RÉPUBLIQUE DU BÉNIN
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ D'ABOMEY-CALAVI

INSTITUT DE FORMATION ET DE RECHERCHE EN INFORMATIQUE

01 BP 526 Cotonou - Bénin
Tel : +229 01 21 14 19 88
https://www.ifri-uac.bj Courriel : contact@ifri.uac.bj

MÉMOIRE

pour l'obtention du

Diplôme de Licence en Informatique

Option : Sécurité Informatique

Présenté par :
Clarence Yannis **ADDA**

Détection d'attaques par déni de service basée sur l'apprentissage automatique

Sous la supervision de :
Dr (MA) Vinasétan Ratheil **HOUNDJI**

Membres du jury :

Max Fréjus Owolabi **SANYA**	Docteur (MA)	IFRI-UAC	Président
Rose **GOHOUE**	Ingénieur	IFRI-UAC	Examinateur
Vinasétan Ratheil **HOUNDJI**	Docteur (MA)	IFRI-UAC	Rapporteur

Année Académique : **2022 - 2023**

Sommaire

Dédicace ii

Remerciements iii

Résumé iv

Abstract v

Table des figures vi

Liste des tableaux vii

Liste des acronymes viii

Introduction 1

1 Etat de l'art 3

2 Analyse et choix techniques 14

3 Résultats et Discussions 26

Conclusion 29

Bibliographie 30

Webographie 31

Table des matières 32

Dédicace

À

Mon père **François ADDA**

Ma mère **Reine DAN**

Mon frère **Ariel Crédo ADDA**

Remerciements

Nous exprimons notre profonde gratitude à Dieu, le Père Tout-Puissant, pour nous avoir accordé la santé, la force et la persévérance nécessaires à l'accomplissement de ce travail. À travers ce mémoire, nous tenons à adresser nos sincères remerciements à :

- l'administration de l'Institut de Formation et de Recherche en Informatique (IFRI), en particulier son directeur, le Professeur Eugène C. EZIN, son directeur adjoint, le Professeur Gaston EDAH, ainsi que l'ensemble des enseignants qui ont contribué à notre formation ;
- Dr (MA) V. Ratheil HOUNDJI, notre encadreur, pour sa disponibilité, son accompagnement et son soutien tout au long de la réalisation de ce mémoire ;
- nos familles et amis, pour leur soutien moral et leur encouragement constant tout au long de notre parcours académique.

Que tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce travail trouvent ici l'expression de notre profonde reconnaissance.

Résumé

Ce mémoire présente une étude sur l'utilisation de modèles d'apprentissage automatique pour la détection des attaques par déni de service (DoS), un enjeu majeur pour la sécurité des systèmes informatiques dans un contexte où ce type d'attaque devient de plus en plus fréquent. L'accent est mis sur l'évaluation des performances de différents algorithmes de classification. Le jeu de données NSL-KDD a été utilisé pour l'entraînement et l'évaluation des modèles. Trois algorithmes populaires ont été testés, à savoir : les forêts aléatoires (RF), les arbres de décision (DT) et les k-plus proches voisins (KNN). Les résultats obtenus montrent que l'algorithme RF a atteint une précision avoisinant les 99%, surpassant les autres modèles en termes de performance. Ce travail met ainsi en lumière l'efficacité des techniques d'apprentissage automatique dans la détection des attaques DoS.

Mots clés : apprentissage automatique, déni de service, algorithmes de classification, NSL-KDD, forêts aléatoires

Abstract

This thesis presents a study on the use of machine learning models for detecting Denial of Service (DoS) attacks, a critical area for ensuring the security of information systems in a context where such attacks are increasingly frequent. The focus is on evaluating the performance of classification algorithms. The NSL-KDD dataset was used for training and evaluating the models. Three popular algorithms were tested : Random Forest (RF), Decision Trees (DT), and k-Nearest Neighbors (KNN). The results show that the RF algorithm achieved an accuracy close to 99%, outperforming the other models in terms of performance. This work thus highlights the effectiveness of machine learning techniques in detecting DoS attacks.

Key words: machine learning, denial of service, classification algorithms, NSL-KDD, random forest

Table des figures

1.1 Illustration d'une attaque par déni de service (DoS) [4] 3
1.2 Illustration d'une attaque par déni de service distribué (DDoS) [4] 4
1.3 Illustration d'une attaque par inondation UDP [5] 5
1.4 Établissement d'une connexion TCP standard à l'aide du processus de poignée de main en trois étapes (Three-Way Handshake) [5] 5
1.5 Illustration d'une attaque SYN Flood [5] 6
1.6 Illustration d'une attaque par amplification DNS [5] 6
1.7 Illustration d'une attaque par réflexion DNS [5] 7
1.8 Établissement d'une connexion HTTP standard [5] 7
1.9 Illustration d'une attaque Slowloris [5] 8

2.1 Répartition des enregistrements dans le jeu de données NSL-KDD selon les deux classes 16
2.2 Répartition des enregistrements par classe dans les ensembles d'entraînement et de test du jeu de données NSL-KDD 17

Liste des tableaux

1.1 Exemples d'applications de l'apprentissage automatique en cybersécurité 10

2.1 Répartition des types d'attaques selon les classes d'attaques dans l'ensemble de données NSL-KDD 16
2.2 Répartition des classes dans l'ensemble d'entraînement NSL-KDD 17
2.3 Répartition des classes dans l'ensemble de test NSL-KDD 17
2.4 Les attributs du jeu de données NSL-KDD 17
2.5 Trie des attributs du plus important au moins important basée sur Random Forest . . 21
2.6 Trie des attributs du plus important au moins important basée sur le Gain d'Information 22
2.7 Les critères fondamentaux 24
2.8 Comparaison des performances des algorithmes sur ces critères 24

3.1 Résultats des prédictions avant sur-échantillonage 27
3.2 Résultats des prédictions après sur-échantillonage 27

Liste des acronymes

DDoS :

Distributed Denial of Service vi, 4

DNS :

Domain Name System vi, 6, 7

DoS :

Denial of Service iv–vi, 1–4, 15, 16, 26, 28

DT :

Decision Tree iv, v, 23–25

GPU :

Graphics Processing Unit 15

HTTP :

Hypertext Transfer Protocol vi, 7

HTTPS :

Hypertext Transfer Protocol Secure 7

IDPS :

Intrusion Detection and Prevention System 1, 2

IP :

Internet Protocol 5, 8

KNN :

K-Nearest Neighbors iv, v, 13, 23–25

ML :

Machine Learning 1, 9, 11, 14

RF :

Random Forest iv, v, 23–25

SVM :

Support Vector Machine 13, 23

TCP :

Transmission Control Protocol vi, 5

UDP :

User Datagram Protocol vi, 5

Introduction Générale

L'intelligence artificielle (IA) est un domaine qui connaît un essor fulgurant ces dernières années. Ce terme désigne des systèmes intelligents capables d'exécuter des tâches complexes. L'un des principaux atouts de l'IA réside dans sa capacité à agir de manière autonome, dans une certaine mesure. Le sous-domaine de l'IA centré sur l'imitation du comportement humain est appelé apprentissage automatique (ou *machine learning*, ML). L'apprentissage automatique peut être réalisé selon différentes approches, supervisées ou non supervisées. En définitive, il s'agit d'entraîner un programme à partir de données afin qu'il puisse prendre des décisions fondées sur cet apprentissage. Aujourd'hui, l'IA est utilisée pour améliorer de nombreux aspects de la société et résoudre des problèmes variés, notamment dans le domaine de la cybersécurité.

Contexte

Avec l'importance économique croissante d'Internet, une menace grandissante liée aux activités criminelles émerge. En plus des attaques déjà répandues telles que le phishing ou les malwares ciblant les utilisateurs finaux, les cybercriminels s'attaquent désormais à la disponibilité des services en ligne. Le déni de service (*Denial of Service* ou DoS) est l'un des types d'attaques les plus fréquents dans cette catégorie. Il peut consister à lancer des opérations volontairement très coûteuses en ressources ou, dans le cadre des communications Internet, à saturer la capacité du réseau en générant artificiellement un trafic important. Face à la montée en puissance et à la complexité croissante de ces attaques, les systèmes de surveillance réseau peuvent tirer parti de l'apprentissage automatique pour détecter et prévenir les attaques DoS avec davantage d'efficacité.

Problématique

Pour détecter et se protéger contre les cyberattaques, les entreprises s'appuient sur une combinaison de systèmes de prévention et de détection des intrusions (IDPS) ainsi que de pare-feux. Les IDPS sont généralement basés sur des règles et ont pour rôle d'émettre des alertes ou de bloquer le trafic malveillant. Les pare-feux fonctionnent de manière similaire, mais se concentrent spécifiquement sur le contrôle du trafic entrant et sortant du réseau. Ces mécanismes, bien que performants face à des attaques bien définies, montrent leurs limites lorsqu'ils sont confrontés à des menaces plus sophistiquées. En particulier, les attaques par déni de service (DoS) ont évolué de manière à imiter le

comportement du trafic légitime, rendant leur détection beaucoup plus difficile avec des approches traditionnelles basées sur des règles.
En quoi l'utilisation de l'apprentissage automatique peut-elle améliorer la détection des attaques par déni de service, afin de permettre à des systèmes de protection tels que les pare-feux ou les IDPS de bloquer efficacement les requêtes malveillantes et de garantir ainsi une meilleure disponibilité des services ?

Objectifs

L'objectif principal de ce travail est de proposer un modèle capable de détecter les attaques par déni de service (DoS) en classifiant les flux réseau à l'aide de techniques d'apprentissage automatique. Pour y parvenir, les objectifs spécifiques suivants ont été définis :

- Réaliser une revue de littérature sur les attaques DoS, les méthodes de détection de ces attaques, ainsi que l'utilisation de l'apprentissage automatique en cybersécurité ;
- Identifier les outils et environnements adaptés à chaque étape du processus de conception et d'entraînement du modèle ;
- Concevoir et entraîner un modèle capable de classifier les flux réseau à l'aide d'algorithmes d'apprentissage automatique.

