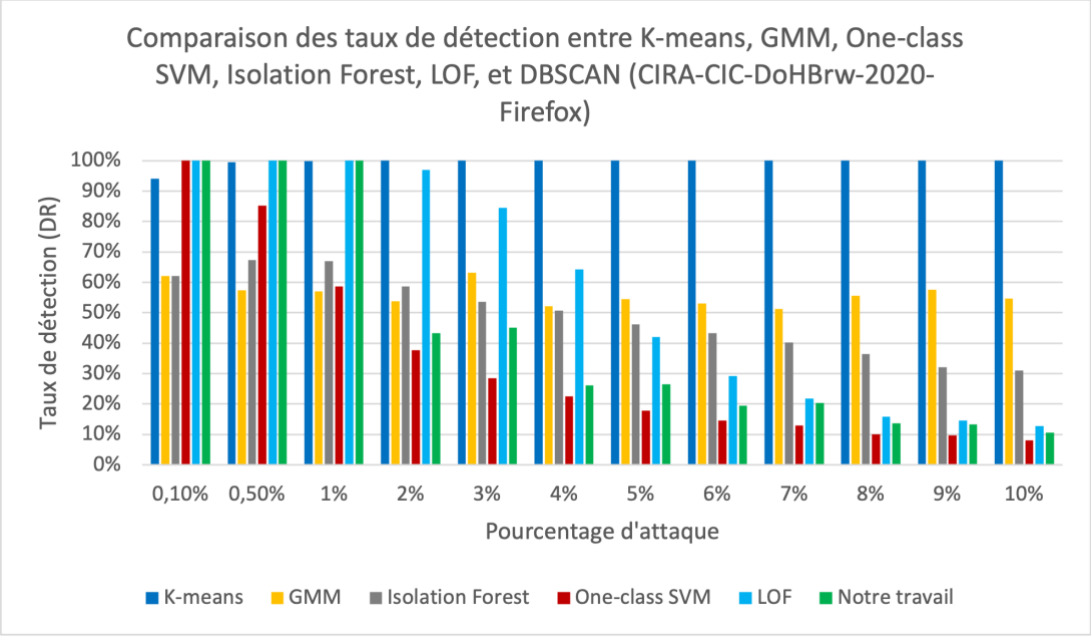


Figure 41. Comparaison des taux de de détection dans CIRA-CIC-DoHBrw-2020 – Firefox

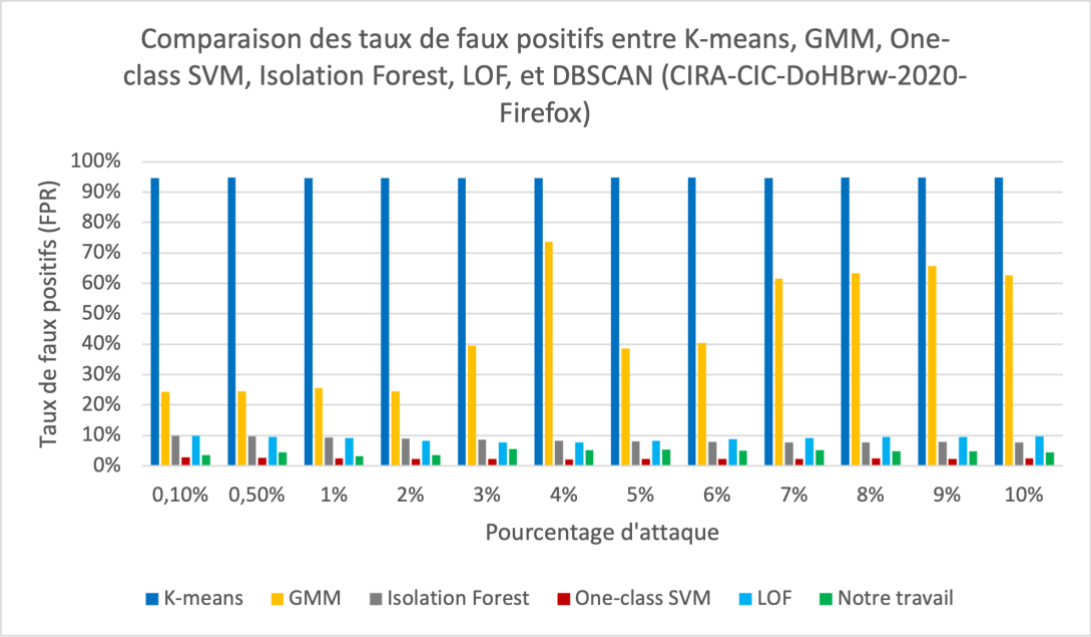


Figure 42. Comparaison des taux de faux positifs dans CIRA-CIC-DoHBrw-2020 – Firefox

Tableau 14. Résultats de CIRA-CIC-DoHBrw-2020-Firefox

CIRA-firefox	K-means		GMM		Isolation Forest		One-class SVM		LOF		Notre travail	
	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)
0.1%	94	94.67	62	24.26	62	9.94	100	2.91	100	9.9	**100**	**3.49**
0.5%	99.4	94.75	57.4	24.55	67.4	9.71	85.2	2.58	100	9.54	**100**	**4.54**
1%	99.9	94.67	57.1	25.6	67	9.42	58.6	2.43	100	9.09	**100**	**3.18**
2%	99.95	94.59	53.7	24.55	58.7	9.00	37.7	2.29	96.85	8.22	**43.25**	**3.62**
3%	99.96	94.63	63.23	39.45	53.53	8.65	28.4	2.21	84.46	7.69	**45.03**	**5.45**
4%	100	94.58	52.075	73.63	50.65	8.3	22.57	2.18	64.3	7.73	**26.12**	**5.09**
5%	100	94.76	54.5	38.59	46.14	8.09	17.74	2.22	42.02	8.31	**26.46**	**5.33**
6%	100	94.73	53.03	40.36	43.26	7.87	14.6	2.,26	29.23	8.77	**19.41**	**4.93**
7%	99.98	94.65	51.31	61.57	40.17	7.72	13.02	2.24	21.82	9.1	**20.35**	**5.26**
8%	100	94.69	55.5	63.25	36.47	7.69	10.12	2.38	15.77	9.49	**13.725**	**4.88**
9%	100	94.72	57.51	65.65	32.14	7.8	9.64	2.33	14.51	9.55	**13.3**	**4.82**
10%	100	94.69	54.6	62.64	31.04	7.66	8.02	2.44	12.79	9.69	**10.64**	**4.49**

Dans les jeux de données avec Firefox (Figure 41, Figure 42, Tableau 14), K-means et GMM donnent de mauvais résultats. Les DR de K-means sont élevés de 94% à 99.96% mais ses FPR sont plus élevés de 94.59% à 94.75%. Les DR de GMM sont faibles de 53.7% à 63.23% et ses FPR sont élevés de 24.26% à 39.45%. Les DR de

Isolation Forest sont faibles de 53.53% à 67.4% et ses FPR sont élevés de 8.65% à 9.94%.

LOF donne de très bonnes valeurs de DR (de 84.46% à 100%) jusqu'à 3% d'attaques mais ses FPR sont toujours élevés de 7.69% à 9.9%. One-class SVM est bon (DR de 85.2% à 100% et des FPR de 2.58% à 2.91%) uniquement jusqu'à 0.5% d'attaque. Notre travail est très bon jusqu'à 1% d'attaque (DR 100% et FPR de 3.18% à 4.54 %). Malheureusement, le DR tombe à partir de 2% d'attaque.

3.4 Évaluation de temps d'exécution

Les figures (Figure 43 – Figure 45) et le Tableau 15 montrent les temps d'exécution sur CIRA-CIC-DoHBrw-2020. K-means, GMM et Isolation Forest sont très rapides mais peu performants pour la détection.

One-class SVM est rapide, en moyenne 1.8 minutes, mais ses performances ne sont bonnes que lorsque le taux d'attaque est très faible à 0.5%. LOF peut détecter correctement lorsque le taux d'attaque à 3% mais avec un FPR élevé. Les temps d'exécution de LOF sont plus longs, en moyenne 7.5 minutes.

Le temps d'exécution de notre travail est un peu plus grand que LOF, en moyenne 8 minutes. Toutefois, si nous séparons le temps en deux parties : le premier temps est pour calculer et trouver le *minPts* de DBSCAN, en moyenne 3.8 minutes et le deuxième temps est pour lancer DBSCAN pour trouver des anomalies, en moyenne 4.2 minutes. Il est possible d'optimiser son utilisation. Pour un type de données particulier, par exemple des trafics d'une entreprise, nous pouvons lancer une fois notre méthode pour calculer et trouver le *minPts*. Ensuite, nous appliquons directement cette valeur dans DBSCAN pour les détections suivantes ce qui permet de diviser le temps d'exécution par 2.

