République algérienne démocratique et populaire
Ministre de l'enseignement supérieur et de la recherche scientifique
Université Saad Dahlab blida 1
Faculté Des Sciences
Département d'Informatique

Mémoire

Présenté pour l'obtention du diplôme de Master 2
En : Informatique
Spécialité : Systèmes Informatiques et Réseaux

Réalisé par

Meghnine Islem

Belhocine Omar

THÈME

Système d'analyse avancée des fichiers log pour neutraliser les cybermenaces.

Use case : Menaces ciblant les serveurs web.

Soutenu le Jeudi 22/06/2023 : devant le jury compposé de :

M.	CHIKHI Nacim Fateh		Présidente
Mme.	BEY Fella		Examinateur
M.	BENYAHIA Mohamed		Promoteur
Mme.	BENSIMESSAOUD Sihem	Attachée de Recherche (CERIST)	Encadreur
Mlle.	ZEMMACHE Amina	Attachée de Recherche (CERIST)	Co-Encadreur

2022/2023

Remerciements

Nous nous réservons ces quelques lignes pour exprimer notre gratitude et notre reconnaissance à tous ceux qui ont aidé à terminer ce travail, et nous le faisons avec le plus grand plaisir.

D'abord et avant tout, nous voulons exprimer notre gratitude à Dieu le Tout-Puissant pour nous avoir donné la force et l'endurance pour terminer cette tâche. Deuxièmement, nous voulons exprimer notre gratitude à nos encadreurs :

Mme BENSIMESSAOUD Sihem, Mlle ZEMMECHE Amina, et notre promoteur M. Benyahia Mohamed, méritent des remerciements particuliers pour leurs conseils et leurs assistances inestimables tout au long de la période de travail. Leurs patiences, leurs disponibilités et, surtout, leurs sages conseils nous ont aidés à réfléchir à nos idées et à organiser notre projet, qui comprenait la rédaction de ce mémoire. Notre sincère gratitude va également aux membres du jury qui ont accepté d'évaluer notre travail :

Nous tenons à exprimer notre gratitude au département de l'informatique de l'Université de Blida 1 pour leur soutien tout au long de ce programme de maîtrise ainsi qu'à l'ensemble de l'équipe Cerist, notamment Mme GUEMRAOUI Lila pour nous avoir donné l'occasion de collaborer avec eux.

Enfin, nous voudrions exprimer notre appréciation pour travailler ensemble sur ce projet.

Dédicace

Je dédis ce travail

A la personne la plus chère à mon cœur, ma mère qui m'a encouragé à poursuivre mes rêves et soutenu dans chaque étape de ma vie. Que Dieu le tout puissant veille sur elle et la protège.

A mon superbe père qui m'indique toujours la bonne voie, qui a fait de moi ce que je suis aujourd'hui, et qui fait tout pour nous rendre heureux.

A mes chers frères Anes et Redouane pour leur appui et leur encouragement.

A mes chers amis.

A mes grands-parents et à toute ma famille maternelle et paternelle.

A mon chère et fidèle binôme et amie Islem Merci de m'avoir encouragé et surtout d'avoir été toujours là pour moi dans les moments les plus dures, ainsi qu'à toute sa famille.

Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fuit de votre soutien infaillible,

Merci d'être toujours là pour moi.

Omar.

Je dédis ce travail

A la personne la plus chère à mon cœur, ma mère qui m'a encouragé à poursuivre mes rêves et soutenu dans chaque étape de ma vie. Que Dieu le tout puissant veille sur elle et la protège.

A mon superbe père qui m'indique toujours la bonne voie, qui a fait de moi ce que je suis aujourd'hui, et qui fait tout pour nous rendre heureux.

A mon cher frère Zaki pour son appui et son encouragement.

A mes chers amis.

A mes grands-parents et à toute ma famille maternelle et paternelle.

A mon chère et fidèle binôme et amie Omar Merci de m'avoir encouragé, ainsi qu'à toute sa famille.

Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fuit de votre soutien infaillible,

Merci d'être toujours là pour moi.

Islem.

Résumé

L'analyse des fichier journaux (logs) web pour la cybersécurité est une pratique essentielle pour détecter et prévenir les attaques web malveillantes. En effet, ces logs contiennent des informations détaillées sur les activités et les requêtes des utilisateurs, ce qui en fait une source précieuse d'informations pour identifier les comportements malveillants. Néanmoins, l'analyse de ces logs s'avère délicate en pratique en raison de leur volume et de leur hétérogénéité.

Dans ce travail, une solution basée sur des technique d'apprentissage supervisé et non supervisé pour détecter les attaques web à partir des fichiers log est proposée. Notre solution permet de mieux détecter les attaques, identifier les anomalies et apporter un niveau de protection élevé contre les nouvelles menaces et vulnérabilités non identifiées. Cette solution est soutenue par une interface de visualisation des données (interface de supervision) qui fournit à l'administrateur de sécurité des représentations visuelles des résultats pour une bonne prise de décision.

Mots clés : Analyse de Fichier log, Serveurs Web, Apprentissage automatique, Attaques Web.

Abstract

Analyzing web logs for cybersecurity is an essential practice for detecting and preventing malicious web attacks. Indeed, these logs contain detailed information about users' activities and requests, making them a valuable source for identifying malicious behavior. However, in practice, analyzing these logs can be challenging due to their volume and heterogeneity.

In this work, a solution based on supervised and unsupervised machine learning techniques is proposed to detect web attacks from log files. Our solution allows for improved attack detection, anomaly identification, and provides a high level of protection against new, unidentified threats and vulnerabilities. This solution is supported by a data visualization interface (supervision interface) that provides security administrators with visual representations of the results for best decision-making.

Keywords : Log File Analysis, Web Servers, Machine Learning, Web Attacks.

الملخص

يعد تحليل سجلات الويب ممارسة أساسية لاكتشاف هجمات الويب الضارة ومنعها. في الواقع ، تحتوي هذه السجلات على معلومات مفصلة حول أنشطة المستخدمين وطلباتهم ، مما يجعلها مصدرًا قيمًا لتحديد السلوك الضار. ومع ذلك ، من الناحية العملية ، يمكن أن يكون تحليل هذه السجلات أمرًا صعبًا نظرًا لحجمها وعدم تجانسها.

في هذا العمل ، تم اقتراح حل يعتمد على تقنيات التعلم الآلي الخاضعة للإشراف وغير الخاضعة للإشراف لاكتشاف هجمات الويب من خلال تحليل سجلات الويب. يسمح الحل الذي نقدمه باكتشاف الهجمات المحسّنة ، وتحديد الأخطاء ، ويوفر مستوى عاليًا من الحماية ضد التهديدات ونقاط الضعف الجديدة غير المحددة. هذا الحل مدعوم بواجهة عرض بيانات (واجهة إشراف) تزود مسؤولي الأمن بتمثيلات مرئية للنتائج لاتخاذ أفضل القرارات.

باستخدام خوارزميات متقدمة ونماذج تعلم آلي، يمكن لأنظمة تحليل ملفات تسجيل أيضًا التنبؤ بالهجمات المستقبلية من خلال تحديد الاتجاهات وأنماط سلوك الجرائم السيبرانية. يساعد ذلك على تعزيز أمان خادم الويب من خلال اتخاذ تدابير وقائية ضد التهديدات قبل حدوثها.