Organisation du mémoire

Ce mémoire est structuré en trois (03) chapitres principaux :

- Le premier chapitre présente une revue de littérature sur les différents types d'attaques par déni de service, les approches existantes pour leur détection, et introduit les concepts fondamentaux de l'intelligence artificielle appliquée à la cybersécurité ;
- Le deuxième chapitre décrit la méthodologie adoptée pour la mise en œuvre de la solution proposée, en détaillant les choix techniques et les étapes de développement du modèle ;
- Le troisième chapitre expose les résultats expérimentaux issus des tests réalisés, suivis d'une discussion critique sur les performances obtenues.

Chapitre 1

Etat de l'art

Introduction

Dans ce chapitre, nous commençons par une étude des concepts fondamentaux liés aux attaques par déni de service, dans le but de mieux cerner la nature du problème à résoudre. Dans un second temps, nous analysons les différentes méthodes existantes de détection, d'atténuation et de prévention de ces attaques, afin d'enrichir notre compréhension du sujet et de guider nos choix techniques pour la suite du travail.

1.1 Attaques par déni de service

1.1.1 Qu'est-ce qu'une attaque par déni de service et comment fonctionne t-elle ?

Une attaque par déni de service (ou DoS) (cf. figure 1.1) est une forme de cyberattaque visant à rendre une machine ou une ressource réseau indisponible pour ses utilisateurs légitimes. Pour ce faire, l'attaquant perturbe temporairement voire indéfiniment les services d'un hôte connecté à un réseau. Le mode opératoire consiste généralement à identifier une vulnérabilité ou une faiblesse dans la cible, puis à l'exploiter en inondant le système de trafic massif, de requêtes répétées ou de données malveillantes. Cette surcharge empêche le traitement du trafic légitime. Les ressources de la machine cible telles que le processeur, la mémoire ou la bande passante sont alors sollicitées au-delà de leurs capacités, ce qui peut entraîner un ralentissement significatif du système, voire une interruption totale de service [1], [6].



FIGURE 1.1 : Illustration d'une attaque par déni de service (DoS) [4]

1.1.2 Formes d'attaques par déni de service

Il existe deux formes générales d'attaques par déni de service : les attaques par inondation et les attaques par crash. Les attaques par inondation consistent à submerger un système de trafic jusqu'à ce que ses ressources (comme les files d'attente ou la mémoire tampon) soient saturées. Le serveur, incapable de gérer ce volume excessif, ralentit considérablement, puis cesse progressivement de répondre aux requêtes légitimes.
Les attaques par crash, quant à elles, exploitent des failles présentes dans le système ou dans les services ciblés. L'attaquant envoie une requête spécialement conçue pour provoquer un dysfonctionnement, un plantage ou une instabilité critique. Cela rend le système soit totalement indisponible, soit partiellement inutilisable.

1.1.3 Attaques par déni de service distribuée

Le déni de service distribué (ou DDoS) (cf. figure 1.2) est une forme évoluée de l'attaque DoS, caractérisée par une exécution coordonnée et à grande échelle visant à perturber la disponibilité des services d'un système ou des ressources d'un réseau. Contrairement à une attaque DoS classique, l'attaque DDoS est lancée de manière indirecte à partir d'un grand nombre de machines compromises, appelées « zombies ». Ces machines sont généralement infectées par des logiciels malveillants et intégrées à un réseau contrôlé par l'attaquant, appelé botnet. Les ordinateurs zombies agissent alors de concert pour envoyer un flot massif de requêtes vers la cible, saturant ses ressources (processeur, mémoire, bande passante) jusqu'à l'arrêt complet ou partiel du service. Cette approche rend l'attaque plus difficile à détecter et à stopper, car elle provient de multiples sources géographiquement dispersées [6].



FIGURE 1.2 : Illustration d'une attaque par déni de service distribué (DDoS) [4]

1.1.4 Types d'attaques par déni de service

1.1.4.1 Attaques volumétriques

Les attaques volumétriques, encore appelées attaques par épuisement de la bande passante, constituent une catégorie courante d'attaques par déni de service. Elles visent à saturer la capacité de bande passante d'une cible spécifique, telle qu'un serveur, un site web ou une infrastructure réseau. Le principe de ces attaques repose sur la génération d'un volume massif de trafic inutile, destiné à submerger la cible. L'objectif est de provoquer une congestion du réseau, empêchant ainsi les utilisateurs légitimes d'accéder aux services.
Pour compliquer la détection et la défense, les attaquants utilisent diverses techniques telles que : la

variation des adresses IP source, IP Spoofing [1] en anglais, la modification dynamique de la taille des paquets et la dispersion du trafic depuis de multiples points d'origine (souvent via des botnets). Les conséquences peuvent aller d'un ralentissement important du service à un arrêt total, en passant par une expérience utilisateur dégradée.
Un exemple typique d'attaque volumétrique est l'inondation UDP (cf. figure 1.3). Dans ce cas, l'attaquant envoie un grand nombre de paquets UDP à des ports aléatoires sur le serveur cible. Celui-ci, en tentant de répondre à chaque requête, voit ses ressources rapidement consommées, ce qui entraîne un ralentissement ou une interruption du trafic légitime [5].

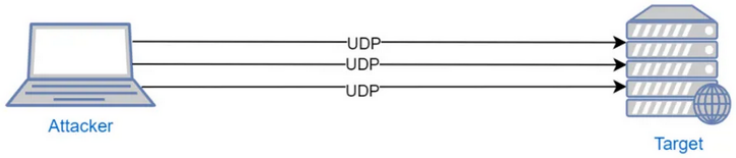


FIGURE 1.3 : Illustration d'une attaque par inondation UDP [5]

1.1.4.2 Attaques par déni de service SYN Flood

Une attaque SYN Flood (cf. figure 1.5) est une forme spécifique d'attaque par déni de service qui exploite le mécanisme de négociation de connexion TCP. Elle consiste à envoyer un grand nombre de requêtes de synchronisation TCP (SYN) incomplètes à un serveur, dans le but de saturer ses ressources et de le rendre indisponible. Le protocole TCP est reconnu pour sa fiabilité, notamment grâce à son processus de connexion en trois étapes, appelé **three-way handshake** (cf. figure 1.4). Ce processus se déroule comme suit [5] :

- Le client envoie une requête SYN au serveur pour initier la connexion.
- Le serveur répond avec un paquet SYN-ACK, confirmant la réception et sa volonté d'établir la connexion.
- Le client envoie un dernier paquet ACK, finalisant la connexion.

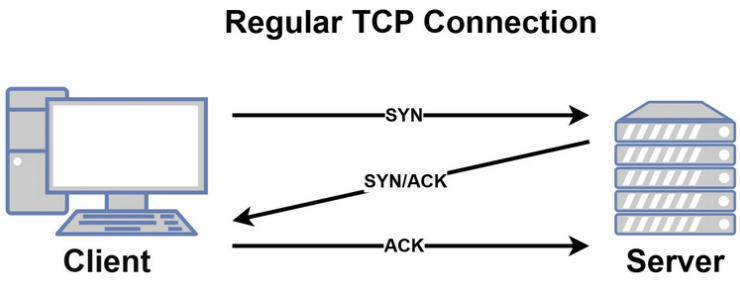


FIGURE 1.4 : Établissement d'une connexion TCP standard à l'aide du processus de poignée de main en trois étapes (Three-Way Handshake) [5]

Dans une attaque SYN Flood, l'agresseur envoie de multiples requêtes SYN, mais n'envoie jamais le paquet ACK final. Le serveur, pensant que les connexions sont toujours en cours d'établissement, conserve chacune d'elles dans une file d'attente limitée en mémoire. Ces connexions incomplètes finissent par saturer la file d'attente, empêchant ainsi de nouvelles connexions légitimes d'être

[1] L'**IP Spoofing** est une technique où un expéditeur falsifie délibérément l'adresse IP source d'un paquet réseau pour tromper les destinataires sur l'origine réelle du paquet, souvent utilisée à des fins malveillantes ou pour des activités illicites en ligne.

acceptées. Les paquets SYN utilisés dans l'attaque semblent légitimes, mais proviennent souvent d'adresses IP falsifiées (IP spoofing) ou de clients incapables de compléter la négociation. Le serveur reste alors dans un état d'attente inutilement prolongé, ce qui entraîne une consommation excessive des ressources mémoire et une dégradation ou un arrêt du service [5].

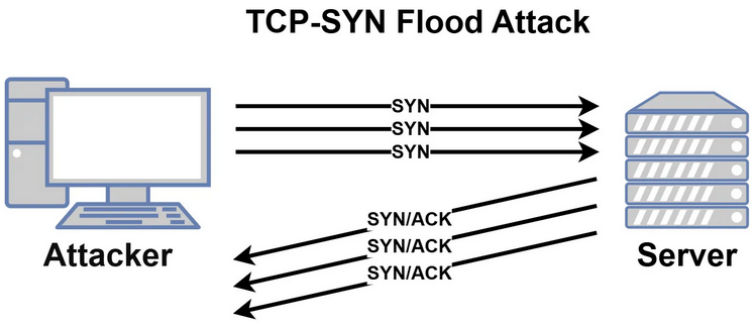


FIGURE 1.5 : Illustration d'une attaque SYN Flood [5]

1.1.4.3 Attaques par amplification/réflexion

Les attaques par amplification consistent à générer un volume important de paquets destinés à submerger la cible sans alerter les intermédiaires. Le principe de l'attaque repose sur l'envoi d'une requête, souvent appelée paquet de déclenchement, à un service vulnérable. Ce dernier, en réponse, émet une réponse bien plus volumineuse que la requête initiale. Grâce à des outils facilement accessibles, l'attaquant peut envoyer des milliers de requêtes à ces services vulnérables, entraînant ainsi une amplification notable du trafic en termes de taille et de bande passante dirigée vers la cible. Cette amplification se traduit soit par l'envoi de plusieurs paquets de réponse pour une seule requête, soit par des paquets ayant une taille supérieure à celle de la demande d'origine. L'amplification DNS (cf. figure 1.6) constitue un exemple typique de cette technique [5].