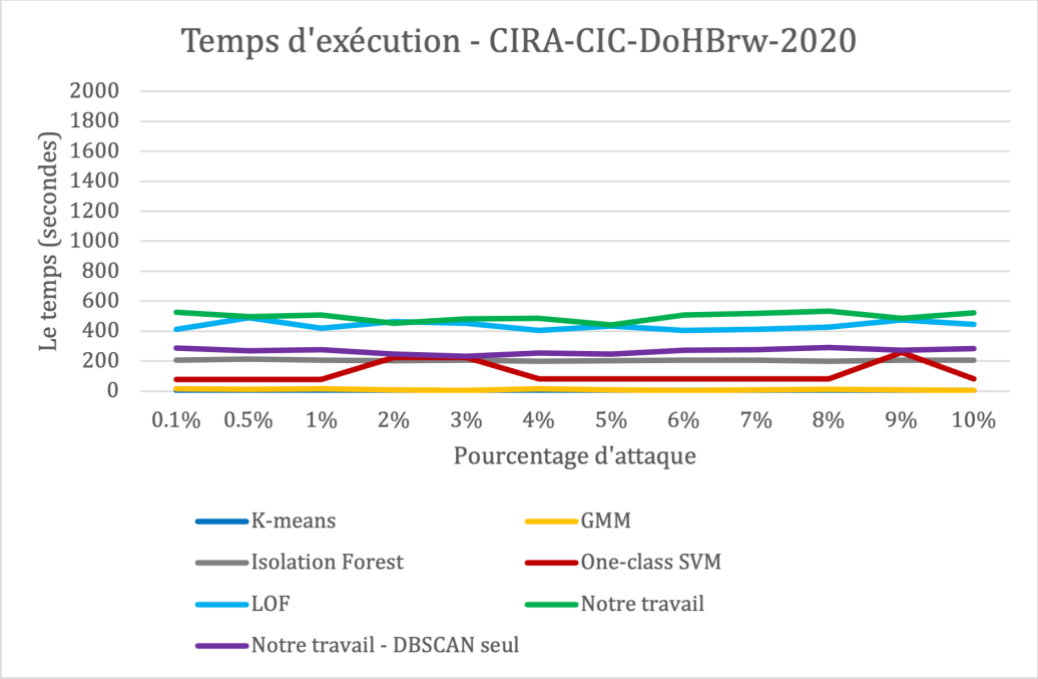


Figure 43. Temps d'exécution sur CIRA-CIC-DoHBrw-2020

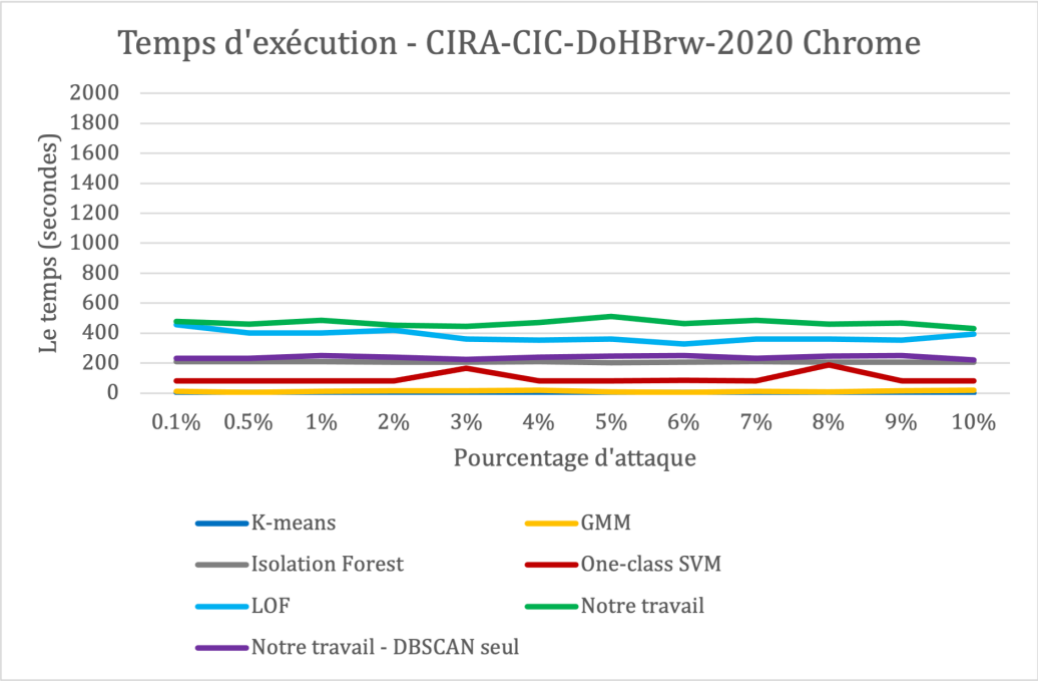


Figure 44. Temps d'exécution sur CIRA-CIC-DoHBrw-2020 Chrome

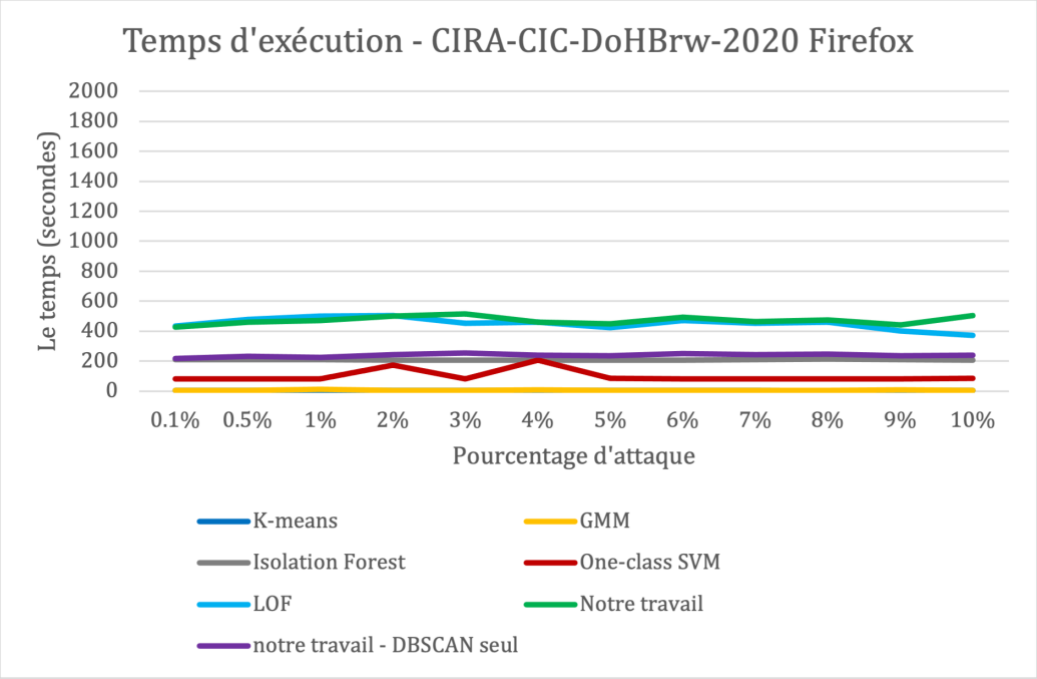


Figure 45. Temps d'exécution sur CIRA-CIC-DoHBrw-2020 Firefox

Tableau 15. Comparaison des temps d'exécution sur CIRA-CIC-DoHBrw-2020

Algorithme	Temps moyen
K-means	0.05 minute
GMM	0.15 minute
Isolation Forest	3.4 minutes
One-Class SVM	1.8 minutes
LOF	7.5 minutes
Notre travail	8 minutes
Notre travail DBSCAN seul	4.2 minutes

4 Validation de la valeur d'initialisation de AutoRoC-DBSCAN

L'algorithme que nous avons proposé pour automatiser le calcul des hyperparamètres de DBSCAN dépend de la valeur de *minPts*. Cette valeur est reprise pour définir la valeur K (le nombre de voisins dans l'algorithme KNN) et Delta (le nombre de points dans chaque plage de valeurs qui sera utilisé pour calculer le taux de changement). Nous avions fixé cette valeur à 3%. En effet, *minPts* étant le nombre de points de données requis pour former un cluster, cette valeur correspondait au seuil de taux

d'attaque que nous nous étions fixé. Cette valeur a donné de bons résultats par rapport aux objectifs de détection qui était un DR supérieur à 80% et un FPR inférieur à 2%.

Nous proposons dans cette section d'étudier l'influence de la valeur *minPts* sur la qualité de détection. Cela permet deux choses. Tout d'abord nous pouvons ainsi valider la valeur que nous avons choisie dans notre travail. Le deuxième intérêt est de permettre à quelqu'un ayant des objectifs de détection différents en termes de DR et FPR de choisir la valeur la plus adaptée.

Nous avons fait une expérimentation en faisant varier la valeur de *minPts* de 0.1%, 0.5%, 1%, 2%, 3%, 4%, et 5%. Nous avons comparé les taux de détection et les taux de faux positifs lors du changement de *minPts*. Les résultats sont montrés dans les figures : Figure 46 - Figure 57.