الكلمات المفتاحية تحليل ملف تسجيل ، خوادم الويب ، التعلم الآلي ، هجمات الويب.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

De nos jours, les applications Web sont à la fois beaucoup plus répandues et beaucoup plus complexes qu'auparavant. Le développement du Web dynamique et la richesse fonctionnelle qu'offrent les nouvelles technologies du Web permettent de répondre à un grand nombre de besoins. Néanmoins, cette richesse fonctionnelle s'accompagne d'une complexité grandissante et cette progression engendre une multiplication du nombre de vulnérabilités informatiques, offrant une surface d'attaque parfaite pour les attaquants informatiques.

Plusieurs services, mécanismes, outils et procédures sont utilisés actuellement pour contrer ces menaces , parmi les plus répondus : l'analyse des fichiers log. En effet, les fichiers log sont une mine d'information pour les professionnels de la sécurité. En utilisant ces informations, on peut détecter les activités malveillantes, identifier les vulnérabilités et les différents types d'attaques. Néanmoins, ces fichiers n'ont de valeur que lorsqu'ils sont analysés d'une manière approfondie et suffisamment élaborée, afin de détecter les indicateurs d'attaques avancées.

Les nouvelles solutions basées sur l'analyse des fichiers logs pour détecter les attaques ciblant les applications Web ne cessent de se développer. Pendant de nombreuses années, les experts analysaient manuellement ces fichiers pour détecter les intrusions, cette approche s'est avérée rapidement obsolète et non pratique vu le grand volume de données à analyser qui accroit exponentiellement.

Pour cette raison, des systèmes qui permettent une analyse rapide d'énorme quantité de données ont émergé. Ces systèmes peuvent identifier une entrée de log comme une attaque ou entrée valide en se basant sur des signatures d'attaques préalablement définies par les experts. L'inconvénient de cette approche est que seules les attaques reconnues par les signatures seront détectées, ce qui nécessite une mise à jour régulière de la base de signatures, par conséquent, elle n'est pas en mesure de détecter les nouvelles attaques. Le développement de l'apprentissage automatique a ouvert de son coté, de nouvelles frontières en rendant ce processus plus intelligent et généralisable.

En effet, l'avancement des technologies d'apprentissage automatique nous permet de réa-

liser des systèmes plus sophistiqués utilisant des méthodes supervisées capables d'apprendre les caractéristiques des attaques Web à partir des journaux étiquetés, et de généraliser les connaissances acquises sur des données jamais vues, sans avoir besoin des signatures prédéfinies des attaques. En outre, les approches d'apprentissage non supervisées sont capables de détecter des anomalies dans les fichiers journaux en utilisant uniquement des données valides générées à partir de n'importe quel serveur sans même avoir besoin d'un expert du domaine.

C'est dans ce contexte que s'inscrit l'objectif de ce projet, qui consiste à la conception et le développement d'une application dédiée à l'analyse avancée des fichiers logs dans le but de : Mieux détecter les attaques, identifier les anomalies et apporter un niveau de protection élevé contre les nouvelles menaces et vulnérabilités non identifiées.

La solution envisagée tente d'augmenter la capacité de détection par la combinaison des algorithmes d'apprentissage automatique supervisés et non supervisés.

PLAN DU MÉMOIRE

Outre cette introduction, le présent manuscrit est subdivisé en quatre (04) chapitres. Nous précisons ici l'objectif de chacun d'entre eux avec bref descriptif de leur contenu.

- **Chapitre 1 :** Un chapitre introductif qui présente une brève introduction au domaine de la sécurité des applications web.
- **Chapitre 2 :** Expose une vue d'ensemble des concepts qui découlent du domaine de l'analyse des fichiers logs, et il aborde aussi l'application de l'apprentissage automatique dans le domaine de l'analyse des fichiers logs où nous avons étudié certains travaux récents.
- **Chapitre 3 :** Détaille notre approche d'analyse de fichiers log. Notre méthodologie y est décrite et les différentes étapes y sont présentées.
- **Chapitre 4 :** Présente l'environnement de développement , les langages et les algorithmes utilisés. Il présente aussi, le produit final et les résultats obtenus.
- **Conclusion et perspectives :** Finalement nous concluons ce rapport en proposant une synthèse de notre étude ainsi qu'un ensemble de perspectives liées à la continuité du travail.
- **Annexes :** Cette partie aborde des descriptions détaillées sur quelques notions rencontrées dans les différents chapitres du mémoire.

Chapitre 1

Sécurité des Applications Web

1.1 Introduction

Avec le développement croissant d'Internet, les applications Web sont devenues une composante essentielle des systèmes d'informations modernes. Elles offrent une multitude de services et de fonctionnalités tels que les achats en ligne, la communication, le partage d'informations, le travail collaboratif, les services bancaires, les réseaux sociaux,etc. Néanmoins, l'exposition de ces applications sur Internet engendre continuellement de nouvelles formes de menaces qui peuvent mettre en péril la sécurité de l'ensemble du système d'information.

En effet, les applications web sont les passerelles potentielles pour un large éventail d'attaques qui représentent une menace sérieuse à la fois pour les organisations qui les déploient et pour les utilisateurs qui y accèdent : les pirates peuvent exploiter les vulnérabilités des applications Web pour voler des données sensibles, compromettre les comptes d'utilisateurs et même prendre le contrôle de systèmes entiers. Par exemple, en février 2022, l'opérateur téléphonique croate A1 Hrvatska a révélé une violation de données qui a affecté 10% de ses clients, soit environ 200 000 personnes [1].

Par conséquent, les attaques d'applications Web représentent une grande menace à la sécurité d'une organisation, effectuer une analyse complète pour identifier les menaces, les attaques, et les vulnérabilités peut aider à se prémunir contre les menaces qu'elle peut faire peser sur le système.

Dans ce chapitre, nous allons donner un aperçu sur les principales vulnérabilités et attaques ciblant les applications web. Ensuite, nous allons présenter les solutions existantes permettant d'assurer leur protection, et nous discuterons leurs limites.

1.2 Vulnérabilités des Applications Web

Étant exposées au public, les applications web ont naturellement une surface d'attaque étendue et éventuellement des fonctionnalités avec un grand nombre d'éléments variés potentiellement vulnérables. Même sur un serveur Web sécurisé s'exécutant sur un système

d'exploitation réputé sûr, des failles de sécurité (vulnérabilités) peuvent exister car elles sont la plupart du temps dues à des fautes de programmation de l'application elle-même ou des erreurs de configuration du serveur l'hébergeant. En effet, il existe une grande variété de vulnérabilités et attaques visant les applications Web. Toutefois certaines sont plus connues et plus dangereuses que d'autres.

Il existe plusieurs bases de données répertoriant ces vulnérabilités avec des statistiques indiquant leur importance relative. Nous citons par exemple les bases de données de vulnérabilités telles que CVE (Common Vulnerabilities and Exposures) [2], NVD (National Vulnerability Database) [3] ou VUPEN (Vulnerability Penetration testing) [4]. Ces bases de données répertorient tous types de vulnérabilités, incluant celles ciblant les serveurs et les applications Web.