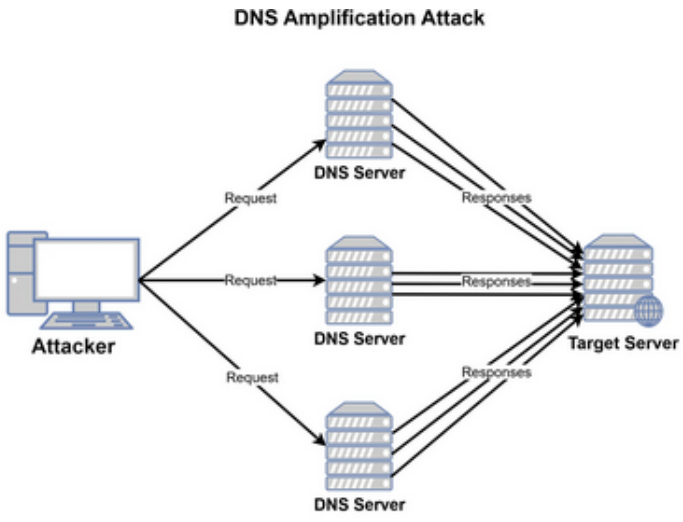


FIGURE 1.6 : Illustration d'une attaque par amplification DNS [5]

Les attaques par réflexion reposent sur l'utilisation de serveurs non compromis pour rediriger le trafic vers la victime, contribuant ainsi à épuiser sa bande passante. Dans ce type d'attaque, l'attaquant envoie des paquets dont l'adresse source est falsifiée au profit de celle de la victime. Ainsi, lorsqu'un serveur reçoit ces requêtes, il renvoie sa réponse directement à la victime et non à l'expéditeur réel du paquet. Cette méthode permet à l'attaquant d'acheminer indirectement un volume

important de trafic vers la cible tout en limitant les risques de détection. La réflexion DNS (cf. figure 1.7) en est un parfait exemple [5].

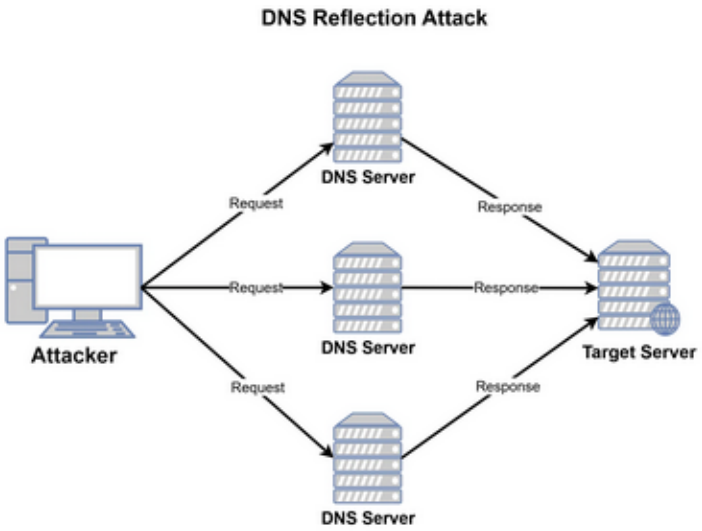


FIGURE 1.7 : Illustration d'une attaque par réflexion DNS [5]

1.1.4.4 Attaques applicatives

Les attaques applicatives ciblent directement les applications web et les services en ligne en exploitant des vulnérabilités spécifiques au niveau de l'application elle-même. Bien que la finalité reste identique, perturber ou rendre indisponible un service , ces attaques se distinguent par leur précision et leur capacité à contourner les systèmes traditionnels de sécurité réseau. Les techniques utilisées dans ce type d'attaque sont souvent variées et sophistiquées, allant de l'exploitation de failles de validation d'entrée à la surcharge volontaire des ressources serveur via des requêtes soigneusement malformées. Les protocoles d'application les plus souvent visés sont HTTP et HTTPS, qui constituent la base des communications web. Un exemple emblématique est l'attaque Slowloris (cf. figures 1.8 et 1.9), qui consiste à ouvrir de multiples connexions HTTP vers le serveur cible, puis à envoyer des en-têtes HTTP très lentement et de manière incomplète. Le serveur, en attente de la fin des requêtes, conserve ces connexions ouvertes, ce qui épuise progressivement ses ressources disponibles et empêche le traitement de nouvelles requêtes légitimes [5].

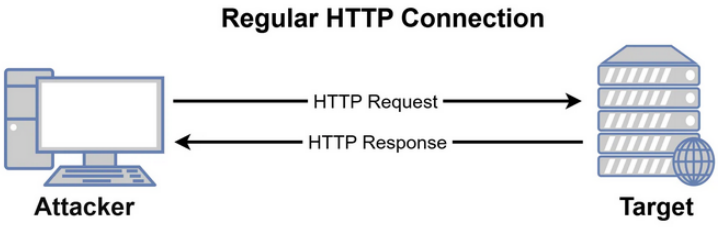


FIGURE 1.8 : Établissement d'une connexion HTTP standard [5]



FIGURE 1.9 : Illustration d'une attaque Slowloris [5]

1.2 Techniques de détection des attaques par déni de service

1.2.1 Détection basée sur la signature

La détection des attaques par déni de service basée sur la signature est une méthode de sécurité qui repose sur la reconnaissance de motifs de trafic ou de comportements spécifiques associés à des attaques déjà répertoriées. Elle fonctionne à partir de signatures prédéfinies, élaborées par des experts en sécurité, qui décrivent les caractéristiques distinctives d'une attaque connue telles que des adresses IP sources, des ports, des types de protocoles ou des séquences spécifiques de requêtes.

Une fois ce système en place, le mécanisme de détection surveille en continu le trafic réseau et le compare aux signatures stockées. Lorsqu'une correspondance est détectée, une alerte est générée afin d'avertir les administrateurs de sécurité. Dans certains cas, des mesures automatiques de mitigation peuvent être déclenchées, comme le blocage immédiat du trafic suspect.

L'un des principaux avantages de cette approche est sa rapidité et sa précision dans l'identification des attaques déjà connues. Elle est également peu gourmande en ressources, car la comparaison de signatures est généralement rapide. Cependant, cette méthode souffre de limites majeures. Elle est inefficace contre les attaques inconnues ou les variantes légèrement modifiées de menaces existantes. Les attaquants peuvent facilement contourner la détection en modifiant certains paramètres pour éviter les correspondances exactes avec les signatures. Cela impose une mise à jour constante des bases de signatures, ce qui rend le système dépendant de la réactivité des équipes de sécurité.

1.2.2 Détection basée sur le comportement

La détection des attaques par déni de service basée sur le comportement repose sur l'analyse dynamique des modèles de trafic réseau afin d'identifier les anomalies par rapport à un comportement jugé normal. Contrairement à la méthode par signature, cette approche ne cherche pas des motifs d'attaques connus, mais plutôt des déviations statistiques ou comportementales dans les données réseau observées en temps réel.

Le système établit un profil de trafic "normal" à partir de données historiques ou d'une phase d'apprentissage initiale, puis le compare en permanence aux flux réseau actuels. Lorsqu'une déviation significative est détectée comme une augmentation soudaine du volume de paquets, des connexions inhabituelles ou un changement dans la distribution des ports ou protocoles utilisés, une alerte est générée, et des mécanismes de défense peuvent être automatiquement déclenchés.

Cette méthode présente l'avantage majeur de pouvoir détecter des attaques nouvelles ou inconnues, grâce à son approche dynamique qui ne dépend pas de signatures prédéfinies ; elle est également capable de s'adapter aux évolutions du trafic et de repérer des comportements anormaux subtils. Toutefois, elle souffre d'un inconvénient notable : elle peut générer un grand nombre de fausses

alertes si le modèle de comportement normal n'est pas bien calibré, et elle requiert une mise en œuvre complexe, souvent appuyée par des techniques avancées comme l'apprentissage automatique, ce qui peut la rendre coûteuse en ressources.

1.2.3 Détection basée sur basée sur l'apprentissage automatique

LLa détection basée sur l'apprentissage automatique représente une approche avancée de la sécurité qui se distingue par l'utilisation de modèles prédictifs formés sur des données historiques pour identifier les comportements anormaux du trafic réseau. Le processus commence par la collecte rigoureuse de données, incluant des informations telles que le volume de trafic, la fréquence des connexions, ou encore les types de protocoles utilisés. Ces données sont utilisées pour entraîner un modèle d'apprentissage automatique, capable de distinguer les comportements légitimes des comportements suspects. Une fois le modèle entraîné, il est déployé pour surveiller le trafic en temps réel. Le système compare continuellement les flux actuels aux schémas appris afin de détecter toute anomalie significative. En cas de comportement potentiellement malveillant, une alerte est générée à l'attention des administrateurs de sécurité, pouvant entraîner des mesures de mitigation automatiques si nécessaire. Cette approche permet de détecter des attaques nouvelles ou évolutives sans dépendre de signatures précises, mais elle nécessite des données d'apprentissage de qualité, une puissance de calcul importante et peut générer des faux positifs si le modèle est mal entraîné ou mal ajusté.

1.2.4 Détection hybride

La détection hybride est une approche qui combine les avantages de la détection basée sur le comportement et de la détection basée sur la signature, afin d'améliorer la précision de la détection tout en réduisant les taux de faux positifs. En croisant l'analyse des signatures connues avec l'identification d'anomalies dans le comportement du trafic réseau, cette méthode permet de détecter à la fois les attaques déjà documentées et celles émergentes ou inconnues. Elle offre ainsi une couverture plus complète et une meilleure réactivité face à la diversité des attaques par déni de service, tout en limitant les inconvénients propres à chaque méthode prise isolément.