Les figures présentant la comparaison des taux de détection lors du changement de *minPts* doivent être lues ainsi. Chaque courbe correspond à un objectif de DR (DR > 75%, DR > 80%, DR > 85%, DR > 90%, DR > 95%). Pour chaque taux de détection désiré, nous pouvons visualiser pour chaque valeur de *minPts* (axe des abscisses) et quel est le pourcentage d'attaque maximal que notre méthode peut détecter avec un DR correspondant à la courbe. Par exemple, dans la Figure 46, nous pouvons voir que lorsque la valeur de *minPts* est égale à 3%, notre algorithme permet d'avoir un DR > 75% jusqu'à 3% d'attaque, un DR > 80% jusqu'à 1% d'attaque, un DR > 90% jusqu'à 0.5% d'attaque et un DR > 95% jusqu'à 0.1% d'attaque.

Dans les figures illustrant la comparaison des taux de faux positifs lors du changement de *minPts*, nous montrons la valeur minimum, la valeur maximum et la valeur moyenne des taux de faux positifs pour chaque *minPts*.

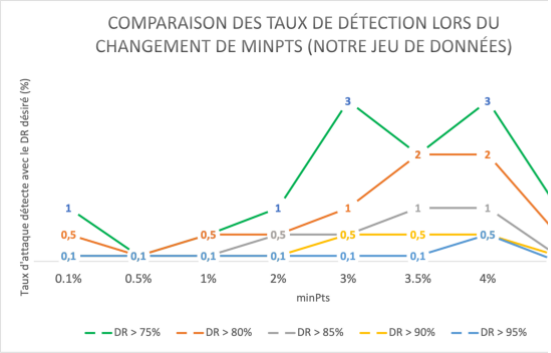


Figure 46. Comparaison des taux de détection lors du changement de *minPts* dans notre jeu de données

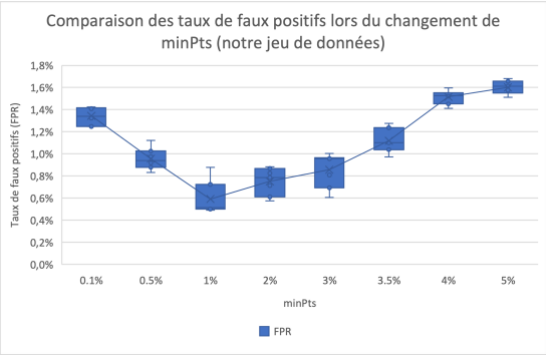


Figure 47. Comparaison des taux de faux positifs lors du changement de *minPts* dans notre jeu de données

Les figures (Figure 46 et Figure 47) montrent la comparaison des taux de détection et des taux de faux positifs lors du changement de *minPts* dans notre jeu de données (tunnel DNS). On peut voir que les taux maximaux d'attaque détectés sont meilleurs

pour les valeurs de *minPts* comprises entre 3% à 4%. Par contre, le FPR croit de manière linéaire dans cette plage de valeurs.

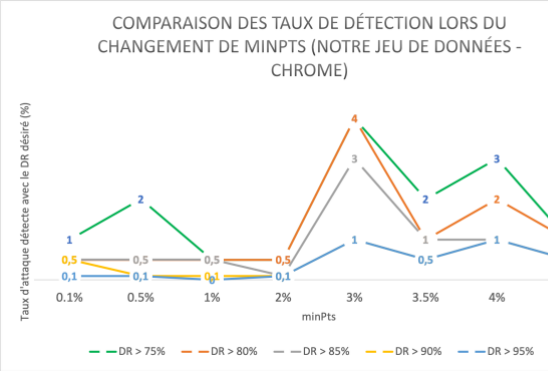


Figure 48. Comparaison des taux de détection lors du changement de *minPts* dans notre jeu de données - Chrome

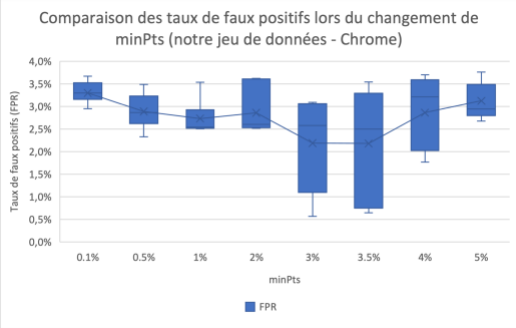


Figure 49. Comparaison des taux de faux positifs lors du changement de *minPts* dans notre jeu de données - Chrome

Les figures (Figure 48 et Figure 49) montrent la comparaison des taux de détection et des taux de faux positifs lors du changement de *minPts* dans notre jeu de données chrome. On peut voir ici que la meilleure valeur de *minPts* est 3% à la fois en terme de taux d'attaque détecté que de FPR.

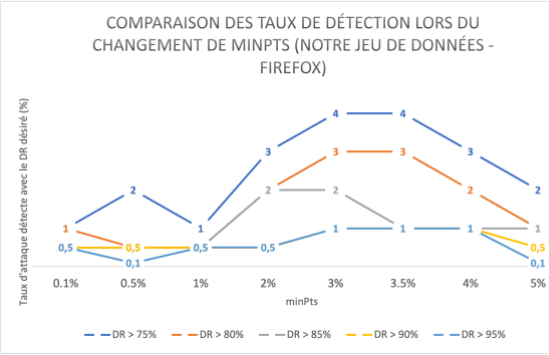


Figure 50. Comparaison des taux de détection lors du changement de *minPts* dans notre jeu de données - Firefox

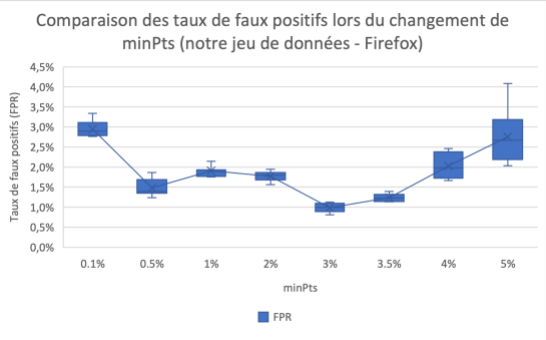


Figure 51. Comparaison des taux de faux positifs lors du changement de *minPts* dans notre jeu de données – Firefox

Les figures (Figure 50 et Figure 51) montrent la comparaison des taux de détection et des taux de faux positifs lors du changement de *minPts* dans notre jeu de données firefox. Cette fois les meilleurs taux d'attaques se situent entre 3% et 4%. Le FPR est sensiblement le même entre 3% et 3.5%.

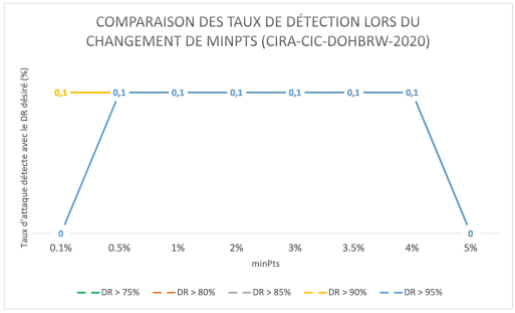


Figure 52. Comparaison des taux de détection lors du changement de *minPts* dans CIRA-CIC-DoHBrw-2020

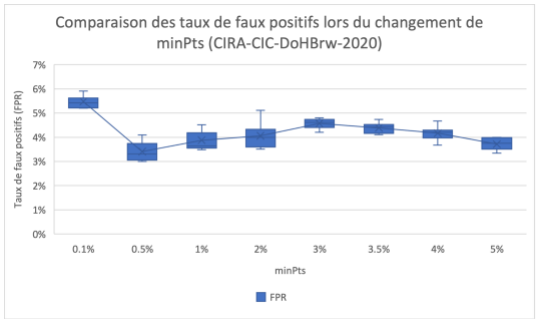


Figure 53. Comparaison des taux de faux positifs lors du changement de *minPts* dans CIRA-CIC-DoHBrw-2020

Les figures (Figure 52 et Figure 53) montrent la comparaison des taux de détection et des taux de faux positifs lors du changement de *minPts* dans le jeu de données CIRA-CIC-DoHBrw-2020. Dans le cas *minPts* est de 0.1% à 4%, les taux d'attaque maximaux détectés sont de 0.1% et les taux de faux positifs sont assez similaires.