Toutefois, la multiplication des vulnérabilités et des attaques web, s'est traduite aussi par la proposition de taxonomies et de classifications pour les vulnérabilités et les attaques web les plus répandues. Parmi ces communautés, nous citons OWASP (Open Web Application Security Project)[5] et WASC (Web Application Security Consortium) [6].

Les membres du "Web Application Security Consortium" (WASC) ont créé ce projet pour développer et promouvoir une terminologie standard décrivant les problèmes de sécurité des applications Web et permettant aux développeurs d'applications, experts en sécurité, développeurs de logiciels et les consultants en sécurité, d'utiliser un langage commun pour interagir entre eux.

D'un autre côté, l'Open Web Application Security Project (OWASP) a défini dans l'un de ses projets nommé "TOP 10" les dix classes de vulnérabilités Web les plus critiques [7]. L'objectif principal du Top 10 de l'OWASP est d'informer les développeurs, concepteurs, managers, et les entreprises au sujet des conséquences des faiblesses les plus importantes inhérentes à la sécurité des applications Web. Le Top 10 fournit également des techniques de base pour se protéger contre ces vulnérabilités.

Quelques types de vulnérabilités, sont listés dans ce qui suit :

1. **Filles D'injections :** qui permettent à un attaquant d'injecter et d'exécuter du code malveillant, ce qui peut entraîner des conséquences graves telles que la compromission des données, la prise de contrôle du système ou des attaques à grande échelle. On distingue généralement différentes familles d'injection en fonction du protocole visé (SQL, XML, XPATH, LDAP) ou du type d'injection par exemple injection d'un programme exécutable dans le cas des failles de type "OsCommanding", ou chargement d'un fichier dans le cas des failles de type "FileUpload", qui concernent les sites permettant à leurs utilisateurs de charger des fichiers, tels que des photos par exemple.
2. **Code source non sécurisé :** Ce terme désigne le code source qui contient des failles de sécurité qui peuvent être exploitées par des attaquants pour accéder à des informations sensibles, altérer des données ou causer des dommages au système. Ces failles

peuvent résulter de pratiques de codage faibles telles que l'utilisation de mots de passe en clair, l'absence de validation des entrées utilisateur, l'usage de bibliothèques tierces peu fiables ou vulnérables, la transmission de données sensibles en clair, des bogues de sécurité dans le code ou des erreurs de configuration.

3. **Stockage cryptographique non sécurisé :** Cette vulnérabilité fait référence à des méthodes de stockage de données sensibles, telles que les mots de passe des utilisateurs, utilisant un algorithme de hachage obsolète ou faible pour le chiffrement, ou l'application d'une technique de substitution incorrecte [34].
4. **Mauvaise configuration de sécurité (Security Misconfiguration) :** La sécurité mal configurée est une vulnérabilité courante qui résulte d'une mauvaise configuration des paramètres de sécurité sur un système ou une application. Cette vulnérabilité peut être causée par l'utilisation de configurations par défaut, une configuration manuelle incorrecte ou la négligence des mises à jour de sécurité. Les exemples typiques de la sécurité mal configurée incluent des mots de passe faibles, le non-respect des mises à jour des systèmes et des applications, la non-configuration des pare-feu, l'activation de fonctionnalités inutiles, la non-validation des entrées utilisateur et la mauvaise gestion des erreurs .
5. **Référence directe non sécurisée à un objet (Unsecured Direct Object Reference) :** C'est une vulnérabilité de sécurité qui se produit lorsque les développeurs d'une application Web exposent directement des références internes à des objets, tels que des identifiants ou des clés, dans les URL, les formulaires ou d'autres paramètres. Cela permet aux attaquants d'accéder à des ressources ou à des fonctionnalités auxquelles ils ne devraient pas avoir accès.

1.3 Attaques Courantes sur les Applications Web

Il existe une grande variété d'attaques visant les applications Web. Toutefois certaines sont plus connues et plus dangereuses que d'autres. Dans ce qui suit, nous allons présenter les attaques Web les plus répandues :

1. **SQL Injection :** SQL injection est une attaque visant à exploiter les vulnérabilités d'injection des applications web pour accéder à des informations confidentielles stockées dans une base de données, apporter des modifications, ou encore supprimer des tables de la base de données. L'attaquant utilise des requêtes SQL malveillantes pour insérer du code dans l'application web et ainsi accéder à des données sensibles ou compromettre le système hôte. Les attaques par injection SQL peuvent prendre différentes formes, telles que les injections basées sur le temps ou sur les erreurs [35].
2. **SQL Injection blind :** Blind SQL (Structured Query Language) est une sorte d'attaque SQL Injection dans laquelle l'attaquant pose des questions vraies ou fausses à la base de données, puis décide de la réponse en fonction de la réaction de l'application.

Cette approche est souvent employée lorsque l'application Web est configurée pour afficher des messages d'erreur génériques mais n'a pas atténué le code vulnérable à l'injection SQL [36].

3. **LFI / RFI : (Local File Inclusion/Remote File Inclusion) :** C'est une méthode d'attaque visant à accéder à des fichiers sur un serveur distant en exploitant une vulnérabilité d'inclusion présente dans une application web. Cette attaque se compose de deux variantes, l'attaque LFI qui consiste à exploiter une faille de sécurité permettant l'inclusion d'un fichier déjà présent sur le serveur qui héberge l'application ciblée par l'attaque. L'objectif de l'attaquant sera de lire des fichiers sensibles, contenant des informations critiques comme des fichiers de configuration par exemple [37].
   Par contre l'attaque RFI vise à inclure un fichier distant sur le serveur de la victime (remote = distant). Pour ce faire l'attaquant va inclure l'URL menant à son fichier malveillant dans un paramètre vulnérable. Le fichier sera exécuté par le serveur de la victime [37].
4. **XSS (Cross-Site Scripting) :** Souvent connu sous le nom d'injection HTML, est un sous-ensemble de l'injection HTML. XSS est le problème de sécurité des applications en ligne le plus répandu et le plus dangereux. Les défauts XSS se produisent chaque fois qu'une application accepte les données fournies par l'utilisateur et les envoie à un navigateur Web sans validation ni codage préalable du contenu [38].
   XSS permet aux attaquants d'exécuter des scripts malveillants dans le navigateur de la victime afin de rediriger les sessions de l'utilisateur, de détourner les sites Web, d'insérer du contenu malveillé, de mener des attaques de phishing et de prendre le contrôle du navigateur d'un utilisateur en utilisant des script malveilleux (Malware Scripting). Le script malveillant est souvent écrit en JavaScript, bien que tout language de programmation pris en charge par le navigateur de la victime est une cible potentielle pour cette attaque [38].
5. **Command Injection :** L'objectif d'une attaque par injection de commande est d'exécuter des instructions arbitraires sur le système d'exploitation hôte via une application sensible. Lorsqu'une application envoie des données dangereuses fournies par l'utilisateur (formules, cookies, en-têtes HTTP, etc.) à une coque système, des attaques d'injection de commande sont concevables. Dans cette attaque, les instructions du système d'exploitation fournies par l'attaquant sont souvent exécutées avec les privilèges de l'application susceptible. Une validation d'entrée inadéquate rend les attaques d'injection de commande viables. [39].