1.3 Apprentissage automatique en cybersécurité

1.3.1 Qu'est-ce que l'apprentissage automatique ?

L'apprentissage automatique, ou *Machine Learning* (ML) en anglais, est un sous-ensemble de l'intelligence artificielle (IA) qui consiste à entraîner des algorithmes à apprendre des modèles à partir de données existantes, dans le but de faire des prédictions sur de nouvelles données. Bien que les termes IA et ML soient souvent utilisés de manière interchangeable, ils désignent des concepts distincts. L'intelligence artificielle englobe l'ensemble des techniques permettant aux machines de simuler des comportements intelligents similaires à ceux des humains, tandis que le *Machine Learning* désigne spécifiquement les systèmes capables d'apprendre automatiquement à partir de données, sans être explicitement programmés pour chaque tâche [7].

1.3.2 Types d'apprentissage automatique

1.3.2.1 Apprentissage supervisé

L'apprentissage supervisé consiste à entraîner un modèle à partir de données d'entrée associées à des étiquettes ou résultats connus. L'objectif est d'apprendre au modèle à établir une relation entre les entrées et les sorties afin qu'il puisse effectuer des prédictions précises sur de nouvelles données non encore vues. En cybersécurité, cette méthode est fréquemment utilisée pour détecter les comportements malveillants : le modèle est entraîné sur un ensemble de données contenant à la fois des exemples de trafic bénin et de trafic malveillant, ce qui lui permet par la suite de classifier automatiquement de nouveaux flux comme sûrs ou suspects [8].

1.3.2.2 Apprentissage non supervisé

L'apprentissage non supervisé consiste à entraîner un modèle sur des données non étiquetées, sans indication préalable des résultats attendus. Le modèle cherche alors seul à identifier des structures, des relations ou des motifs dans les données, comme des groupes (clusters) ou des regroupements naturels. En cybersécurité, cette approche est particulièrement utile pour découvrir de nouveaux comportements malveillants ou des anomalies dans de larges volumes de données, ce qui facilite la détection d'attaques inconnues [8].

1.3.2.3 Apprentissage semi-supervisé

L'apprentissage semi-supervisé combine des éléments des approches supervisée et non supervisée. Il utilise un petit ensemble de données étiquetées accompagné d'un grand volume de données non étiquetées pour entraîner un modèle. Cette méthode permet de réduire significativement le temps et les efforts nécessaires à l'annotation manuelle des données, notamment lorsque les bases de données sont très volumineuses. Elle est par exemple employée dans la classification d'images ou la détection d'anomalies en cybersécurité.

1.3.2.4 Apprentissage par renforcement

L'apprentissage par renforcement repose sur un processus d'apprentissage par essais et erreurs, où le modèle ne reçoit pas d'exemples étiquetés, mais apprend à maximiser une récompense cumulative en fonction des actions qu'il effectue dans un environnement donné. Cette méthode s'inspire étroitement de l'apprentissage humain et est adaptée pour trouver des solutions innovantes à des problèmes complexes. En cybersécurité, elle est utilisée notamment pour la gestion autonome des systèmes cyberphysiques, la détection d'intrusions automatiques, ainsi que la défense contre les attaques par déni de service distribué [8].

1.3.3 Utilisation de l'apprentissage automatique en cybersécurité

L'apprentissage automatique est utilisé dans un large éventail de cas (cf. tableau 1.1) en constante évolution dans le domaine de la cybersécurité [7] :

TABLE 1.1 : Exemples d'applications de l'apprentissage automatique en cybersécurité

Cas d'utilisation	Description

Détection d'anomalies	Identifie les anomalies dans les données pour éclairer l'évaluation des risques et orienter les enquêtes sur les menaces.
Classification des Menaces	Permet de classer les types d'attaques en fonction des caractéristiques du trafic réseau ou des événements de sécurité.
Analyse de Comportement Utilisateur et Entité (UEBA)	Surveille et analyse les comportements des utilisateurs et des entités au sein d'un réseau pour détecter des anomalies potentiellement malveillantes.
Détection de Malwares	Identifie les logiciels malveillants en analysant des caractéristiques telles que les signatures de fichiers, les comportements d'exécution, et les modèles de réseau.
Prévention des Intrusions	Permet de prévenir les intrusions en anticipant et en bloquant les comportements malveillants avant qu'ils ne causent des dommages.
Analyse médico-légale	Exécute des opérations de contre-espionnage pour analyser la progression des attaques et identifier les vulnérabilités du système

1.3.4 Avantages de l'apprentissage automatique en cybersécurité

L'application de l'apprentissage automatique aux problèmes de cybersécurité présente de nombreux avantages. Parmi ceux-ci, on peut citer [7] :

- **Synthétisation rapide de gros volumes de données :** l'un des plus grands défis auxquels sont confrontés les analystes est la nécessité de synthétiser rapidement les renseignements générés sur leur surface d'attaque, qui sont généralement générés beaucoup plus rapidement que ce que leurs équipes peuvent traiter manuellement. L'apprentissage automatique est capable d'analyser rapidement de gros volumes de renseignements historiques et dynamiques, permettant aux équipes d'exploiter les données provenant de diverses sources en temps quasi réel.
- **Activation de l'intelligence experte à grande échelle :** des cycles de formation réguliers permettent aux modèles d'apprendre en continu à partir de leur population d'échantillons en évolution, qui comprend des détections étiquetées par des analystes ou des alertes examinées par des analystes. Cela évite les faux positifs récurrents et permet aux modèles d'apprendre et d'appliquer la vérité fondamentale générée par les experts.
- **Automatisation des tâches répétitives et manuelles :** l'application de l'apprentissage automatique à des tâches spécifiques peut aider les équipes de sécurité à s'affranchir des tâches répétitives et banales, agissant comme un multiplicateur de force qui leur permet d'adapter leur réponse aux alertes entrantes et de réorienter le temps et les ressources vers des projets complexes et stratégiques.
- **Augmentation de l'efficacité des analystes :** l'apprentissage automatique peut augmenter la perspicacité des analystes avec des renseignements actualisés en temps réel, permettant aux analystes des opérations de recherche de menaces et de sécurité de hiérarchiser efficacement les ressources pour traiter les vulnérabilités critiques de leur organisation et d'enquêter sur les détections d'alertes ML urgentes.

1.3.5 Défis et contraintes de l'apprentissage automatique

Bien que les modèles d'apprentissage automatique puissent être des outils puissants, chaque modèle fonctionne avec des limitations uniques :

- **Données de qualité suffisantes :** la formation de modèles de confiance élevée nécessite souvent l'accès à de grands ensembles de données, à la fois pour former et tester des modèles d'apprentissage automatique. Pour tester les modèles, un sous-ensemble de données est généralement mis à part de l'ensemble d'apprentissage afin de tester les performances du modèle. Ces données doivent avoir un chevauchement minimal de caractéristiques avec les données d'apprentissage ; par exemple, elles doivent représenter une période de collecte de données différente ou provenir d'une source de données différente. Si les données de qualité sont insuffisantes, un espace de problème donné peut ne pas être un scénario approprié pour l'apprentissage automatique appliqué.

- **Compromis entre les vrais et les faux positifs :** comme indiqué précédemment, la sensibilité de chaque modèle doit être calibrée pour équilibrer le seuil de détection entre les vrais et les faux positifs afin de maximiser l'efficacité de la détection.

- **Explicabilité :** l'explicabilité fait référence à la capacité d'expliquer comment et pourquoi un modèle fonctionne comme il le fait. Cela permet aux équipes de science des données de comprendre quelles caractéristiques d'un échantillon influencent les performances du modèle et leurs pondérations relatives. L'explicabilité est essentielle pour favoriser la responsabilisation, instaurer la confiance, garantir le respect des politiques de données et, en fin de compte, permettre une amélioration continue des performances de l'apprentissage automatique.

- **Répétabilité :** également appelée reproductibilité, elle fait référence à la capacité des expériences d'apprentissage automatique à être reproduites de manière cohérente. La répétabilité favorise la transparence sur la manière dont l'apprentissage automatique est utilisé, les types de modèles utilisés, les données sur lesquelles ils sont formés et les environnements ou versions de logiciels dans lesquels ils fonctionnent. La répétabilité minimise l'ambiguïté et les erreurs potentielles lorsque les modèles passent des tests au déploiement et à travers les futurs cycles de mise à jour.

- **Optimisation pour l'environnement cible :** chaque modèle doit être optimisé pour son environnement de production cible. Chaque environnement varie en termes de disponibilité des ressources de calcul, de mémoire et de connectivité. Par la suite, chaque modèle doit être conçu pour fonctionner dans son environnement de déploiement, sans surcharger ni interrompre les opérations de l'hôte cible.

- **Renforcement contre les attaques adverses :** les modèles d'apprentissage automatique ont leur propre surface d'attaque qui peut être vulnérable aux attaques adverses, où ces derniers peuvent tenter d'exploiter ou de modifier le comportement du modèle (par exemple en faisant en sorte que le modèle classe mal les échantillons). Pour minimiser la surface d'attaque exploitable des modèles, les data scientists « renforcent » les modèles lors de la formation afin de garantir des performances et une résilience robustes contre les attaques.

1.4 Autres Travaux

Les techniques d'apprentissage automatique se sont révélées parmi les méthodes les plus efficaces pour lutter contre les intrusions dans les systèmes informatiques, qu'il s'agisse des applications, des hôtes ou des réseaux. Dans notre revue de la littérature, nous avons exploré divers travaux cherchant à identifier les meilleures approches dans ce domaine.