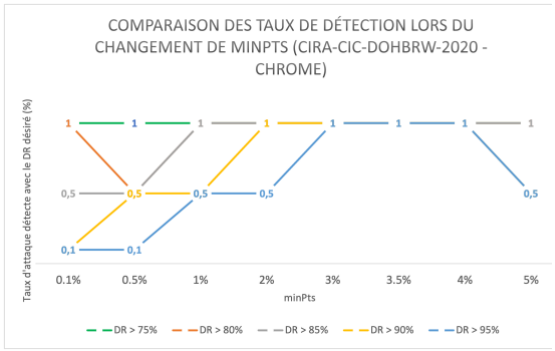


Figure 54. Comparaison des taux de détection lors du changement de *minPts* dans CIRA-CIC-DoHBrw-2020 - Chrome

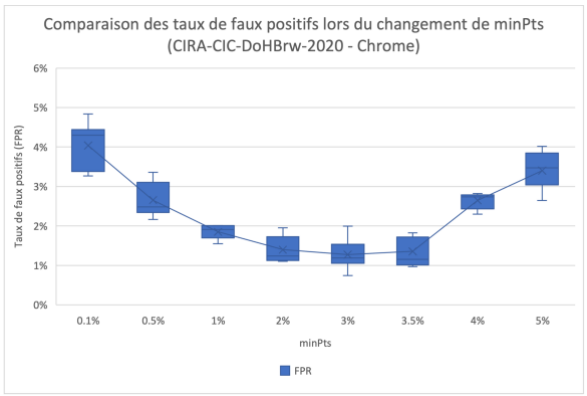


Figure 55. Comparaison des taux de faux positifs lors du changement de *minPts* dans CIRA-CIC-DoHBrw-2020 - Chrome

Lors que nous séparons les trafics normaux en chrome et firefox, les résultats obtenus sont meilleurs. Les figures (Figure 54 et Figure 55) montrent la comparaison des taux de détection et des taux de faux positifs lors du changement de *minPts* dans le jeu de données CIRA-CIC-DoHBrw-2020 - chrome. Les meilleurs taux d'attaque détectés se situent entre des valeurs de 3% à 4% pour *minPts*. Dans cette plage de valeur, le FPR est similaire entre 3% et 3.5% mais augmente significativement à 4%.

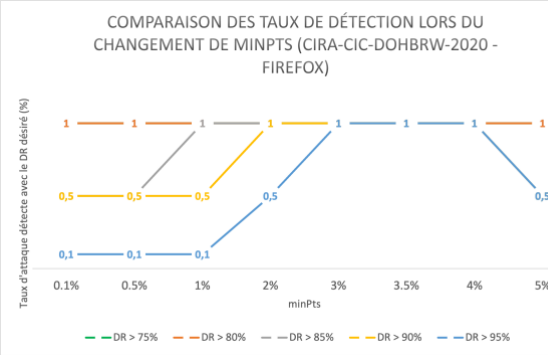


Figure 56. Comparaison des taux de détection lors du changement de *minPts* dans CIRA-CIC-DoHBrw-2020 - Firefox

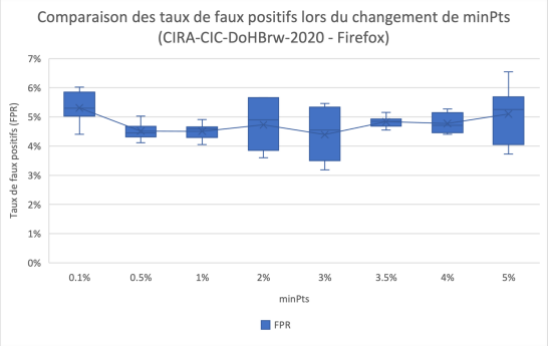


Figure 57. Comparaison des taux de faux positifs lors du changement de *minPts* dans CIRA-CIC-DoHBrw-2020 - Firefox

Les figures (Figure 56 et Figure 57) montrent la comparaison des taux de détection et des taux de faux positifs lors du changement de *minPts* dans le jeu de données CIRA-CIC-DoHBrw-2020 - firefox. Nous voyons ici aussi que les meilleures valeurs de *minPts* se situent entre 3% et 4%.

Comme les résultats présentés ci-dessus montrent que quelque soit le jeu de données choisi, les meilleures valeurs de *minPts* sont 3% et 3.5%. La valeur de 3% semble obtenir de meilleur FPR ce qui confirme notre choix initial.

5 Conclusion

Dans ce chapitre, nous avons évalué les performances en termes de détection et de temps d'exécution de notre travail. Nous avons utilisé 2 jeux de données : un jeu de données contenant les trafics DNS normaux et les tunnels DNS que nous avons créé ; un jeu de données contenant les trafics non-DoH, les trafics DoH normaux et les trafics tunnels DoH. Nous avons utilisé CICFlowMeter pour extraire des caractéristiques du réseau pour appliquer notre méthode.

Nous avons comparé notre méthode avec cinq autres algorithmes : K-means, GMM, Isolation Forest, One-class SVM, et LOF. Notre travail offre des résultats plus équilibrés que les autres algorithmes en termes de DR, FPR et taux d'attaque dans le jeu de données. En effet, K-means, GMM, Isolation Forest obtiennent de mauvais résultats de détection. One-class SVM est rapide à exécuter mais ne détecte que lorsque le taux d'attaque est extrêmement bas. LOF offre de très bon DR qui peuvent être meilleurs que notre algorithme mais le FPR avoisine tout le temps les 10%. Les performances d'exécution de notre algorithme et de LOF sont assez comparables. Par contre, si nous ne recalculons pas tout le temps les valeurs des hyperparamètres de DBSCAN, alors notre approche est deux fois plus rapide que LOF.

Enfin, nous avons validé expérimentalement la valeur de *minPts* qui détermine les calculs des hyperparamètres de DBSCAN.

Chapitre 6. Conclusion et Perspectives

Parallèlement à la forte croissance d'Internet, le nombre de cyberattaques a également augmenté et la protection de la vie privée et des données sensibles est aujourd'hui un grand défi. En plus des attaquants solitaires qui essaient de trouver les vulnérabilités des systèmes informatiques pour gagner de l'argent ou simplement faire leurs preuves, il existe de plus en plus de groupes d'attaquants structurés et hautement qualifiés qui développent des attaques complexes, sophistiquées, préméditées et ciblées pour atteindre des objectifs prédéterminés. Au lieu d'attaquer le système immédiatement, ces attaques sont planifiées et contrôlées afin de rester non détectées le plus longtemps possible et pénétrer profondément dans le système ciblé. En particulier, pour envoyer des ordres aux machines infectées ou exfiltrer des données hors du réseau ciblé, les attaquants utilisent souvent des canaux C&C cachés dans un trafic autorisé tels que les protocoles DNS ou HTTPS. L'objectif de cette thèse est donc de fournir une approche pour détecter ces trafics autorisés mais illégitimes.

Nous nous sommes concentrés sur la détection du trafic C&C encapsulé dans des protocoles réseau autorisés afin de détecter des trafics autorisés mais illégitimes et en particulier l'utilisation frauduleuse du protocole DNS. En effet, c'est un protocole essentiel au bon fonctionnement du réseau et par conséquent les systèmes d'information de sécurité des entreprises doivent l'autoriser. Les paquets DNS contiennent de nombreux champs et en-têtes dans lesquels des données peuvent être cachées. Ainsi, le protocole DNS fait partie des protocoles légitimes ciblés par les attaquants pour dissimuler leurs canaux de communication C&C [103]. Plus récemment, un nouveau protocole appelé DoH a été proposé pour améliorer la protection de la vie privée des personnes naviguant sur Internet en encapsulant les requêtes et les réponses DNS dans des connexions HTTPS. Si cela améliore la sécurité des personnes, cela offre en même temps une nouvelle opportunité aux attaquants pour cacher leur tunnel DNS dans un trafic chiffré et autorisé HTTPS [104]. C'est une double encapsulation qui rend encore plus complexe la détection des tunnels C&C.

Les problèmes qui ont été soulevés dans ce travail de thèse sont les suivants :

1. Comment détecter des attaques qui sont contrôlées pour passer sous les seuils de détection des méthodes traditionnels comme les IDS ou les SIEMs ?
2. Comment distinguer le trafic malveillant du trafic normal lorsqu'ils utilisent le même numéro de port et la même structure de message ?
3. Comment garantir la performance de notre approche dans la détection des trafics C&C ?

Nous ne pouvons pas détecter les attaques en écrivant simplement des règles basées sur l'adresse IP, le port ou même des champs du protocole applicatif des messages. Il

est nécessaire de mettre en œuvre une analyse du comportement réseau pour détecter des anomalies qui ont des signaux inhabituels. Ainsi, dans notre problème qui consiste à détecter des tunnels C&C, il y a un grand nombre de contraintes ou de caractéristiques à prendre en compte. Écrire des règles manuellement est par conséquent une tâche fortement complexe voire impossible. Nous nous sommes donc retournés vers les techniques d'apprentissage automatique pour leur capacité à prendre en compte et combiner de nombreux critères de détection. Parmi ces algorithmes d'apprentissage automatique, le problème des algorithmes supervisés est la labélisation qui a un coût très important car ils demandent des experts en sécurité réseau pour faire l'étiquetage. C'est pourquoi nous avons choisi de travailler avec les algorithmes dits non supervisés.

Dans un premier temps, nous avons étudié les algorithmes d'apprentissage automatique non supervisés qui permettent de trouver des anomalies. En effet, nous avons pris comme hypothèse que le pourcentage du trafic DNS illégitime est très faible par rapport au trafic légitime (par exemple, moins de 2 ou 3 % du trafic). De fait, si le taux d'attaque est important, des méthodes classiques utilisant des seuils en terme de quantité d'octets échangés peuvent les détecter. Nous avons commencé par évaluer quatre algorithmes (Kmeans, GMM, LOF, et DBSCAN) sur le jeu de données BOTS version 1. Les résultats ont montré le potentiel de ces algorithmes dans la détection des attaques complexes, en particulier l'algorithme DBSCAN. Néanmoins, nous avons été confrontés à plusieurs difficultés. Tout d'abord, les performances de DBSCAN dépendent de deux hyperparamètres qui doivent être déterminés expérimentalement. Ensuite, le jeu de données BOTS version 1 est limité (pas de données brutes initiales, nombre de caractéristiques limité).