1.4 Mécanismes de Protection des Applications Web

Différentes méthodes et techniques peuvent être mises en œuvre par les développeurs et les administrateurs en charge de la sécurité informatique pour faire face aux diverses

menaces qui visent les applications Web. Dans la suite nous présentons quelques exemples de moyens permettant d'assurer leur sécurité :

1.4.1 Pare-feu applicatifs Web (Web Application Firewall, WAF)

Un pare-feu applicatif web (ou WAF) est un dispositif de sécurité qui surveille et filtre le trafic HTTP/HTTPS entrant et sortant des applications web. Il est conçu pour protéger les applications web contre les attaques et les vulnérabilités spécifiques aux applications, telles que les injections SQL, les attaques par cross-site scripting (XSS), les attaques par déni de service (DoS), etc. Il protège généralement les applications Web des attaques notamment de type falsification de site croisé, XSS, d'inclusion de fichier et d'injection SQL [8].
Le fonctionnement d'un pare-feu applicatif web repose généralement sur des règles de sécurité prédéfinies et sur l'inspection du trafic des applications web en temps réel. Voici quelques fonctionnalités et avantages clés des pare-feu applicatifs web :

- **Filtrage des requêtes et des réponses :** Le WAF analyse le trafic des applications web en inspectant les requêtes entrantes et les réponses sortantes. Il filtre les requêtes malveillantes ou suspectes, bloquant ainsi les attaques connues et inconnues.
- **Détection des attaques basée sur des signatures :** Le WAF utilise une base de données de signatures d'attaques connues pour identifier les tentatives d'exploitation. Il compare le trafic entrant avec les signatures d'attaques et bloque les requêtes correspondantes.
- **Détection des anomalies :** En plus de la détection basée sur des signatures, le WAF peut également utiliser des mécanismes de détection d'anomalies pour identifier des schémas de trafic inhabituels ou des comportements anormaux qui pourraient indiquer une attaque en cours.
- **Contrôle d'accès :** Le WAF peut mettre en place des politiques de contrôle d'accès pour restreindre l'accès aux applications web en fonction de critères tels que l'adresse IP, le pays d'origine, l'heure de la journée, etc.
- **Rapports et journaux :** Les WAF génèrent généralement des rapports détaillés et des journaux d'activité pour permettre la surveillance, l'analyse et la réponse aux incidents de sécurité.

1.4.2 Système de détection d'intrusions (IDS)

Un système de détection d'intrusion (IDS pour Intrusion Détection System) est un outil qui analyse et surveille les activités malveillantes dans un réseau informatique ou un système, et envoie des alertes aux administrateurs lorsqu'une telle activité est découverte, afin qu'ils prennent des mesures rapides pour l'arrêter [9] Selon les méthodes de détection, il existe deux principales catégories des IDS, la première basée sur la reconnaissance des signatures d'attaques et la deuxième, basée sur la détection d'anomalies.

- **IDS basé sur les signatures :** Ce type d'IDS utilise des bases de données de signatures d'attaques connues pour comparer le trafic réseau ou les journaux système avec ces signatures. Lorsqu'une correspondance est détectée, une alerte est déclenchée pour signaler une attaque potentielle.
- **IDS basé sur la détection d'anomalies :** Ce type d'IDS utilise des algorithmes et des modèles pour analyser les schémas de trafic, les journaux système et les activités des utilisateurs afin de détecter des comportements anormaux. Lorsque des comportements suspects sont identifiés, une alerte est générée pour signaler une activité potentiellement malveillante.

1.4.3 Scanner de Vulnérabilités

Les scanners de vulnérabilité sont des outils automatisés qui permettent de détecter les vulnérabilités et les faiblesses potentielles dans les systèmes informatiques, les réseaux ou les applications et qui pourraient être exploitées par des attaquants. Ils permettent également de classifier ces vulnérabilités selon le risque qu'elles présentent [10]. L'analyse des vulnérabilités est une pratique courante sur les réseaux d'entreprise pour améliorer la sécurité de l'entreprise. Il existe deux classes d'analyses de vulnérabilités : les analyses internes et les analyses externes.

- **Analyses de vulnérabilité externes :** ces scans sont effectués depuis l'extérieur du réseau ou du système. Ils sont faits pour tester la force du système de sécurité de l'extérieur, et pour déterminer l'exposition aux attaques des serveurs accessibles depuis Internet.
- **Analyses de vulnérabilité internes :** ces analyses sont effectuées depuis l'intérieur du réseau. Elles sont faites pour tester jusqu'à où les attaquants peuvent aller à l'intérieur du système s'ils ont réussi à accéder au réseau local.

Les scanners de vulnérabilités peuvent être utilisés dans des objectifs illicites, où les attaquants utilisent les mêmes outils pour trouver les failles dans les systèmes des entreprises pour les exploiter à leur avantage.

1.5 Limites des mécanismes de protection existants

Bien que les mécanismes de protection des applications web existants soient essentiels pour renforcer la sécurité, ils présentent également certaines limites. Nous citons :

- Les mécanismes de protection des applications web existants, tels que les pare-feu applicatifs web (WAF), peuvent générer des faux positifs (bloquer des requêtes légitimes) ou des faux négatifs (laisser passer des attaques). Cela peut entraîner des inconvénients pour les utilisateurs légitimes ou permettre aux attaques de passer inaperçues.

- Ces mécanismes sont conçus pour détecter et bloquer les attaques connues. Cependant, de nouvelles vulnérabilités peuvent émerger régulièrement, ce qui peut permettre aux attaquants de contourner les mécanismes existants et d'exploiter les failles de sécurité inconnues [11].
- Dépendance des mises à jour : Les mécanismes de protection des applications web nécessitent des mises à jour régulières pour maintenir leur efficacité contre les nouvelles menaces. Si les mises à jour de sécurité ne sont pas appliquées de manière adéquate, les applications restent vulnérables aux attaques connues.
- Complexité des applications : Les applications web modernes sont souvent complexes, avec de multiples couches, frameworks, modules et dépendances. Cette complexité accrue rend plus difficile la mise en place de mécanismes de protection efficaces et peut créer des vulnérabilités potentielles supplémentaires.
- Ressources limitées : La mise en place et la gestion des mécanismes de protection des applications web nécessitent des ressources humaines, financières et techniques. Les organisations disposant de ressources limitées peuvent rencontrer des difficultés pour mettre en œuvre et maintenir efficacement ces mécanismes de protection.

Pour contrecarrer ces limites, les organismes ont adopté de nouvelles technologies afin d'améliorer la sécurité au fur et à mesure de l'évolution des attaques. Dans ce contexte, plusieurs applications de cybersécurité ont été développées notamment celles basées sur l'analyse des fichiers logs en utilisant l'apprentissage automatique.

1.6 Conclusion

Dans ce chapitre nous avons survolé brièvement le domaine de la sécurité des applications Web en exposant les différentes vulnérabilités et attaques, ainsi que les différents mécanismes et contre-mesures utilisés pour la protection des applications web. Nous avons essayé de mettre l'accent sur les attaques web ainsi que les limites des systèmes de protection existants face aux nouveaux défis.

Nous avons évoqué la nécessité d'avoir recours à des analyses plus approfondies, notamment celles basées sur les techniques de l'apprentissage automatique pour analyser les fichiers log. Dans le prochain chapitre, nous allons introduire le concept des fichiers logs et les techniques issues de l'apprentissage automatique pour les analyser.