Des techniques telles que K-Nearest Neighbors (KNN), Naives Bayes, Decision Tree, Random Forest, Support Vector Machine (SVM) et AdaBoost, ont été utilisées par des auteurs comme Chavan, N., et al (2022), T. E. RAZANADRAKOTO (2022) ainsi que Avci et Koca (2023) pour la détection d'intrusions. Ces chercheurs ont proposé différentes méthodes entraînées sur plusieurs jeux de données, notamment le NSL-KDD, obtenant des résultats prometteurs.

Ces approches basées sur l'apprentissage automatique surpassent généralement les méthodes classiques, ce qui confirme une fois de plus l'efficacité de cette technique pour la détection d'intrusions.

Conclusion

Cette section a exploré en profondeur les principes fondamentaux des attaques par déni de service, leurs différents types, ainsi que les diverses méthodes employées pour leur détection. Nous avons mis en lumière les avantages, les limites et les domaines d'application spécifiques de chaque approche. Ainsi, les méthodes basées sur les signatures se révèlent efficaces pour identifier des attaques connues, tandis que les techniques d'apprentissage automatique offrent une capacité précieuse à détecter des attaques inconnues grâce à leurs modèles prédictifs. Cette base théorique ouvre la voie à l'analyse et aux choix techniques de notre approche.

Chapitre 2

Analyse et choix techniques

Introduction

Dans ce chapitre, nous présentons les outils de machine learning (ML) que nous avons utilisés pour entraîner notre modèle, ainsi que les différentes étapes du processus suivi. Nous examinons également plusieurs méthodes de sélection et de classification des attributs basées sur l'apprentissage automatique. Ces techniques sont particulièrement adaptées pour différencier efficacement les attaques des requêtes bénignes.

2.1 Outils de développement

2.1.1 Langage utilisé

Pour notre étude, nous avons choisi d'utiliser le langage Python. Python est un langage de programmation polyvalent, adapté à de nombreux domaines grâce à son écosystème riche en bibliothèques spécialisées. Il est largement utilisé dans des secteurs comme l'intelligence artificielle, le machine learning, le traitement des Big Data, ainsi que dans l'informatique quantique. Sa simplicité et sa flexibilité en font un choix privilégié pour le développement de modèles de détection d'attaques par déni de service.

2.1.2 Bibliothèques utilisées

2.1.2.1 NumPy

NumPy est une bibliothèque Python très populaire, principalement utilisée pour réaliser des calculs mathématiques et scientifiques. Elle offre une large gamme de fonctionnalités et d'outils performants pour la manipulation de tableaux multidimensionnels, ce qui est essentiel dans les projets de data science et d'apprentissage automatique [9].

2.1.2.2 Pandas

Pandas est une bibliothèque open-source spécialement conçue pour la manipulation et l'analyse de données en Python. Elle permet de travailler facilement avec des structures de données tabulaires,

comme les DataFrames, et propose des outils puissants pour le nettoyage, la transformation et l'exploration des données, tout en restant simple à utiliser [9].

2.1.2.3 Scikit-Learn

Scikit-Learn est une bibliothèque Python dédiée au machine learning. Grâce à son interface simple et unifiée, elle offre un accès à une grande variété d'algorithmes pour la classification, la régression, le clustering, la réduction de dimensionnalité, ainsi que pour la sélection de modèles [10].

2.1.2.4 Matplotlib

Matplotlib est une bibliothèque open source de visualisation de données en Python, permettant de créer facilement des graphiques, tracés et visualisations variées, utiles pour analyser et présenter les résultats.

2.1.3 Environnements

2.1.3.1 Jupyter Notebook

Jupyter Notebook est une application web open-source qui permet de créer des cahiers électroniques combinant texte, images, code informatique et équations mathématiques dans un même document. Il offre un environnement interactif de calcul basé sur le web, très apprécié pour le prototypage et l'analyse de données [9].

2.1.3.2 Kaggle

Kaggle est une plateforme web regroupant la plus grande communauté mondiale de data scientists. Elle propose des outils et des ressources puissants, tels que l'accès gratuit à des GPU, de nombreuses bases de données, et un large éventail de codes partagés par la communauté, facilitant ainsi le développement et le partage de projets en science des données [9].

2.2 Jeu de données

La sélection d'un jeu de données adapté constitue un facteur clé pour la réussite de notre étude. Il doit contenir un volume suffisant d'échantillons représentatifs à la fois de trafic bénin et de trafic malveillant de type DoS, issus de divers environnements. Toutefois, dans le domaine de la détection d'intrusion, les jeux de données sont généralement rares, en raison de leur caractère sensible pour les entreprises.
Pour répondre à nos exigences, nous avons choisi d'utiliser **NSL-KDD**, un jeu de données accessible au public. Il s'agit d'une version améliorée du jeu de données **KDD'99**, qui lui-même provient d'une initiative de la **DARPA** (Defense Advanced Research Projects Agency) en 1998, visant à évaluer les performances des systèmes de détection d'intrusion. Le jeu de données NSL-KDD a été nettoyé afin d'éliminer les doublons présents dans KDD'99, ce qui le rend plus équilibré et pertinent pour l'évaluation des modèles de machine learning. Il a été généré à partir de la simulation d'un environnement réseau, ce qui permet d'éviter les problèmes de confidentialité associés aux données réelles d'entreprise.

NSL-KDD comprend un total de 148 517 enregistrements, répartis entre trafic normal et malveillant. Environ 53 % du trafic est normal, tandis que 47 % est considéré comme anormal (cf. figure 2.1). Cette distribution équilibrée en fait un excellent choix pour l'entraînement et la validation de modèles de détection d'attaques par déni de service.

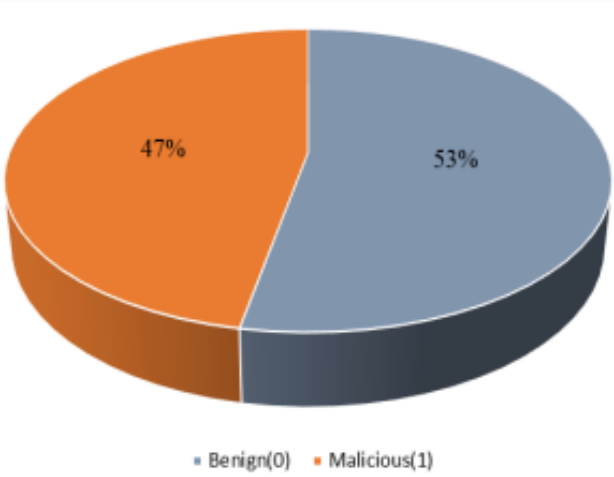


FIGURE 2.1 : Répartition des enregistrements dans le jeu de données NSL-KDD selon les deux classes

L'ensemble d'apprentissage englobe quatre classes d'attaques : Denial of Service (DoS), Probe, User to Root (U2R), et Remote to Local (R2L). Ces attaques sont subdivisées en types (cf. tableau 2.1) plus précis.

TABLE 2.1 : Répartition des types d'attaques selon les classes d'attaques dans l'ensemble de données NSL-KDD

Classes d'attaques	**Types d'attaques**
DoS	Apache2, Back, Land, Mailbomb, Neptune, Pod, Processtable, Smurf, Teardrod, Udpstorm, Worm
Probe	Ipsweep, Mscan, Nmap, Portsweep, Saint, Satan
R2L	Ftp_write, Guess_passwd, Httptunnel, Imap, Multihop, Named, Phf, Sendmail, Snmpgetattack, Snmpguess, Spy, Warezclient, Warezmaster, Xlock, Xsnoop
U2R	Buffer_overflow, Loadmodule, Perl, Ps, Rootkit, Sqlattack, Xterm

Il est structuré en deux fichiers distincts : un pour l'entraînement des modèles et l'autre pour la validation. Le nombre d'enregistrements dans les ensembles d'entraînement et de test NSL-KDD (cf. tableaux 2.2 ; 2.3 et figure 2.2) sont raisonnables, ce qui rend abordable l'exécution des expériences sur l'ensemble complet sans qu'il soit nécessaire de sélectionner au hasard une petite partie.

TABLE 2.2 : Répartition des classes dans l'ensemble d'entraînement NSL-KDD

Classes	Nombre d'enregistrements
DoS	45927
Probe	11656
R2L	995
U2R	52
Normal	67343

TABLE 2.3 : Répartition des classes dans l'ensemble de test NSL-KDD

Classes	Nombre d'enregistrements
DoS	7460
Probe	2885
R2L	2421
U2R	67
Normal	9711

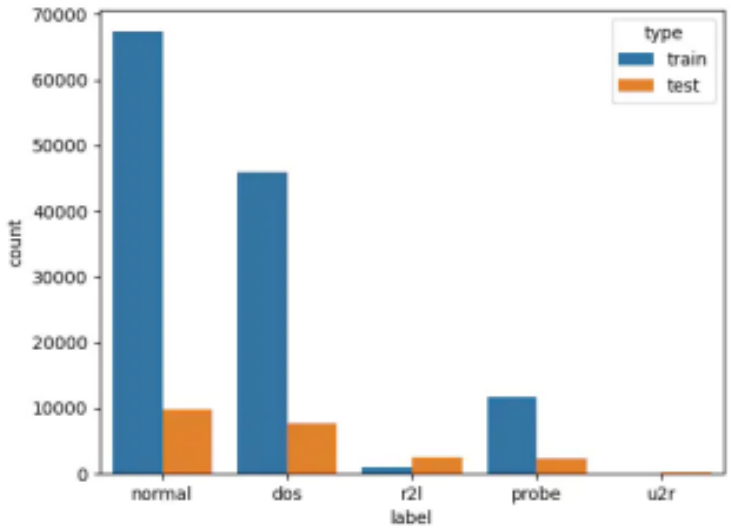


FIGURE 2.2 : Répartition des enregistrements par classe dans les ensembles d'entraînement et de test du jeu de données NSL-KDD

Pour classifier la nature du trafic, l'ensemble d'apprentissage exhibe 41 caractéristiques (cf. tableau 2.4) représentant l'état actuel du réseau.