Dans un deuxième temps, nous avons donc proposé une méthode afin de trouver automatiquement les valeurs optimales des hyper paramètres de DBSCAN. Cette méthode est basée sur le pourcentage d'attaques maximal que nous visons. Nous avons aussi créé un jeu de données qui contient des trafics de tunnel DNS afin d'avoir plus de contrôle et d'assurance sur les tests des algorithmes.

Dans un troisième temps, nous avons évalué notre méthode avec comme objectif de démontrer sa généricité. Pour cela, nous avons effectué des tests à la fois sur le jeu de données que nous avons créé, mais aussi le jeu de données CIRA-CIC-DoHBrw-2020 qui contient des trafics HTTPS normaux, des trafics DoH normaux, et des trafics DoH malveillants pour évaluer notre méthode. Nous avons comparé notre méthode avec cinq autres algorithmes de détection de valeurs aberrantes (Kmeans, GMM, Isolation Forest, One-class SVM, et LOF) sur ces deux jeux de données. Notre méthode donne de bonnes performances de détection en particulier les taux de faux positifs sont très faibles donnant ainsi plus d'assurance sur les alertes qui pourraient être détectées. Lorsque le protocole DNS est encapsulé dans HTTPS, les résultats sont moins bons

mais nous avons pu constater qu'en séparant les trafics selon les navigateurs (ici Chrome et Firefox), les résultats obtenus sont bien meilleurs.

En l'état actuel de nos recherches et au travers des expérimentations effectuées pour évaluer les performances de notre algorithme, nous sommes persuadés que notre algorithme peut aider les experts sécurité à détecter des flux C&C cachés dans des tunnels DNS ou DoH dans le cadre d'analyses différées d'attaques (par exemple, analyses post-mortem ou approches forensiques lors d'enquêtes après incident). Il manque toutefois de valider notre algorithme sur des données du monde réel. Pour cela, une première approche pourrait consister à initier des tunnels DNS dans une infrastructure réseau d'une entreprise pendant une période déterminée.

Par contre, si nous voulons pouvoir utiliser notre approche dans le cadre d'analyses temps réel (approche proactive ou réactive), il est nécessaire d'améliorer les performances pour réduire le temps d'exécution. Pour cela, plusieurs pistes peuvent être envisagées :

- Nos expérimentations ont pu mettre en évidence que One-class SVM est entre trois à quatre fois plus rapide que notre algorithme mais ses capacités à détecter les attaques sont limitées à des pourcentages très faibles (environ 0,5%). Il pourrait être intéressant dans un premier temps de proposer une stratégie combinant notre algorithme avec One-class SVM. Sans arriver à des performances pour faire de l'analyse temps réel, nous pourrions détecter les anomalies plus rapidement et en gardant un bon taux de détection.
- AutoRoC-DBSCAN permet d'optimiser DBSCAN pour calculer les clusters et donc les anomalies n'appartiennent pas aux clusters. Il n'est peut-être pas nécessaire d'exécuter ce processus à chaque fois. Nous pourrions partir d'un ensemble de clusters et anomalies calculés à un instant t faire l'analyse temps réel du trafic réseau en calculant uniquement la distance d'un flux avec les clusters existants du modèle à l'instant t. Les clusters seraient mis à jour régulièrement en relançant AutoRoC-DBSCAN périodiquement pour optimiser le modèle et donc la performance de détection.
- Ainsi, nous visons à développer notre méthode pour répondre aux attaques en temps réel. Lorsqu'il y a un nouveau trafic entrant, nous calculerons la distance entre ce trafic et les clusters créés par DBSCAN pour savoir si ce trafic appartient à un cluster ou non. Si tel est le cas, ce trafic est considéré comme un trafic authentique. Sinon, ce trafic est considéré comme un trafic anormal.
- Enfin, nous avons ciblé la détection de tunnels C&C uniquement sur les protocoles DNS et DoH. D'autres protocoles sont utilisés par les attaquants pour mettre en œuvre des tunnels C&C comme FTP, IMAP/SMTP, IRC, SMB ou encore SSH [105]. Il serait intéressant d'évaluer les performances de notre approche sur ces différents protocoles réseau pour étudier sa généricité.

Acronymes

ANN	Artificial Neural Network
ANSSI	Agence Nationale de la Sécurité des Systèmes d'Information
API	Application Programming Interface
APT(s)	Advanced Persistent Threat(s)
AUC	Area under the Curve
BDE	Binary Differential Evolution
C&C	Command and Control
CERT	Computer Emergency Response Team
CIC	Institut Canadien pour la Cybersécurité
CNN	Convolutional Neural Networks
CSIRT	Computer Security Incident Response Team
DAGMM	Deep Autoencoding Gaussian Mixture Model
DARPA	Defense Advanced Research Projects Agency
DBN	Deep Belief Network
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DGA	Domain Generation Algorithm
DID	Defensive-in-Depth
DMZ	Demilitarized Zone
DNN	Deep Neural Network
DNS	Domain Name System
DoH	DNS over HTTPS
DR	Detection Rate
DSP	Digital Service Providers

EAC	Evidence Accumulation Clustering
ENISA	Agence européenne pour la cybersécurité
FDAGMM	Deep Autoencoding Gaussian Mixture Model Assisté par Federated Learning
FPR	False Positive Rate
FSN	Fournisseurs de Service Numérique
FTP	File Transfer Protocol
GBDT	Gradient Boosting Decision Tree
GCN	Convolutional Network of Graphs
GHSOM	Growing Hierarchical Self-Organizing Maps
GMM	Gaussian Mixture Model
HIDS	Host-based Intrusion Detection System
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HSOM	Hierarchical Self-Organizing Maps
IAM	Identity and Access Management
ID	Identity
IDS	Intrusion Detection System
IoA	Indicator of Attack
IoC	Indicator of Compromise
IP	Internet Protocol
IT	Information Technology
KNN	K-nearest neighbors
LGBM	Light Gradient Boosting Machine
LOF	Local Outlier Factor

LSTM	Long Short-Term Memory
MDE	Maximal Density Estimator
MISP	Malware Information Sharing Platform
MOGA	Multiobjective Genetic Algorithm
NADS	Systèmes de détection d'anomalies de réseau
NB	Naïve Bayes
NDAE	Non-symmetric Deep Auto-Encoder
NIDS	Network Intrusion Detection System
NIS	Network and Information Security
OV	Objectif visé
OSE	Opérateur de service essentiel
PACS	Plan d'amélioration continue de la sécurité
PCA	Plan de continuité d'activité
PCA	Principal Component Analysis
PSSI	Politique de Sécurité des réseaux et Systèmes d'Information
RM	Risk Manager
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic curve
RR	Resource Record
SDN	Software Defined networking
SEM	Security Event Management
SI	Système d'Information
SIE	Système d'Information Essentiel
SIM	Security Information Management
SIEM	Security information and event management

S-NDAE	Random forest - Non-Symmetric Deep Auto-Encoder
SOC	Security Operations Center
SOGA-DBSCAN	Single-objective genetic algorithm-based DBSCAN
SOM	Self-Organizing Map
SR	Sources de Risque
SSC	Sub-Space Clustering
STIX	Structured Threat Information Expression
SVM	Support Vector Machine
TAXII	Trusted Automated Exchange of Intelligence Information
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTPs	Tactics, Techniques and Procedures
UDP	User Datagram Protocol
UE	Union Européenne
UEBA	User Entity Behavior Analytics
UNC	Unsupervised Niche Clustering
UNIDS	Système de détection d'intrusion réseau non supervisé
VPN	Virtual Private Network
WAF	Web Application Firewall
ZTDP	Zero Trust Data Protection
ZTNA	Zero Trust Network Access