Chapitre 2

Analyse des Fichiers Logs Basée sur l'Apprentissage Automatique

2.1 Introduction

Aujourd'hui, tous les événements affectant les différentes composantes d'une infrastructure informatique (machines, serveurs, applications, systèmes, réseaux ...etc.) sont enregistrés dans un fichier appelé journal (ou log en anglais). En général, les enregistrements y sont archivés par ordre chronologique permettant ainsi de garder une empreinte de tous les évènements produits. De ce fait, l'analyse de cette mine d'informations peut s'avérer d'une importance cruciale pour différents types de besoins, notamment pour la cyber-sécurité. Ces journaux constituent l'un des aspects les plus importants de l'administration et de la surveillance des systèmes informatiques. Ils permettent aux administrateurs de repérer et découvrir les problèmes qui peuvent mettre en danger la sécurité et la disponibilité du système. Pour faire face aux différents défis liés à la manipulation des fichiers log, l'apprentissage automatique constitue une solution prometteuse.

Ce chapitre est divisé en deux parties :

La première partie constitue une introduction aux fichiers logs, où nous allons présenter les différents concepts relatifs à ce domaine, en mettant l'accent sur leur apport pour la cyber sécurité, et sur la façon dont nous pouvons les analyser efficacement, en introduisant les défis rencontrés dans l'analyse.

La deuxième partie mettra en évidence l'application de l'apprentissage automatique (machine learning) dans le domaine de l'analyse des fichiers logs. D'abord nous donnons un aperçu des concepts fondamentaux du machine learning (ML), ensuite nous expliquerons le processus d'analyse des logs basé sur ML. Nous terminerons le chapitre avec une étude comparative des techniques d'analyse des logs web basées sur ML.

2.2 Fichiers Logs

les fichiers logs permettent de stocker un historique des événements survenus sur un serveur, un ordinateur ou une application. Les informations contenues permettront ensuite de mieux comprendre les usages, résoudre les problèmes rencontrés et améliorer les performances [72].

2.2.1 Concepts de base

Dans ce qui suit, nous présentons les définitions de quelques concepts utilisés dans notre mémoire [12] :

- **Evènement :** C'est une occurrence unique dans un environnement, impliquant généralement une tentative de changement d'état, il inclut généralement une notion de temps d'occurrence et tous les détails explicitement liés à l'évènement ou à l'environnement pouvant aider à comprendre ses causes ou ses effets.
- **Champ d'événement :** Décrit une caractéristique d'un événement, par exemple : la date, l'heure, l'adresse IP source, et l'identification de l'hôte.
- **Entrée de journal ou enregistrement d'événement :** C'est une collection de champs d'événement qui, ensemble, décrivent un seul événement.
- **Journal :** Un journal est formé par la collection d'enregistrements d'évènement.

La relation entre les termes décrits ci-dessus se résume comme suit : Un événement est décrit via ses champs, qui sont répertoriés dans un ou plusieurs enregistrements, ces derniers sont collectés dans un journal (log) [12].

2.2.2 Sources de journaux

Les fichiers journaux sont divers et contiennent des informations provenant de différentes sources. Parmi ces sources, nous citons [13] :

- Les fichiers logs provenant des logiciels de sécurité tels que les pare-feu, IDS/IPS, anti-virus et autres équipements de surveillance du réseau.
- Les fichiers logs provenant des systèmes d'exploitation qui contiennent des événements enregistrés par les composants du système tels que les modifications de périphérique, les pilotes de périphérique, les modifications du système, etc.
- Les fichiers logs provenant des applications/serveurs tels que les serveurs web. Chaque fois qu'un navigateur demande une ressource (HTML, images, ...), le serveur ajoute une ligne dans le journal.

2.2.3 Contenu d'un message log Web

Lorsque nous utilisons des applications web, nous sommes exposés à des menaces et des attaques. Contre ces menaces, les responsables de sécurité analysent les fichiers journaux qui répertorient toutes les actions effectuées. Ces fichiers journaux communément appelés "access.log" génèrent différents types d'informations, stockées de manière standardisées de façon à ce qu'il soit possible de procéder à leur analyse. Un fichier log web offre des informations sur l'utilisateur, son matériel, la date et l'heure de la requête, la page requise, le code de la réponse, la page de référence ainsi que quelques informations sur le protocole d'échange. Voici un exemple d'une ligne d'un fichier log web :

```
41.109.239.5 - - [24/Apr/2017 :00 :01 :47 +0100] "GET /index.php/fr/ HTTP/1.1"
200 97158 "https ://www.google.com/" "Mozilla/5.0 (Windows NT 6.1 ;WOW64)
AppleWebKit/537.36 (KHTML,like Gecko) Chrome/56.0.2924.87 Safari/537.36
OPR/43.0.2442.1144"
```

Décortiquons cette ligne :

- **41.109.239.5 :** représente l'adresse IP du client qui a effectué la demande de ressource HTTP.
- **- - :** le trait d'union indique que l'information est manquante. Le premier montre l'identité de la machine client et le second l'identifiant utilisé par le client pour l'authentification.
- **[24/Apr/2017 :00 :01 :47 +0100] :** la date et l'heure à laquelle le visiteur à surfer sur l'application.
- **"GET /index.php/fr/ HTTP/1.1" :** la méthode d'appel (GET), l'url ou la page visitée (index.php), et le protocole utilisé (ici HTTP 1.1) ce qui représente la requête.
- **200 :** le code de la réponse envoyée par le serveur au client, indiquant le succès ou l'échec de la requête HTTP, dans notre exemple represente le succès.
- **97158 :** la taille en octets de données envoyés par le serveur, ne comprenant pas l'en-tête HTTP.
- **Https ://www.google.com/ :** c'est le « Referer », c'est-à-dire la page que le client a visité avant de venir sur la page « index.php ».
- **"Mozilla/5.0 (Windows NT 6.1 ; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36 OPR/43.0.2442.1144" :** le User-Agent du navigateur du visiteur. Grâce à ce morceau, nous pouvons connaître l'OS et le navigateur de l'utilisateur ici c'est Chrome 56.0 sur Windows NT 6.1.

2.2.4 Analyse des fichiers logs pour la cyber-sécurité

La technique consistant à extraire des informations pertinentes à partir des fichiers journaux est connue sous le nom d'analyse des journaux. Elle est essentielle dans le secteur

informatique, où pratiquement tous les systèmes et services créent d'énormes journaux.
Les fichiers logs sont un élément crucial de la cybersécurité car ils fournissent une source d'informations importante pour les professionnels de la sécurité. L'objectif de leur analyse étant de détecter les activités malveillantes, d'identifier les vulnérabilités et les différents types d'attaques, ainsi que pour détecter les erreurs de configuration et les anomalies de performance.
Les informations pertinentes contenues dans les fichiers log représentent les éléments de preuve nécessaires à une enquête [44]. C'est la seule information que l'attaquant laisse derrière lui après son introduction dans le réseau, ce qui représente l'empreinte de l'attaquant, c'est pour cela que les fichiers journaux sont une partie importante dans l'investigation numérique (Digital Forensic) [24]. C'est le seul moyen pour identifier l'attaquant afin qu'il puisse être poursuivi .