TABLE 2.4 : Les attributs du jeu de données NSL-KDD

No	Attribut	Description
1	duration	Durée de la connexion en secondes
2	protocol_type	Protocole utilisé dans la connexion (icmp, tcp, udp)
3	service	Service réseau de destination utilisé (http, telnet, ftp_data...)
4	flag	Statut de la connexion - Normal ou Erreur (SF, REJ, S0, S1...)

5	src_bytes	Nombre d'octets de donnée transférés de la source à la destination en une seule connexion
6	dst_bytes	Nombre d'octets de données transférés de la destination à la source dans une connexion unique
7	land	Si les adresses IP source et destination et les numéros de port sont les mêmes, alors land=1 sinon land=0
8	wrong_fragment	Nombre total de fragments incorrects dans cette connexion
9	urgent	Nombre de paquets urgents dans cette connexion
10	hot	Nombre d'indicateurs « hot » dans le contenu
11	num_failed_logins	Nombre de tentatives de connexion échouées
12	logged_in	Si connecté avec succès, alors logged_in=1 sinon logged_in=0
13	num_compromised	Nombre de conditions compromises
14	root_shell	1 Si le root shell est obtenu, 0 sinon
15	su_attempted	1 Si la commande "su root" a été tentée ou utilisée, 0 sinon
16	num_root	Nombre d'accès "root" ou nombre d'opérations effectuées en tant que racine dans la connexion
17	num_files_creations	Nombre d'opérations de création de fichiers dans la connexion
18	num_shells	Nombre d'invités du shell
19	num_access_files	Nombre d'opérations sur les fichiers de contrôle d'accès
20	num_outbound_cmds	Nombre de commandes sortantes dans une session FTP
21	is_host_login	1 si la connexion appartient à la liste du « hot » (root ou admin), 0 sinon
22	is_guest_login	1 si la connexion est une connexion «guest », 0 sinon
23	count	Nombre de connexions au même hôte de destination que la connexion actuelle au cours des deux dernières secondes
24	srv_count	Nombre de connexions au même service (numéro de port) que la connexion actuelle au cours des deux dernières secondes
25	serror_rate	Pourcentage de connexions qui ont activé l'indicateur "flag" s0, s1, s2 ou s3, parmi les connexions agrégées dans count
26	srv_serror_rate	Pourcentage de connexions qui ont activé l'indicateur "flag" s0, s1, s2 ou s3, parmi les connexions agrégées dans srv_count
27	rerror_rate	Pourcentage de connexions qui ont activé l'indicateur "flag" REJ, parmi les connexions agrégées dans count
28	srv_rerror_rate	Pourcentage de connexions qui ont activé l'indicateur "flag" REJ, parmi les connexions agrégées dans srv_count
29	same_srv_rate	Le pourcentage de connexions qui sont au même service, parmi les connexions agrégées dans count
30	diff_srv_rate	Le pourcentage de connexions qui sont aux différents services, parmi les connexions agrégées dans count
31	srv_diff_host_rate	Le pourcentage de connexions qui sont vers différentes machines de destination, parmi les connexions agrégées dans srv_count

32	dst_host_count	Nombre de connexions ayant la même adresse IP d'hôte de destination
33	dst_host_srv_count	Nombre de connexions ayant le même numéro de port
34	dst_host_same_srv_rate	Le pourcentage de connexions qui sont au même service, parmi les connexions agrégées dans dst_host_count
35	dst_host_diff_srv_rate	Le pourcentage de connexions qui sont aux différents services, parmi les connexions agrégées dans dst_host_count
36	dst_host_same_src_port_rate	Pourcentage de connexions qui sont sur le même port source, parmi les connexions agrégées dans dst_host_srv_count
37	dst_host_srv_diff_host_rate	Pourcentage de connexions qui sont vers différentes machines de destination, parmi les connexions agrégées dans dst_host_srv_count
38	dst_host_serror_rate	Pourcentage de connexions qui ont activé l'indicateur "flag" s0, s1, s2 ou s3, parmi les connexions agrégées dans dst_host_count
39	dst_host_srv_serror_rate	Pourcentage de connexions qui ont activé l'indicateur "flag" s0, s1, s2 ou s3, parmi les connexions agrégées dans dst_host_srv_count
40	dst_host_rerror_rate	Pourcentage de connexions qui ont activé l'indicateur "flag" REJ, parmi les connexions agrégées dans dst_host_count
41	dst_host_srv_rerror_rate	Pourcentage de connexions qui ont activé l'indicateur "flag" REJ, parmi les connexions agrégées dans dst_host_srv_count

2.3 Prétraitement des données

2.3.1 Agrégation et nettoyage des données

- **Agrégation de données** : nous avons regroupé les alertes présentes dans la colonne cible en fonction des différentes classes d'attaques, afin de simplifier l'analyse et la modélisation.
- **Nettoyage de données** : cette étape vise à garantir la qualité et la fiabilité du jeu de données pour permettre des analyses plus précises. Nous avons d'abord détecté les valeurs manquantes, puis supprimé les lignes contenant des données corrompues ou nulles. Enfin, nous avons éliminé les enregistrements correspondant aux classes d'attaques Probe, R2L et U2R, afin de nous concentrer uniquement sur la détection des attaques de type DoS.

2.3.2 Encodage et normalisation des données

- **Encodage de données** : certaines variables telles que protocol_type, service et flag étaient de type catégoriel. Afin de les rendre exploitables par les algorithmes d'apprentissage automatique, nous les avons converties en valeurs numériques à l'aide de techniques d'encodage adaptées.
- **Normalisation de données** : plusieurs attributs présentaient des écarts d'échelle importants, susceptibles d'influencer disproportionnellement les performances du modèle. Pour remédier

à cela, nous avons appliqué une mise à l'échelle des données en utilisant la méthode du **Min-Max Scaling**, qui transforme les valeurs afin qu'elles se situent dans un intervalle défini, généralement [0, 1] ou [−1, 1], selon les besoins. La formule de cette normalisation est la suivante :

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{2.1}$$

Où :

X est la valeur initiale,

X' est la valeur normalisée,

X_{min} **et** X_{max} représentent respectivement les valeurs minimale et maximale de la variable X dans l'ensemble des données.

2.4 Sélection des caractéristiques

La sélection des caractéristiques critiques ayant une influence significative sur le résultat de la classification est une étape essentielle pour améliorer la précision et l'efficacité des modèles d'apprentissage automatique. Elle permet de réduire la dimensionnalité, d'éliminer les attributs redondants ou peu pertinents et d'optimiser le temps de calcul. Dans le cadre de notre projet, nous avons adopté une approche combinée basée sur deux méthodes principales : l'évaluation de l'importance des caractéristiques à l'aide de l'algorithme Random Forest et le calcul du Gain d'Information pour affiner la sélection et écarter les attributs non pertinents.

- **sélection par importance via Random Forest :** la méthode Random Forest repose sur un ensemble d'arbres de décision, chacun entraîné sur un sous-ensemble aléatoire de données. L'importance des caractéristiques est ensuite estimée en analysant l'impact de chaque attribut sur la réduction de l'impureté dans les arbres. Voici les étapes de cette approche :

Algorithme 1 : Sélection de caractéristiques basée sur Random Forest

Data : Ensemble de données d'entrée X avec ses attributs
Result : Sous-ensemble de caractéristiques sélectionnées X'

1 Construire un ensemble d'arbres de décision sur des sous-ensembles aléatoires de X;
2 Évaluer l'importance de chaque caractéristique à partir des arbres construits ;
3 Trier les caractéristiques selon leur importance décroissante;
4 Sélectionner les k caractéristiques les plus importantes pour former l'ensemble X';

- **calcul de l'entropie pour chaque attribut :**

$$H(x_i) = -\sum_{j=1}^{n} p(x_j|c_1)\log_2 p(x_j|c_1) + p(x_j|c_2)\log_2 p(x_j|c_2) \tag{2.2}$$

Où :

x_i : représente un attribut.

x_j : représente une valeur particulière de l'attribut x

c_1, c_2 : dénotent les deux classes de classification (normale, attaque)

n : dénote le nombre de valeurs de l'attribut x

p : la probabilité

- **calcul de gain pour chaque attribut :**

$$Gain = Entropie_E - H(x_i) \tag{2.3}$$

Où :

$$Entropie_E = \frac{nb_{\text{normal}}}{nb_{\text{connexion}}} \log_2(\frac{nb_{\text{normal}}}{nb_{\text{connexion}}}) + \frac{nb_{\text{attack}}}{nb_{\text{connexion}}} \log_2(\frac{nb_{\text{attack}}}{nb_{\text{connexion}}}) \tag{2.4}$$

Avec :

$nb_{\textbf{normal}}$: représente le nombre de connexions qui sont classifiées comme normal

$nb_{\textbf{attack}}$: représente le nombre de connexions qui sont classifiées comme une tentative d'attaque

$nb_{\textbf{connexion}}$: représente le nombre total de connexions dans la base d'apprentissage

- **élimination des attributs ayant un gain inférieur à un seuil donné**
- **utilisation du sous-ensemble d'attributs restant pour la classification**

Les résultats obtenus à l'issue des deux méthodes de sélection sont synthétisés dans les tableaux (cf. tableaux 2.5 et 2.6) suivants, permettant de comparer l'importance attribuée à chaque attribut selon l'approche utilisée.