Références

[1] "Digital Around the World," DataReportal – Global Digital Insights. Accessed: May 19, 2022. [Online]. Available: https://datareportal.com/global-digital-overview
[2] "Anomali Cybersecurity Insights Report 2022: The State of Cyber Resilience." Accessed: May 19, 2022. [Online]. Available: https://www.anomali.com/resources/whitepapers/anomali-cybersecurity-insights-report
[3] "Check Point Research: Cyber Attacks Increased 50% Year over Year," Check Point Software. Accessed: May 19, 2022. [Online]. Available: https://blog.checkpoint.com/2022/01/10/check-point-research-cyber-attacks-increased-50-year-over-year/
[4] "What is an Advanced Persistent Threat (APT)?" Accessed: Apr. 30, 2022. [Online]. Available: https://www.varonis.com/blog/advanced-persistent-threat
[5] "Matrix - Enterprise | MITRE ATT&CK®." Accessed: Jun. 12, 2020. [Online]. Available: https://attack.mitre.org/matrices/enterprise/
[6] "APT32, SeaLotus, OceanLotus, APT-C-00, Group G0050 | MITRE ATT&CK®." Accessed: Jun. 12, 2020. [Online]. Available: https://attack.mitre.org/groups/G0050/
[7] "Cobalt Strike | Adversary Simulation and Red Team Operations," Cobalt Strike. Accessed: Nov. 21, 2023. [Online]. Available: https://www.cobaltstrike.com
[8] P. E. Hoffman and P. McManus, "DNS Queries over HTTPS (DoH)," Internet Engineering Task Force, Request for Comments RFC 8484, Oct. 2018. doi: 10.17487/RFC8484.
[9] "What is SIEM and how does it work?," FireEye. Accessed: Aug. 24, 2021. [Online]. Available: https://www.fireeye.com/products/helix/what-is-siem-and-how-does-it-work.html
[10] "Qu'est-ce le C&C ? Explications sur l'infrastructure de Commande et Contrôle | Varonis." Accessed: Mar. 06, 2023. [Online]. Available: https://www.varonis.com/fr/blog/quest-ce-que-le-cc
[11] "New Wekby Attacks Use DNS Requests As Command and Control Mechanism," Unit42. Accessed: Sep. 07, 2021. [Online]. Available: https://unit42.paloaltonetworks.com/unit42-new-wekby-attacks-use-dns-requests-as-command-and-control-mechanism/
[12] Ron, "Introduction." Aug. 24, 2021. Accessed: Aug. 25, 2021. [Online]. Available: https://github.com/iagox86/dnscat2
[13] "dns2tcp | Kali Linux Tools," Kali Linux. Accessed: Feb. 07, 2022. [Online]. Available: https://www.kali.org/tools/dns2tcp/
[14] "kryo.se: iodine (IP-over-DNS, IPv4 over DNS tunnel)." Accessed: Aug. 25, 2021. [Online]. Available: https://code.kryo.se/iodine/
[15] "APT - Ce qu'il faut savoir sur les menaces persistantes avancées." Accessed: Apr. 30, 2022. [Online]. Available: https://www.varonis.com/fr/blog/apt-ce-quil-faut-savoir-sur-les-menaces-persistantes-avancees
[16] "Qu'est-ce Qu'une Menace Persistante Avancée (APT) ? | CrowdStrike," crowdstrike.fr. Accessed: Feb. 14, 2023. [Online]. Available: https://www.crowdstrike.fr/cybersecurity-101/advanced-persistent-threat-apt/
[17] "Threat Group-3390, Earth Smilodon, TG-3390, Emissary Panda, BRONZE UNION, APT27, Iron Tiger, LuckyMouse, Group G0027 | MITRE ATT&CK®." Accessed: Jan. 01, 2024. [Online]. Available: https://attack.mitre.org/groups/G0027/

[18] “APT28, IRON TWILIGHT, SNAKEMACKEREL, Swallowtail, Group 74, Sednit, Sofacy, Pawn Storm, Fancy Bear, STRONTIUM, Tsar Team, Threat Group-4127, TG-4127, Group G0007 | MITRE ATT&CK®.” Accessed: Jan. 01, 2024. [Online]. Available: https://attack.mitre.org/groups/G0007/
[19] “OilRig, COBALT GYPSY, IRN2, APT34, Helix Kitten, Evasive Serpens, Group G0049 | MITRE ATT&CK®.” Accessed: Jan. 01, 2024. [Online]. Available: https://attack.mitre.org/groups/G0049/
[20] “APT trends report Q3 2022.” Accessed: Apr. 04, 2023. [Online]. Available: https://securelist.com/apt-trends-report-q3-2022/107787/
[21] “Cybersécurité: une nouvelle loi pour renforcer la résilience européenne | Actualité | Parlement européen.” Accessed: Jan. 09, 2023. [Online]. Available: https://www.europarl.europa.eu/news/fr/press-room/20221107IPR49608/cybersecurite-une-nouvelle-loi-pour-renforcer-la-resilience-europeenne
[22] “Orange Cyberdefense: Directive NIS 2/2 : quels impacts pour les entreprises ?” Accessed: Sep. 05, 2022. [Online]. Available: https://www.orangecyberdefense.com/fr/insights/blog/reglementation-de-la-cyber/directive-nis-2-2-quels-impacts-pour-les-entreprises
[23] “[Infographie] les 23 règles de la directive NIS,” MANIKA. Accessed: Sep. 29, 2022. [Online]. Available: https://www.manika-consulting.com/infographie-les-23-regles-de-la-directive-nis/
[24] “Publication : La méthode EBIOS Risk Manager – Le guide,” ANSSI. Accessed: Jan. 12, 2023. [Online]. Available: https://www.ssi.gouv.fr/guide/la-methode-ebios-risk-manager-le-guide/
[25] “Le modèle Zero Trust,” ANSSI. Accessed: Feb. 20, 2023. [Online]. Available: https://www.ssi.gouv.fr/agence/publication/le-modele-zero-trust/
[26] “What is Zero Trust Security?,” Netskope. Accessed: Feb. 20, 2023. [Online]. Available: https://www.netskope.com/security-defined/what-is-zero-trust
[27] “ZTNA | Zero Trust Network Access,” Netskope. Accessed: Feb. 20, 2023. [Online]. Available: https://www.netskope.com/platform/zero-trust-network-access
[28] “What is Zero Trust Data Protection?,” Netskope. Accessed: Feb. 20, 2023. [Online]. Available: https://www.netskope.com/security-defined/zero-trust-data-protection
[29] “What is Defense in Depth | Benefits of Layered Security | Imperva,” Learning Center. Accessed: Feb. 21, 2023. [Online]. Available: https://www.imperva.com/learn/application-security/defense-in-depth/
[30] “Qu’est-ce qu’un pare-feu ?” Accessed: Feb. 21, 2023. [Online]. Available: https://blogs.oracle.com/oracle-france/post/quest-ce-quun-pare-feu
[31] “Qu’est-ce qu’un pare-feu ? Un guide de démarrage pour les différents types de pare-feu et pour savoir si vous en avez besoin,” Kinsta®. Accessed: May 21, 2023. [Online]. Available: https://kinsta.com/fr/blog/qu-est-ce-qu-un-pare-feu/
[32] “What is WAF | Types, Security & Features Explained | Imperva,” Learning Center. Accessed: May 22, 2023. [Online]. Available: https://www.imperva.com/learn/application-security/what-is-web-application-firewall-waf/
[33] “Qu’est-ce qu’un VPN et pourquoi est-ce essentiel en 2023.” Accessed: Feb. 21, 2023. [Online]. Available: https://fr.vpn-mentors.com/popular/le-b-ba-le-guide-vpn-de-vpnmentor-pour-debutants/?keyword=vpn%20c%20est%20quoi&geo=9055236&device=&ad=545184967834&cq_src=google_ads&cq_cmp=270495911&cq_term=vpn%20c%20est

%20quoi&cq_plac=&cq_net=g&cq_plt=gp&gclid=CjwKCAiA9NGfBhBvEiwAq5vSy0yiH_CCVYVcbVycxycRBeo69BoJw9slQyXvib-RLKgHhSUZE2QhNhoCsZwQAvD_BwE
[34] "Types de VPN - VPN One Click." Accessed: May 23, 2023. [Online]. Available: https://www.vpnoneclick.com/fr/types-de-vpn/
[35] "Qu'est-ce que la gestion des identités et des accès (IAM)?," Cisco. Accessed: Feb. 23, 2023. [Online]. Available: https://www.cisco.com/c/fr_ca/products/security/identity-services-engine/what-is-identity-access-management.html
[36] "Intrusion Detection System (IDS)," GeeksforGeeks. Accessed: May 23, 2023. [Online]. Available: https://www.geeksforgeeks.org/intrusion-detection-system-ids/
[37] "Difference between HIDs and NIDs," GeeksforGeeks. Accessed: Sep. 06, 2022. [Online]. Available: https://www.geeksforgeeks.org/difference-between-hids-and-nids/
[38] "What is UEBA? Definition and use," FireEye. Accessed: Sep. 07, 2022. [Online]. Available: https://www.fireeye.fr/products/helix/what-is-ueba.html
[39] "What is User Entity and Behavior Analytics (UEBA)?," Fortinet. Accessed: May 24, 2023. [Online]. Available: https://www.fortinet.com/resources/cyberglossary/what-is-ueba
[40] "What is a Honeypot? How It Can Trap Cyberattackers | CrowdStrike," crowdstrike.com. Accessed: Oct. 04, 2022. [Online]. Available: https://www.crowdstrike.com/cybersecurity-101/honeypots-in-cybersecurity-explained/
[41] "What Is SIEM? | Security Information and Event Management | Trellix." Accessed: May 24, 2023. [Online]. Available: https://www.trellix.com/en-us/security-awareness/operations/what-is-siem.html
[42] "C'est quoi un outil SIEM (security information event management)?," Splunk. Accessed: May 24, 2023. [Online]. Available: https://www.splunk.com/fr_fr/data-insider/what-is-siem.html
[43] "Un CSIRT, à quoi ça CERT ? – CYBER-SECURITE.FR." Accessed: Sep. 07, 2022. [Online]. Available: https://www.cyber-securite.fr/2013/12/13/un-csirt-a-quoi-ca-cert/
[44] "What is SOC (Security Operation Center)?," Check Point Software. Accessed: Sep. 06, 2022. [Online]. Available: https://www.checkpoint.com/cyber-hub/threat-prevention/what-is-soc/
[45] Snoweb, "Quel processus de gestion de crise pour une cyberattaque ?" Accessed: Jun. 20, 2023. [Online]. Available: https://www.c-risk.com/fr/blog/gestion-de-crise/
[46] "Publication : Recommandations relatives à l'interconnexion d'un système d'information à Internet," ANSSI. Accessed: Jun. 13, 2023. [Online]. Available: https://www.ssi.gouv.fr/guide/definition-dune-architecture-de-passerelle-dinterconnexion-securisee/
[47] X. Jin and J. Han, "K-Means Clustering," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds., Boston, MA: Springer US, 2010, pp. 563–564. doi: 10.1007/978-0-387-30164-8_425.
[48] D. Reynolds, "Gaussian Mixture Models," in *Encyclopedia of Biometrics*, S. Z. Li and A. Jain, Eds., Boston, MA: Springer US, 2009, pp. 659–663. doi: 10.1007/978-0-387-73003-5_196.
[49] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data Via the *EM* Algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, Sep. 1977, doi: 10.1111/j.2517-6161.1977.tb01600.x.