2.2.5 Défis dans l'analyse des fichiers Log

L'analyse des fichiers de logs présente certains défis, notamment :

a. **L'analyse manuelle des journaux est difficile pour les grands systèmes :** Afin de détecter les problèmes en analysant manuellement les journaux, l'expert ou l'administrateur doit comprendre les comportements du système en question et les logs qu'il génère. Mais le nombre des applications existantes dans le système est assez important et leurs journaux sont très différents et contiennent beaucoup d'informations. Il s'avère donc difficile de trouver un tel expert ayant toutes les connaissances nécessaires, En outre, l'analyse manuelle prend du temps et est inefficace. Par conséquent, leur analyse automatisée est souhaitable [17].

b. **L'utilisation de la programmation explicite est difficile :** Un système à grande échelle est souvent constitué de plusieurs composants, chacun d'entre eux pouvant être construit à l'aide d'un langage de programmation différent, les informations de journal des différents composants peuvent varier considérablement. Étant donné qu'il n'existe pas de norme de structure de sortie standard pour tous les fichiers journaux, la mise en œuvre d'un analyseur de journaux unique pour un système à grande échelle via la programmation explicite constitue un sérieux défi [17].

c. **L'apprentissage automatique peut aider :** Comme il semble impossible de traiter les journaux manuellement ou par le biais de la programmation explicite, les techniques d'apprentissage automatique peuvent être utilisées pour automatiser cette analyse et généraliser les connaissances acquises à partir de l'ensemble de données d'entraînement. L'utilisation de l'apprentissage automatique permet de découvrir des problèmes qui se cachent dans les journaux, même si ces derniers présentent des formats et des contenus différents, sans nécessiter de programmation explicite ni de connaissances spécialisées en analyse de journaux [17].

Dans la section suivante, nous allons étudier le processus d'analyse des fichiers log basé sur les techniques de l'apprentissage automatique.

2.3 Apprentissage Automatique « Machine Learning »

L'intelligence artificielle (IA) est un vaste domaine de recherche qui constitue « l'ensemble de théories et de techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine ». Il est constitué de plusieurs branches, dans notre étude, nous nous concentrerons sur une seule branche, à savoir l'apprentissage automatique.

2.3.1 Définition de l'Apprentissage Automatique

L'apprentissage automatique (ML en anglais pour Machine Learning) est défini en 1959 par Arthur Samuel comme suit : « L'apprentissage automatique est la discipline donnant aux ordinateurs la capacité d'apprendre sans qu'ils soient explicitement programmés. » [18]. C'est la branche de l'IA qui vise à permettre aux machines d'apprendre à partir d'un nombre important d'exemples grâce à des modèles mathématiques. L'objectif de cette approche est d'extraire les informations pertinentes d'un ensemble de données d'entraînement afin d'obtenir les paramètres optimaux pour un modèle, qui permettront d'obtenir les meilleures performances lors de l'exécution de la tâche assignée au modèle (pouvoir appliquer ce qu'il a appris aux cas futurs) [19].

2.3.2 Notions élémentaires

2.3.2.1 Jeu de données

Une donnée est un enregistrement au sens des bases de données, ou « instance » (terminologie orientée objet au sens informatique). Une donnée est caractérisée par un ensemble d'attributs. Un jeu de données est une collection de données organisées de manière structurée, généralement sous forme de tableau ou de fichier. Il est utilisé dans l'apprentissage automatique pour entraîner un modèle en servant d'exemples à l'algorithme d'apprentissage pour apprendre et faire des prédictions. Le jeu de données complet est souvent réparti en 3 ensembles :

a. **L'ensemble d'apprentissage :** ou population d'entraînement constitue l'ensemble des candidats ou exemples (textes, images, BD,...) utilisés pour générer le modèle d'apprentissage.
b. **L'ensemble de validation :** Il peut être utilisé lors de l'apprentissage (comme sous population de l'ensemble d'apprentissage) afin de valider le modèle et d'éviter le sur-apprentissage.

c. **L'ensemble de test :** Cet ensemble est utilisé à la fin du processus d'entraînement, il est constitué des candidats sur lesquels sera appliqué le modèle d'apprentissage (pour tester et corriger l'algorithme), les données de cet ensemble sont inconnues du modèle et servent à tester son exactitude.

2.3.2.2 Classe

Une classe enseigne sur un ensemble d'objets ou échantillons de même nature et ayant le même vecteur descripteur comme modèle de description.

2.3.2.3 Classification

La classification est la tâche consistant à attribuer une classe à une donnée qu'on veut classer, ou autrement dit, à assigner une donnée à une classe de données.

2.3.3 Types d'Apprentissage Automatique

En fonction de la nature du problème traité et des données disponibles, l'apprentissage automatique peut être classé dans les catégories suivantes : supervisé, non supervisé, semi supervisé et par renforcement.

2.3.3.1 Apprentissage supervisé

L'apprentissage supervisé nécessite une base d'apprentissage où chaque instance est préalablement étiquetée avec sa classe respective. Les différentes étiquettes indiquent les classes d'appartenance. Les observations étiquetées utilisées pour construire le modèle sont appelées données d'entraînement. Le modèle créé fait des prédictions quant à la classe de tout nouveau cas inconnu (données de test) qui ne fait pas partie de l'ensemble de données d'entraînement. Une classification supervisée s'opère donc en deux phases : la phase d'entraînement et la phase de détection [20].
En pratique, il s'agit de fournir à l'algorithme un ensemble de caractéristiques X (décrivant les données d'entraînement) associés à des sorties Y (étiquettes), et l'algorithme va trouver une fonction de mapping entre les entrées X et les sorties Y, la fonction de mapping décrivant une relation entre X et Y s'appelle un modèle de prédiction [21].

Un exemple consiste à prédire le prix d'une voiture à partir des valeurs d'un certain nombre d'attributs ou variables qu'on appelle caractéristiques d'une observation ou « features » en anglais. Ces variables peuvent être le kilométrage, l'âge, la marque, etc [21].

Machine Learning avec supervision peut se subdiviser en deux types [22] :

- **Classification :** La variable de sortie est une catégorie (qualitative).
- **Régression :** La variable de sortie est une valeur spécifique (quantitative).

Les principaux algorithmes avec supervision sont les suivants : arbres décisionnels, forêts aléatoires, méthode du k plus proche voisin (KNN), régression linéaire, classification naïve bayésienne, machine à vecteurs de support (SVM), et boosting des gradients [22].

2.3.3.2 Apprentissage non supervisé

L'apprentissage non supervisé consiste en la conception d'un modèle structurant l'information, Il implique un entraînement basé sur des données qui n'ont pas d'étiquettes ou un résultat spécifique défini. Ce type d'apprentissage peut être utilisé pour découvrir des clusters formés par l'ensemble des données. Il s'agit de partitionner les instances en différents ensembles homogènes tel que [22] :

- Les instances d'un même cluster partagent des caractéristiques communes, qui correspondent à des critères de proximité que l'on définit le plus souvent grâce à des mesures de distance entre les paires d'instances.
- Les instances appartenant à des clusters différents soient différenciées (éloignées).