TABLE 2.5 : Trie des attributs du plus important au moins important basée sur Random Forest

No	Attribut	No	Attribut
1.	src_bytes	**20.**	dst_host_rerror_rate
2.	flag	**21.**	dst_host_srv_rerror_rate
3.	same_srv_rate	**22.**	dst_host_count
4.	dst_host_serror_rate	**23.**	dst_host_srv_diff_host_rate
5.	diff_srv_rate	**24.**	num_compromised
6.	count	**25.**	rerror_rate
7.	dst_bytes	**26.**	hot
8.	srv_serror_rate	**27.**	srv_rerror_rate
9.	serror_rate	**28.**	srv_diff_host_rate
10.	dst_host_srv_serror_rate	**29.**	duration
11.	dst_host_same_srv_rate	**30.**	is_guest_login
12.	protocol_type	**31.**	land
13.	dst_host_diff_srv_rate	**32.**	root_shell
14.	srv_count	**33.**	num_file_creations
15.	logged_in	**34.**	num_root
16.	dst_host_srv_count	**35.**	su_attempted
17.	service	**36.**	num_access_files
18.	wrong_fragment	**37.**	num_shells
19.	dst_host_same_src_port_rate	**38.**	num_failed_logins

TABLE 2.6 : Trie des attributs du plus important au moins important basée sur le Gain d'Information

No	Attribut	No	Attribut
1.	src_bytes	**19.**	srv_count
2.	diff_srv_rate	**20.**	srv_diff_host_rate
3.	service	**21.**	protocol_type
4.	same_srv_rate	**22.**	dst_host_srv_rerror_rate
5.	flag	**23.**	duration
6.	dst_bytes	**24.**	dst_host_rerror_rate
7.	count	**25.**	rerror_rate
8.	dst_host_srv_serror_rate	**26.**	srv_rerror_rate
9.	dst_host_serror_rate	**27.**	hot
10.	dst_host_same_srv_rate	**28.**	wrong_fragment
11.	srv_serror_rate	**29.**	num_compromised
12.	serror_rate	**30.**	is_guest_login
13.	dst_host_diff_srv_rate	**31.**	num_file_creations
14.	dst_host_srv_count	**32.**	num_access_files
15.	logged_in	**33.**	num_root
16.	dst_host_srv_diff_host_rate	**34.**	num_failed_logins
17.	dst_host_count	**35.**	su_attempted
18.	dst_host_same_src_port_rate	**36.**	urgent

2.5 Equilibrage des données

Pour résoudre le problème du déséquilibre des classes dans les ensembles de données NSL-KDD, nous avons appliqué une technique de sur-échantillonnage aléatoire, consistant à augmenter artificiellement la taille des classes minoritaires afin de les aligner sur la classe majoritaire (dans notre cas, la classe "normale"). Cette approche améliore la représentation équilibrée des différentes catégories dans le jeu de données et permet au modèle de mieux apprendre les motifs associés à chaque type d'attaque.

Algorithme 2 : Sur-échantillonage aléatoire

Data : Ensemble de données déséquilibré D
Result : Ensemble de données équilibré D'

$D' \leftarrow D$;
$N \leftarrow$ Nombre d'instances dans la classe minoritaire;
$M \leftarrow$ Nombre d'instances dans la classe majoritaire;
while $N < M$ **do**
    Sélectionner une instance aléatoire x de la classe minoritaire;
    Dupliquer x et l'ajouter à D';
    $N \leftarrow N + 1$;
end

2.6 Choix des algorithmes

Dans le cadre de cette étude, nous avons opté pour une approche d'apprentissage supervisé, particulièrement adaptée à la classification du trafic réseau. L'objectif est de prédire si une connexion est normale ou malveillante, à partir d'un ensemble d'attributs connus. Plusieurs algorithmes de classification supervisée ont été considérés pour leur pertinence dans ce contexte [2, 3] :

- **Arbres de Décision (Decision Tree - DT)** : les arbres de décision sont des structures arborescentes qui permettent de classer les données en suivant des règles conditionnelles. Chaque nœud de l'arbre représente une décision basée sur une caractéristique, tandis que les feuilles correspondent aux prédictions finales. Cet algorithme est intuitif, interprétable et bien adapté aux problèmes de classification, bien qu'il puisse souffrir de surapprentissage lorsqu'il est utilisé seul.
- **Forêts Aléatoires (Random Forest - RF)** : la forêt aléatoire est un algorithme d'ensemble basé sur une multitude d'arbres de décision, construits à partir de sous-échantillons aléatoires du jeu de données. Elle corrige les faiblesses des arbres de décision seuls en réduisant la variance et en améliorant la robustesse du modèle. C'est un des algorithmes les plus utilisés en cybersécurité pour ses performances élevées.
- **K-Plus Proches Voisins (K-Nearest Neighbors - KNN)** : KNN est un algorithme simple et non paramétrique qui classe une observation en fonction des classes des K instances les plus proches dans l'espace des caractéristiques. Il repose sur la notion de distance (souvent euclidienne) pour identifier les voisins. Cet algorithme est efficace sur des jeux de données bien préparés, mais peut être sensible au bruit et aux dimensions élevées.
- **Machines à Vecteurs de Support (Support Vector Machine - SVM)** : les SVM sont des classificateurs linéaires puissants qui cherchent à trouver l'hyperplan optimal séparant les classes avec une marge maximale. Cet algorithme est performant pour les données linéairement séparables, mais son efficacité diminue lorsque les données sont complexes ou bruitées. Dans notre contexte, il est moins adapté compte tenu de la complexité du trafic réseau simulé.
- **Classificateur Bayésien Naïf (Naive Bayes)** : basé sur le théorème de Bayes, cet algorithme suppose l'indépendance entre les attributs. Il est particulièrement performant dans les tâches de traitement du langage naturel, mais peut montrer des limites dans des contextes où les attributs sont fortement corrélés, comme c'est souvent le cas dans les jeux de données de réseau.
- **Méthodes de Boosting (XGBoost, LightGBM, Gradient Boosting)** : les algorithmes de boosting sont des méthodes d'ensemble qui construisent des modèles séquentiellement, chaque nouveau modèle corrigeant les erreurs du précédent. XGBoost et LightGBM, en particulier, sont connus pour leur efficacité, leur rapidité et leurs performances de pointe en classification. Ils sont adaptés à des contextes complexes comme la détection d'intrusions.

Avant de commencer notre projet de classification basé sur l'apprentissage automatique, il est essentiel d'évaluer les algorithmes candidats selon plusieurs critères fondamentaux (cf. tableaux 2.7 et 2.8). Cette évaluation nous permet de sélectionner les modèles les plus adaptés à la détection d'attaques DoS dans un environnement réaliste.

TABLE 2.7 : Les critères fondamentaux

Critère	Explication
Précision (Accuracy)	Mesure la proportion de bonnes prédictions. Pertinent si les classes sont équilibrées.
F1-score	Moyenne harmonique entre précision et rappel. Très utile en cas de classes déséquilibrées.
AUC-ROC	Évalue la capacité du modèle à distinguer les classes (pertinent pour les détections de DoS vs normal).
Robustesse au bruit	Capacité à gérer les valeurs aberrantes et données bruitées.
Interprétabilité	Facilité avec laquelle un humain peut comprendre les décisions du modèle.
Complexité temporelle	Temps d'entraînement et de prédiction requis par le modèle (important sur de grands ensembles de données).
Scalabilité	Capacité du modèle à s'adapter à des volumes de données croissants.
Exigences matérielles	Ressources nécessaires à l'exécution du modèle (CPU, RAM, GPU).

TABLE 2.8 : Comparaison des performances des algorithmes sur ces critères

Critère	DT	RF	KNN	SVM	NB	Boosting (XGBoost)
Accuracy	4	5	4	4	3	5
F1-score	4	5	4	4	3	5
AUC-ROC	3	5	3	4	3	5
Robustesse	3	5	2	4	2	5
Interprétabilité	5	3	4	2	4	3
Temps d'entraînement	4	3	2	2	5	3
Scalabilité	3	5	2	2	4	5
Exigences Matérielles	2	3	2	3	1	4

Légende :

- 5 : Excellent
- 4 : Bon
- 3 : Moyen
- 2 : Faible
- 1 : Très Faible

2.7 Justification du choix des algorithmes

Sur la base de ce benchmark, nous avons sélectionné les algorithmes DT, RF et KNN pour notre étude :

- **Decision Tree (DT)** : pour sa simplicité, sa rapidité d'exécution et sa forte interprétabilité, ce qui est idéal pour expliquer les résultats.
- **Random Forest (RF)** : pour ses excellentes performances globales en termes de précision, de robustesse au bruit, et sa bonne capacité à généraliser.
- KNN : bien que plus coûteux en prédiction, il est intéressant pour explorer une approche basée sur la proximité et les similarités locales.

Cette combinaison d'algorithmes permet de couvrir plusieurs approches méthodologiques tout en assurant un bon compromis entre performance, interprétabilité et exigences de mise en œuvre.

Conclusion

Dans ce chapitre, nous avons détaillé les outils de machine learning exploités pour l'entraînement de notre modèle de détection, ainsi que les étapes clés du processus d'analyse et de traitement des données. Nous avons mis en œuvre des techniques rigoureuses de prétraitement, sélection de caractéristiques, équilibrage de classes, et choix d'algorithmes de classification, dans le but de maximiser la capacité du modèle à distinguer avec précision les requêtes malveillantes des comportements légitimes. Nous présentons à présent les résultats obtenus et analysons les performances de notre modèle.

Chapitre 3

Résultats et Discussions

Introduction

Dans ce chapitre, nous présentons les résultats obtenus après l'entraînement des modèles de classification sur le jeu de données NSL-KDD. Nous analysons les performances des algorithmes sélectionnés à l'aide de plusieurs métriques d'évaluation. Enfin, nous discutons la pertinence de ces résultats dans le contexte de la détection des attaques par déni de service (DoS), en mettant en évidence les points forts et les limites de chaque approche.