[50] C. Maklin, “Gaussian Mixture Models Clustering Algorithm Explained,” Medium. Accessed: Jun. 12, 2020. [Online]. Available: https://towardsdatascience.com/gaussian-mixture-models-d13a5e915c8e
[51] F. T. Liu, K. Ting, and Z.-H. Zhou, “Isolation Forest,” Jan. 2009, pp. 413–422. doi: 10.1109/ICDM.2008.17.
[52] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, “Support Vector Method for Novelty Detection,” presented at the NIPS, Jan. 1999, pp. 582–588.
[53] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: Identifying Density-Based Local Outliers,” p. 12.
[54] M. Ester, H.-P. Kriegel, and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” p. 6.
[55] M. A. Kramer, “Nonlinear principal component analysis using autoassociative neural networks,” *AIChE Journal*, vol. 37, no. 2, pp. 233–243, 1991, doi: 10.1002/aic.690370209.
[56] G. E. Hinton and R. R. Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks,” *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006, doi: 10.1126/science.1127647.
[57] N. S. Chauhan, “Introduction to AutoEncoder and Variational AutoEncoder(VAE),” The AI dream. Accessed: Sep. 13, 2023. [Online]. Available: https://www.theaidream.com/post/an-introduction-to-autoencoder-and-variational-autoencoder-vae
[58] K. Shiruru, “AN INTRODUCTION TO ARTIFICIAL NEURAL NETWORK,” *International Journal of Advance Research and Innovative Ideas in Education*, vol. 1, pp. 27–30, Sep. 2016.
[59] A. Dertat, “Applied Deep Learning - Part 1: Artificial Neural Networks,” Medium. Accessed: Sep. 13, 2023. [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6
[60] T. Kohonen, *Self-Organizing Maps*, vol. 30. in Springer Series in Information Sciences, vol. 30. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995. doi: 10.1007/978-3-642-97610-0.
[61] T. Kohonen, “Self-organized formation of topologically correct feature maps,” *Biol. Cybern.*, vol. 43, no. 1, pp. 59–69, Jan. 1982, doi: 10.1007/BF00337288.
[62] A. Ali, “Self Organizing Map(SOM),” The Art of Data Scicne. Accessed: Apr. 18, 2023. [Online]. Available: https://medium.com/machine-learning-researcher/self-organizing-map-som-c296561e2117
[63] “Introduction to Self-Organizing Maps,” Rubik’s Code. Accessed: Jul. 11, 2023. [Online]. Available: https://rubikscode.net/2018/08/20/introduction-to-self-organizing-maps/
[64] Y. Kazato, Y. Nakagawa, and Y. Nakatani, “Improving Maliciousness Estimation of Indicator of Compromise Using Graph Convolutional Networks,” in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, USA: IEEE, Jan. 2020, pp. 1–7. doi: 10.1109/CCNC46108.2020.9045113.
[65] Y. Chen, J. Zhang, and C. K. Yeo, “Network Anomaly Detection Using Federated Deep Autoencoding Gaussian Mixture Model,” in *Machine Learning for Networking*, vol. 12081, S. Boumerdassi, É. Renault, and P. Mühlethaler, Eds., in Lecture Notes in Computer Science, vol. 12081. , Cham: Springer International Publishing, 2020, pp. 1–14. doi: 10.1007/978-3-030-45778-5_1.
[66] M. Perera Jayasuriya Kuranage, K. Piamrat, and S. Hamma, “Network Traffic Classification Using Machine Learning for Software Defined Networks,” in *Machine Learning for Networking*, vol. 12081, S. Boumerdassi, É. Renault, and P.

Mühlethaler, Eds., in Lecture Notes in Computer Science, vol. 12081. , Cham: Springer International Publishing, 2020, pp. 28–39. doi: 10.1007/978-3-030-45778-5_3.
[67] R. Aliakbarisani, A. Ghasemi, and S. Felix Wu, "A data-driven metric learning-based scheme for unsupervised network anomaly detection," *Computers & Electrical Engineering*, vol. 73, pp. 71–83, Jan. 2019, doi: 10.1016/j.compeleceng.2018.11.003.
[68] "Traffic Data from Kyoto University's Honeypots." Accessed: Sep. 20, 2020. [Online]. Available: http://www.takakura.com/Kyoto_data/
[69] "NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB." Accessed: Sep. 20, 2020. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html
[70] Md. S. Alam *et al.*, "Memristor Based Autoencoder for Unsupervised Real-Time Network Intrusion and Anomaly Detection," in *Proceedings of the International Conference on Neuromorphic Systems*, Knoxville TN USA: ACM, Jul. 2019, pp. 1–8. doi: 10.1145/3354265.3354267.
[71] X. Qu *et al.*, "A Survey on the Development of Self-Organizing Maps for Unsupervised Intrusion Detection," *Mobile Netw Appl*, Oct. 2019, doi: 10.1007/s11036-019-01353-0.
[72] R. S. M. Carrasco and M.-A. Sicilia, "Unsupervised intrusion detection through skip-gram models of network behavior," *Computers & Security*, vol. 78, pp. 187–197, Sep. 2018, doi: 10.1016/j.cose.2018.07.003.
[73] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018, doi: 10.1109/TETCI.2017.2772792.
[74] P. V. Amoli, T. Hamalainen, G. David, and M. Zolotukhin, "Unsupervised Network Intrusion Detection Systems for Zero-Day Fast- Spreading Attacks and Botnets," p. 13.
[75] "1999 DARPA Intrusion Detection Evaluation Dataset | MIT Lincoln Laboratory." Accessed: Sep. 18, 2020. [Online]. Available: https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset
[76] P. Casas, J. Mazel, and P. Owezarski, "Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge," *Computer Communications*, vol. 35, no. 7, pp. 772–783, Apr. 2012, doi: 10.1016/j.comcom.2012.01.016.
[77] S. Zanero and G. Serazzi, "Unsupervised learning algorithms for intrusion detection," in *NOMS 2008 - 2008 IEEE Network Operations and Management Symposium*, Salvador, Bahia, Brazil: IEEE, 2008, pp. 1043–1048. doi: 10.1109/NOMS.2008.4575276.
[78] H. Gunes Kayacik, A. Nur Zincir-Heywood, and M. I. Heywood, "A hierarchical SOM-based intrusion detection system," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 4, pp. 439–451, Jun. 2007, doi: 10.1016/j.engappai.2006.09.005.
[79] W. Wang and R. Battiti, "IDENTIFYING INTRUSIONS IN COMPUTER NETWORKS BASED ON PRINCIPAL COMPONENT ANALYSIS," p. 16.
[80] J. Zhang and M. Zulkernine, "Anomaly Based Network Intrusion Detection with Unsupervised Outlier Detection," in *2006 IEEE International Conference on Communications*, Istanbul: IEEE, 2006, pp. 2388–2393. doi: 10.1109/ICC.2006.255127.
[81] E. Leon, O. Nasraoui, and J. Gomez, "Anomaly detection based on unsupervised niche clustering with application to network intrusion detection," in *Proceedings of*