Dans l'apprentissage non supervisé, on donne à l'algorithme des données éventuellement non structurées, et on le laisse trouver une sorte de structure dans nos données, c-à-d trouver des relations entre ces données. Lorsqu'on fait du clustering, on ne connaît pas à l'avance quelles sont les classes des données manipulées, c'est d'ailleurs souvent ce qu'on cherche à déterminer.
Dans les méthodes d'apprentissage non supervisé, il existe plusieurs classes comme les méthodes basées sur la densité (LOF (local Outlier Factor)), les méthodes basées sur la distance (k-means), les méthodes de réduction de la dimensionnalité, etc.

2.3.3.3 Apprentissage semi-supervisé

L'apprentissage semi-supervisé utilise un jeu de données partiellement étiquetées qui comporte quelques données étiquetées et beaucoup de données non étiquetées. Il vise à résoudre les problèmes avec des données non étiquetées à l'aide de l'ensemble d'informations étiquetées. Cette approche est utile lorsque l'étiquetage des données est difficile ou coûteux, ou lorsqu'il est difficile de trouver suffisamment de données étiquetées pour entraîner un modèle de manière satisfaisante [21].
Un exemple illustrant l'utilisation d'un apprentissage semi-supervisé est le service d'hébergement d'images "Google Photos". Une fois avoir téléchargé des photos de famille sur ce service, le système arrive à reconnaître qu'une personne A apparaît sur telle ou telle photos et qu'une personne B sur telle autre. Cela est dû à la partie non supervisée de l'algorithme. Une fois que vous avez identifié ces personnes, juste une étiquette par personne, le système sera capable de nommer les personnes figurant sur chaque photo, ce qui est utile pour des recherches ultérieures [21].

2.3.3.4 Apprentissage par renforcement

Cette méthode d'apprentissage consiste à mettre en place un système à travers lequel l'algorithme va apprendre, par expériences successives, à résoudre un problème en adoptant un comportement idéal. Contrairement aux autres méthodes d'apprentissage il ne dispose pas de données mais plutôt il sera confronté à des variables de son environnement dont il utilisera pour concevoir une stratégie de résolution.
Il s'agit d'un programme informatique qui interagit avec un environnement dynamique dans lequel il doit atteindre un certain but, par exemple conduire un véhicule ou affronter un adversaire dans un jeu. Le programme apprenti reçoit du feedback sous forme de « récompenses » et de « punitions » pendant qu'il navigue dans l'espace du problème et qu'il apprend à identifier le comportement le plus efficace dans le contexte considéré [22].
En 2013, c'était déjà un algorithme ML par renforcement (Q-Learning) qui s'était rendu célèbre en apprenant comment gagner dans six jeux vidéo « Atari » sans aucune intervention d'un programmeur [22].
Les principaux algorithmes ML par renforcement sont les suivants : Q-Learning, Deep Q Network (DQN) et SARSA (State-Action-Reward-State-Action) [22].

2.3.4 Évaluation de l'efficacité des Modèles ML

L'efficacité d'un modèle d'apprentissage automatique est mesurée à l'aide des métriques de performance. Il existe différents types de métriques de performance pour évaluer un modèle ML. Il est très important de choisir la métrique appropriée pour contrôler la performance et l'optimiser. Le résultat de l'évaluation permet également de comparer le modèle avec d'autres modèles.

Les mesures d'évaluation, que nous avons utilisées, sont définies dans l'annexe A.

2.3.5 Apports du ML pour l'Analyse des Fichiers Log

L'utilisation de l'apprentissage automatique pour analyser les fichiers journaux nous permet de :

- **Catégoriser les données rapidement :** Les fichiers journaux peuvent être considérés comme des données textuelles, ce qui signifie que les techniques NLP (Natural Language Processing) peuvent être appliquées pour rassembler les mêmes journaux de manière organisée, ce qui permet de rechercher des types spécifiques de journaux [23].
- **Identifier automatiquement les problèmes :** L'un des avantages de l'apprentissage automatique est qu'il détecte automatiquement les problèmes, l'analyse manuelle des fichiers journaux peut être fastidieuse et coûteuse en termes de temps et de ressources. L'apprentissage automatique peut aider à automatiser une grande partie de ce processus, réduisant ainsi les coûts et le temps nécessaires pour analyser les fichiers

journaux [23].

- **Détection précoce des anomalies :** L'un des avantages les plus puissants de l'apprentissage automatique pour l'analyse des fichiers journaux est la détection précoce des anomalies. Dans la plupart des événements catastrophiques, il y a toujours une anomalie initiale qui n'a pas été détectée et corrigée. L'apprentissage automatique peut détecter cette anomalie avant qu'elle ne crée un problème majeur (détecter les premiers indices de l'anomalie) [23].

2.3.6 Processus d'Analyse des Logs Basé sur ML

Le processus d'analyse des logs basé sur ML contient principalement trois phases : la préparation initiale des données, l'apprentissage et la validation.

2.3.6.1 Préparation initiales de données

La bonne préparation de données est l'étape préalable clé dans une analyse basée sur ML. La préparation des données prend environ 60 à 80% du temps consacré au processus d'extraction de données. Dans la détection d'attaques, cette première phase consiste à trouver un échantillon d'attaque types qui doit être représentatif de la sorte de données à classer. Ces données de fichiers logs sont des données texte non structurées par défaut. Afin de les transformer en journaux structurés il faut passer par un prétraitement qui se fait généralement en quatre phases qui sont :

a. **Parsing :** C'est le processus de division des données en blocs d'informations plus faciles à manipuler et stocker. Dans le but de les reconnaître et de les regrouper de manière significative [12].

b. **Nettoyage :** Cette phase consiste à :
   - **Éliminer les doublons et les erreurs de saisie :** Les données redondantes vont donner plus d'importance aux valeurs répétées. Une erreur de saisie pourra à l'inverse occulter une répétition.
   - **Vérifier l'intégrité de domaine :** Un contrôle sur les domaines des valeurs permet de retrouver des valeurs aberrantes.
   - **Traiter les informations manquantes :** C'est le cas où des champs ne contiennent aucune donnée. Parfois, il est intéressant de conserver ces enregistrements car l'absence d'information peut être une information. D'autre part, les valeurs contenues dans les autres champs non vides risquent aussi d'être utiles.

c. **La numérisation et la normalisation :** C'est la numérisation des valeurs symboliques et la normalisation des valeurs numériques. La normalisation consiste à la mise à l'échelle des valeurs numériques dans l'intervalle [0,1], sans que les différences de plages de valeurs ne soient faussées et sans perte d'informations. Les caractéristiques de type booléen ne sont pas concernées par la numérisation (par ce que ces valeurs sont

soit 0 soit 1). C'est une méthode de prétraitement des données qui permet de réduire la complexité des modèles. C'est également un préalable à l'application de certains algorithmes.

d. **Extraction des attributs et Enrichissement :**La dernière phase consiste à sélectionner que les attributs qui sont susceptibles d'être pertinents pour l'analyse (les données en accord avec les objectifs imposés). Cela permet de diminuer la quantité de données à traiter tout en maintenant la qualité du résultat [12].
L'enrichissement se traduit par l'ajout de nouveaux champs ou attributs en conservant le même nombre d'enregistrements.