3.1 Métriques d'évaluation

Pour évaluer les performances des différents modèles de classification, nous avons utilisé quatre métriques couramment employées en apprentissage automatique :

- **Accuracy (Précision globale) :** mesure la proportion de prédictions correctes (positives et négatives) parmi l'ensemble des prédictions.

$$\textbf{Accuracy} = \frac{TruePositives + TrueNegatives}{TruePositives + TrueNegatives + FalsePositives + FalseNegatives} \tag{3.1}$$

- **Recall (Rappel ou Sensibilité) :** indique la capacité du modèle à identifier correctement les instances positives réelles (par exemple, les attaques détectées correctement).

$$\textbf{Recall} = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{3.2}$$

- **Precision (Précision) :** mesure la proportion des prédictions positives qui sont réellement positives. Cela permet de savoir si le modèle fait beaucoup de fausses alertes.

$$\textbf{Precision} = \frac{TruePositives}{TruePositives + FalsePositives} \tag{3.3}$$

- **F1-Score :** représente la moyenne harmonique entre la précision et le rappel.

$$\textbf{F1-Score} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.4}$$

Où :

$TruePositives$: attaques correctement détectées

$TrueNegatives$: connexions normales correctement classées

$FalsePositives$: connexions normales classées à tort comme attaques

$FalseNegatives$: attaques non détectées (classées comme normales)

3.2 Résultats

Les performances des trois algorithmes sélectionnés ont été comparées sur une sélection de 22 caractéristiques pertinentes extraites de l'ensemble de données NSL-KDD. L'objectif était d'évaluer leur efficacité à détecter les attaques par déni de service avant et après équilibrage des classes via sur-échantillonnage. Les résultats obtenus sont synthétisés dans les tableaux (cf. tableaux 3.1 et 3.2) suivants, qui présentent les performances des modèles avant et après l'application du sur-échantillonnage.

TABLE 3.1 : Résultats des prédictions avant sur-échantillonage

Algorithme	Accuracy (%)	Recall (%)	Precision (%)	F1-Score (%)
DT	92.8	84.2	86.5	85.3
KNN	90.9	82.7	84.9	83.8
RF	97.5	94.6	95.8	95.2

TABLE 3.2 : Résultats des prédictions après sur-échantillonage

Algorithme	Accuracy (%)	Recall (%)	Precision (%)	F1-Score (%)
DT	94.9	90.7	91.2	91.0
KNN	93.7	88.9	89.8	89.3
RF	98.9	98.3	98.5	98.4

L'analyse de ces résultats montre clairement que l'algorithme Random Forest surpasse les autres en termes de précision, rappel et F1-score, tant avant qu'après le sur-échantillonnage. Il se distingue notamment par sa robustesse et sa capacité de généralisation.

On observe également une nette amélioration des performances pour tous les algorithmes après l'application du sur-échantillonnage aléatoire, en particulier pour le rappel et le F1-score. Cela confirme l'importance d'un bon équilibrage des classes dans les problèmes de classification, notamment en cybersécurité, où les classes minoritaires représentent souvent des attaques rares mais critiques.

3.3 Discussions

Dans cette étude, le modèle proposé a montré des performances prometteuses, notamment après la sélection rigoureuse des caractéristiques les plus pertinentes et le choix d'algorithmes de machine learning adaptés. Ce travail nous a permis de mieux comprendre le fonctionnement des algorithmes de classification supervisée, ainsi que les étapes clés de la construction d'un système de détection d'attaques par déni de service (DoS).
L'approche adoptée a mis en évidence la capacité des modèles d'apprentissage automatique à détecter des attaques inconnues ou émergentes, ce qui représente un atout majeur par rapport aux techniques traditionnelles basées sur des signatures. En particulier, l'algorithme Random Forest s'est distingué par sa robustesse, sa précision et sa tolérance au bruit.
Cependant, cette approche n'est pas exempte de limitations. Tout d'abord, la qualité des données d'entraînement joue un rôle fondamental : des données incomplètes, bruitées ou déséquilibrées peuvent significativement nuire à la performance du modèle. Il est donc crucial de disposer de données historiques fiables et bien annotées.

Conclusion

Dans ce chapitre, nous avons présenté les résultats obtenus à l'issue de la mise en œuvre et de l'évaluation de notre modèle de détection d'attaques par déni de service. Les performances observées confirment que les algorithmes d'apprentissage automatique, en particulier le Random Forest, sont capables de détecter efficacement des attaques, y compris des variantes inconnues, à condition d'être entraînés sur des données de qualité telles que celles du jeu de données NSL-KDD.
L'utilisation de techniques de prétraitement comme le sur-échantillonnage a permis d'améliorer sensiblement la qualité des prédictions, en équilibrant les classes et en réduisant les biais liés à la distribution initiale des données.

Conclusion Générale

L'intégration d'algorithmes d'apprentissage automatique dans la détection des attaques par déni de service constitue une approche à la fois prometteuse et performante. Les résultats obtenus au cours de ce projet, notamment avec l'algorithme Random Forest, ont démontré une précision élevée atteignant près de 99%, soulignant ainsi l'efficacité de ces techniques pour identifier les comportements malveillants au sein du trafic réseau.

Cette précision remarquable montre que l'apprentissage automatique peut jouer un rôle clé dans le renforcement des systèmes de détection d'intrusion, en particulier dans des contextes où la rapidité et la fiabilité sont essentielles, comme la protection des infrastructures critiques.

Toutefois, cette étude s'inscrit dans un champ de recherche plus vaste. De nombreux défis restent à relever, notamment l'adaptabilité des modèles à des contextes réseau variés, la détection d'attaques inconnues ou évolutives, et la prise en compte des contraintes liées au temps réel et à la consommation de ressources.

En somme, cette démarche confirme que l'apprentissage automatique, s'il est bien conçu et entraîné sur des données représentatives, constitue un levier puissant pour anticiper, détecter et contrer les menaces de plus en plus sophistiquées qui pèsent sur la cybersécurité.

Bibliographie

[1] ANSSI. Comprendre et anticiper les attaques ddos, 2015.

[2] C.-A. Azencott. Introduction au machine learning, 2019.

[3] M. KHICHANE. Le machine learning avec python, de la théorie à la pratique, 2021.

Webographie

[4] https://www.zenarmor.com/docs/network-security-tutorials/what-is-denial-of-service-attack. Consulté en Aout 2024.

[5] https://www.sirar.com.sa/articles/ddos-resilience-silent-threat-silent-defense/. Consulté en Septembre 2024.

[6] https://fr.wikipedia.org/wiki/Attaque_par_déni_de_service. Consulté en Juillet 2024.

[7] https://www.crowdstrike.com/fr-fr/cybersecurity-101/artificial-intelligence/machine-learning/. Consulté en Juillet 2024.

[8] https://www.databricks.com/fr/glossary/machine-learning-models. Consulté en Janvier 2025.

[9] https://datascientest.com/. Consulté en Septembre 2024.

[10] https://www.intelligence-artificielle-school.com/ecole/technologies/quest-ce-que-scikit-learn-en-machine-learning/. Consulté en Octobre 2024.

Table des matières

Dédicace ii

Remerciements iii

Résumé iv

Abstract v

Table des figures vi

Liste des tableaux vii

Liste des acronymes viii

Introduction 1

1 Etat de l'art 3

Introduction . . . 3

1.1 Attaques par déni de service . . . 3

1.1.1 Qu'est-ce qu'une attaque par déni de service et comment fonctionne t-elle ? . . . 3

1.1.2 Formes d'attaques par déni de service . . . 4

1.1.3 Attaques par déni de service distribuée . . . 4

1.1.4 Types d'attaques par déni de service . . . 4

1.1.4.1 Attaques volumétriques . . . 4

1.1.4.2 Attaques par déni de service SYN Flood . . . 5

1.1.4.3 Attaques par amplification/réflexion . . . 6

1.1.4.4 Attaques applicatives . . . 7

1.2 Techniques de détection des attaques par déni de service . . . 8

1.2.1 Détection basée sur la signature . . . 8

1.2.2 Détection basée sur le comportement . . . 8

1.2.3 Détection basée sur basée sur l'apprentissage automatique . . . 9

1.2.4 Détection hybride . . . 9

1.3 Apprentissage automatique en cybersécurité . . . 9

1.3.1 Qu'est-ce que l'apprentissage automatique ? . . . 9

1.3.2 Types d'apprentissage automatique . . . 10

1.3.2.1 Apprentissage supervisé . . . 10

1.3.2.2 Apprentissage non supervisé . . . 10

1.3.2.3 Apprentissage semi-supervisé . . . 10

1.3.2.4 Apprentissage par renforcement . . . 10

1.3.3 Utilisation de l'apprentissage automatique en cybersécurité . . . 10
1.3.4 Avantages de l'apprentissage automatique en cybersécurité . . . 11
1.3.5 Défis et contraintes de l'apprentissage automatique . . . 12
1.4 Autres Travaux . . . 13
Conclusion . . . 13

2 Analyse et choix techniques **14**
Introduction . . . 14
2.1 Outils de développement . . . 14
2.1.1 Langage utilisé . . . 14
2.1.2 Bibliothèques utilisées . . . 14
2.1.2.1 NumPy . . . 14
2.1.2.2 Pandas . . . 14
2.1.2.3 Scikit-Learn . . . 15
2.1.2.4 Matplotlib . . . 15
2.1.3 Environnements . . . 15
2.1.3.1 Jupyter Notebook . . . 15
2.1.3.2 Kaggle . . . 15
2.2 Jeu de données . . . 15
2.3 Prétraitement des données . . . 19
2.3.1 Agrégation et nettoyage des données . . . 19
2.3.2 Encodage et normalisation des données . . . 19
2.4 Sélection des caractéristiques . . . 20
2.5 Equilibrage des données . . . 22
2.6 Choix des algorithmes . . . 23
2.7 Justification du choix des algorithmes . . . 24
Conclusion . . . 25

3 Résultats et Discussions **26**
Introduction . . . 26
3.1 Métriques d'évaluation . . . 26
3.2 Résultats . . . 27
3.3 Discussions . . . 28
Conclusion . . . 28

Conclusion **29**

Bibliographie **30**

Webographie **31**

Table des matières **32**