the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753), Portland, OR, USA: IEEE, 2004, pp. 502–508. doi: 10.1109/CEC.2004.1330898.
[82] "KDD Cup 1999 Data." Accessed: Sep. 18, 2020. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[83] J. Miglani and C. Thorpe, "Employing Machine Learning Paradigms for detecting DNS Tunnelling," avril 2021.
[84] M. Singh, M. Singh, and S. Kaur, "10 Days DNS Network Traffic from April-May, 2016," vol. 2, May 2019, doi: 10.17632/zh3wnddzxy.2.
[85] F. Palau, C. Catania, J. Guerra, S. Garcia, and M. Rigaki, "DNS Tunneling: A Deep Learning based Lexicographical Detection Approach," *arXiv:2006.06122 [cs]*, Jun. 2020, Accessed: Aug. 24, 2021. [Online]. Available: http://arxiv.org/abs/2006.06122
[86] M. MontazeriShatoori, L. Davidson, G. Kaur, and A. H. Lashkari, "Detection of DoH Tunnels using Time-series Classification of Encrypted Traffic," in *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, Aug. 2020, pp. 63–70. doi: 10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00026.
[87] Y. Banadaki and S. Robert, "Detecting Malicious DNS over HTTPS Traffic in Domain Name System using Machine Learning Classifiers," *Journal of Computer Sciences and Applications*, vol. 8, pp. 46–55, Aug. 2020, doi: 10.12691/jcsa-8-2-2.
[88] S. K. Singh and P. K. Roy, "Detecting Malicious DNS over HTTPS Traffic Using Machine Learning," in *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, Dec. 2020, pp. 1–6. doi: 10.1109/3ICT51146.2020.9312004.
[89] H. Lin, G. Liu, and Z. Yan, "Detection of Application-Layer Tunnels with Rules and Machine Learning," in *Security, Privacy, and Anonymity in Computation, Communication, and Storage*, vol. 11611, G. Wang, J. Feng, M. Z. A. Bhuiyan, and R. Lu, Eds., in Lecture Notes in Computer Science, vol. 11611. , Cham: Springer International Publishing, 2019, pp. 441–455. doi: 10.1007/978-3-030-24907-6_33.
[90] A. Berg and D. Forsberg, "Identifying DNS-tunneled traffic with predictive models," *arXiv:1906.11246 [cs]*, Jun. 2019, Accessed: Jun. 09, 2020. [Online]. Available: http://arxiv.org/abs/1906.11246
[91] A. Almusawi and H. Amintoosi, "DNS Tunneling Detection Method Based on Multilabel Support Vector Machine," *Security and Communication Networks*, vol. 2018, pp. 1–9, 2018, doi: 10.1155/2018/6137098.
[92] I. Homem and P. Papapetrou, "HARNESSING PREDICTIVE MODELS FOR ASSISTING NETWORK FORENSIC INVESTIGATIONS OF DNS TUNNELS," p. 12, 2017.
[93] V. T. Do, P. Engelstad, B. Feng, and T. van Do, "Detection of DNS Tunneling in Mobile Networks Using Machine Learning," in *Information Science and Applications 2017*, vol. 424, K. Kim and N. Joukov, Eds., in Lecture Notes in Electrical Engineering, vol. 424. , Singapore: Springer Singapore, 2017, pp. 221–230. doi: 10.1007/978-981-10-4154-9_26.
[94] A. L. Buczak, P. A. Hanke, G. J. Cancro, M. K. Toma, L. A. Watkins, and J. S. Chavis, "Detection of Tunnels in PCAP Data by Random Forests," in *Proceedings of the 11th Annual Cyber and Information Security Research Conference*, Oak Ridge TN USA: ACM, Apr. 2016, pp. 1–4. doi: 10.1145/2897795.2897804.
[95] M. Aiello, M. Mongelli, and G. Papaleo, "DNS tunneling detection through statistical fingerprints of protocol messages and machine learning: DNS

TUNNELING DETECTION," *Int. J. Commun. Syst.*, vol. 28, no. 14, pp. 1987–2002, Sep. 2015, doi: 10.1002/dac.2836.
[96] "Applications | Research | Canadian Institute for Cybersecurity | UNB." Accessed: Aug. 24, 2021. [Online]. Available: https://www.unb.ca/cic/research/applications.html
[97] T. Q. Nguyen, R. Laborde, A. Benzekri, and B. Qu'hen, "Detecting abnormal DNS traffic using unsupervised machine learning," in *2020 4th Cyber Security in Networking Conference (CSNet)*, Oct. 2020, pp. 1–8. doi: 10.1109/CSNet50428.2020.9265466.
[98] P. Cunningham and S. Delany, "k-Nearest neighbour classifiers," *Mult Classif Syst*, vol. 54, avril 2007, doi: 10.1145/3459665.
[99] "scipy.signal.find_peaks — SciPy v1.8.0 Manual." Accessed: Feb. 08, 2022. [Online]. Available: https://docs.scipy.org/doc/scipy-1.8.0/html-scipyorg/reference/generated/scipy.signal.find_peaks.html#scipy.signal.find_peaks
[100] A. Starczewski, P. Goetzen, and M. J. Er, "A New Method for Automatic Determining of the DBSCAN Parameters," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 10, no. 3, pp. 209–221, Jul. 2020, doi: 10.2478/jaiscr-2020-0014.
[101] Z. Falahiazar, A. Bagheri, and M. Reshadi, "Determining the Parameters of DBSCAN Automatically Using the Multi-Objective Genetic Algorithm," *J. Inf. Sci. Eng.*, 2021.
[102] A. Karami and R. Johansson, "Choosing DBSCAN Parameters Automatically using Differential Evolution," *IJCA*, vol. 91, no. 7, pp. 1–11, Apr. 2014, doi: 10.5120/15890-5059.
[103] "Application Layer Protocol: DNS, Sub-technique T1071.004 - Enterprise | MITRE ATT&CK®." Accessed: Jun. 06, 2023. [Online]. Available: https://attack.mitre.org/techniques/T1071/004/
[104] "Protocol Tunneling, Technique T1572 - Enterprise | MITRE ATT&CK®." Accessed: Jun. 06, 2023. [Online]. Available: https://attack.mitre.org/techniques/T1572/
[105] "Command and Control, Tactic TA0011 - Enterprise | MITRE ATT&CK®." Accessed: Sep. 12, 2023. [Online]. Available: https://attack.mitre.org/tactics/TA0011/

Annexes

Tableau 16. Caratéristiques du jeu de données

	Caractéristiques	Description
1	src_port	Port source
2	dst_port	Port de destination
3	flow_duration	Durée du flux
4	flow_byts_s	Nombre d'octets de flux par seconde
5	flow_pkts_s	Nombre de paquets de flux par seconde
6	fwd_pkts_s	Nombre de paquets envoyés par seconde
7	bwd_pkts_s	Nombre de paquets reçus par seconde
8	tot_fwd_pkts	Nombre total de paquets dans le sens envoyer
9	tot_bwd_pkts	Nombre total de paquets dans le sens recevoir
10	bytes	Taille totale du paquet
11	bytes_sent	Taille totale du paquet dans le sens envoyer
12	bytes_receive	Taille totale du paquet en sens recevoir
13	fwd_pkt_len_mean	Taille moyenne du paquet dans le sens envoyer
14	fwd_pkt_len_std	Taille de l'écart standard du paquet dans le sens envoyer
15	bwd_pkt_len_mean	Taille moyenne du paquet dans le sens recevoir
16	bwd_pkt_len_std	Taille de l'écart standard du paquet dans le sens recevoir
17	pkt_len_mean	Longueur moyenne d'un paquet
18	pkt_len_std	Longueur d'écart standard d'un paquet
19	pkt_len_var	Longueur de variance d'un paquet
20	fwd_header_len	Nombre total d'octets utilisés pour les en-têtes dans le sens envoyer
21	bwd_header_len	Nombre total d'octets utilisés pour les en-têtes dans le sens recevoir

22	fwd_act_data_pkts	Nombre de paquets avec au moins 1 octet de charge utile de données TCP dans le sens envoyer
23	flow_iat_mean	Temps moyen entre deux paquets envoyés dans le flux
24	flow_iat_std	Temps d'écart standard entre deux paquets envoyés dans le flux
25	fwd_iat_tot	Temps total entre deux paquets envoyés dans le sens envoyer
26	fwd_iat_mean	Temps moyen entre deux paquets envoyés dans le sens envoyer
27	fwd_iat_std	Temps d'écart standard entre deux paquets envoyés dans le sens envoyer
28	bwd_iat_tot	Temps total entre deux paquets envoyés dans le sens recevoir
29	bwd_iat_mean	Temps moyen entre deux paquets envoyés dans le sens recevoir
30	bwd_iat_std	Temps d'écart standard entre deux paquets envoyés dans le sens recevoir
31	pkt_size_avg	Taille moyenne du paquet
32	active_mean	Temps moyen pendant lequel un flux était actif avant de devenir inactif
33	active_std	Temps d'écart standard pendant lequel un flux était actif avant de devenir inactif
34	idle_mean	Temps moyen d'inactivité d'un flux avant de devenir actif
35	idle_std	Temps d'écart standard pendant lequel un flux était inactif avant de devenir actif
36	hour	Heure à laquelle le trafic se produit
37	minute	Minute où le trafic se produit
38	weekno	Jour de la semaine où le trafic se produit
39	label	Défini normal ou attaque