2.3.6.2 L'apprentissage et la validation

À l'aide de l'ensemble d'apprentissage, l'algorithme apprend quelles sortes d'attaques se retrouvent dans quelles classes. Lors du test des capacités du modèle, il est important de séparer les données en ensemble d'apprentissage et ensemble de test. Nous ajustons d'abord les paramètres du modèle sur l'ensemble d'apprentissage, puis évaluons les capacités prédictives sur l'ensemble de test. Ceci est pour s'assurer que le modèle se généralise bien sur des données non déjà vu (dont la classification est inconnue). Donc, pour maximiser les capacités de généralisation du modèle, la sélection des paramètres peut être pilotée en sélectionnant différentes combinaisons de paramètres, puis en évaluant certaines mesures comme le score F1 du modèle pour prendre la combinaison qui a donné le meilleur résultat. Cependant, si nous utilisons l'ensemble de test pour évaluer le modèle pour plusieurs combinaisons de paramètres, puis sélectionner la meilleure, nous avons en fait sur-ajusté le modèle sur les données de test (sur-apprentissage). C'est-à-dire que nous avons sélectionné des paramètres de modèle qui fonctionnent spécifiquement bien pour l'ensemble de test, et pas nécessairement sur les nouvelles données [25]. Pour cela nous introduisons «l'ensemble de validation». *Le choix de l'algorithme d'apprentissage est important pour la classification et dépend de la tâche à accomplir, le temps d'apprentissage, le nombre de données dans la base d'apprentissage, ainsi que le nombre de classes sont tous des facteurs à prendre en considération. Dans l'annexe B, nous détaillons le fonctionnement des différents algorithmes d'apprentissage supervisé et non supervisé que nous avons testés.*

2.4 Etude Comparative des Techniques d'Analyse des Logs Web Basées sur ML

Les deux tableaux ci-dessous présentent des travaux basés sur l'analyse des fichiers logs web pour la détection des attaques web en utilisant des techniques de machine learning supervisés et non supervisés, ce sont des articles récents qui ont utilisé des techniques de ML et deep learning.

	PRÉTRAITEMENT	APPRENTISSAGE				Précision
		Attributs	Algorithme ML	Dataset utilisé	Attaques détectées	
APPRENTISSAGE SUPERVISÉ						
« Web Application Attacks Detection Using Machine Learning Techniques » [26]	-Décodage d'url -Extraction des features Sc1 : Tokenisation standard par mots Sc2, Sc3 : Tokenisation basée sur des features définies par un expert en sécurité. -Numérisation Sc1 : Calcul des fréquences TF-IDF des tokens du vocabulaire existants dans chaque requête HTTP. Sc2, Sc3 : pour chaque requête, il compte le nombre d'apparitions de chaque feature.	Scénario 1 : - Paramètres de requête - Les en-têtes - Corps de la requête Scénario 2 et 3 : Les features spécifiques aux attaques. (Dans les 3 scénarios, les features sont validés en appliquant l'algorithme de gain d'information)	Sc1 : - RF - KNN - SVM Sc2 : - RF - KNN Sc3 : - Expectation Maximization (EM)	Trois DATASET HTTP pour chaque scénarios : 1- PKDD 2007 [40] 2- CSIC 2010 [41] 3- DRUPAL [26]	-SqlInjection -XSS -XPathInjection -OsComanding LdapInjection . . .	SVM :98% RF : 96% EM : 75%
« Web attack detection using deep learning models. » [27]	- Parsing pour créer un fichier au format CSV - Suppression des valeurs manquantes - Suppression des doublons et suppression des colonnes non nécessaires - Numérisation avec le codage ordinal - Normalisation 'Données mises à l'échelle de 0 à 1"	- Méthode - URL - Host - Cookie - Payload	ANN CNN RNN	DATASET HTTP CSIC 2010	- SQL injection Buffer Overflow - CRLF injection - XSS, etc	ANN : 71% CNN : 86% RNN : 98 %
« Web Server Attack Detection using Machine Learning » [29]	-Simple feature extraction : Les tokens sont extraits de la partie requête de l'URL, si un token est présent dans la requête, il est noté 1, sinon 0. -Text-based-feature : TF-IDF sont calculés pour les Url + L'application de PCA pour la réduction de la dimensionnalité.	URL	Decision Tree SVM KNN	Les journaux du serveur Web sont générés et collectés en créant un réseau privé à l'aide d'un serveur Apache WAMP..	-DOS -SQLi -XSS	DT : 98% SVM : 97% KNN :88%
"Detecting Common Web Attacks Based on Supervised Machine Learning Using Web Logs" [30]	-Extraction Des caractéristiques URI basée sur la Technique n-gram -La vectorisation Des caractéristiques URI basée sur la Combinaison de méthodes (TF-IDF)	URI	-Naïve Bayes -SVM -Decision Tree -Random Forest (RF)	- HTTP Param Dataset (trafic web généré) logs Web collectés à partir des serveurs Web réels	-SQLi -XSS -CMDi -Path traversal	NB : 88% SVM : 98% DT : 99% RF : 99.68%

TABLE 2.1 – Travaux basés sur l'apprentissage supervisé

	Jeu de données utilisé	Attributs utilisés	Méthode de Prétraitement	Techniques utilisées : Métriques de performance
"Anomaly detection of Malicious users behaviors for Web applications based on web logs" [31]	Les journaux Web sont collectés à partir de certains sites Web scolaires sites BUPT.	- URL - HTTP - Code d'état - Longueur de contenu - Référent - Adresse IP - Agent utilisateur	- Extraction des caractéristiques pour l'URI et Référent - Encodage pour le code d'état - Normalisation pour la longueur de contenu - Vectorisation pour l'adresse IP et l'agent utilisateur	K-Means : Précision = 96.67%
"Anomaly Detection in Log Files Using Machine Learning Techniques" [32]	Dataset fourni par Ericsson	EventID, EventTemplate	- Parser	Local Outlier Factor : Précision = 82.3% K-Means+PCA : Précision = 93.5%
"Anomaly Detection in Log Files Using Machine Learning" [33]	Le fichier journal reçu de Mobilaris de l'un de leurs services, Tag Vibration Service (TVS)	Les attributs constants dans le log	Représentation vectorielle des caractéristiques OR IDF	K-Means+vectorisation +PCA : Recall = 0,83 K-Means+PCA+IDF : Recall = 0,91 DBSCAN+vectorisation+PCA : Recall = 0,90 DBSCAN+IDF+PCA : Recall = 0,78

TABLE 2.2 – Travaux basés sur l'apprentissage non supervisé

2.5 Synthèse

Les tableaux 2.1 et 2.2 présentent plusieurs travaux qui utilisent des techniques ML pour analyser les fichiers log dans le but de détecter les attaques Web. Chacun des travaux utilise des attributs différents pour entraîner les modèles, tels que l'URL, les paramètres de l'URL, la longueur de l'URL, le code d'état HTTP, le nombre de paramètres de la requête, etc. Les techniques ML utilisées incluent des algorithmes de classification et de clustering tels que Naive Bayes, SVM, Random Forest, Decision Tree, KNN, K-Mean, et LOF, et des techniques de deep learning tels que ANN, RNN et CNN. Les jeux de données utilisés dans ces travaux contiennent différents types d'attaques tels que SQLi, XSS, CMDi, etc.
Pour le prétraitement, les attributs sous forme de chaîne de caractères tel que l'URL, et le user-agent, étant des attributs importants, des méthodes NLP ont été utilisées pour les numériser. Pour valider le choix des attributs, la méthode de gain d'information est souvent utilisée.
La précision de détection des attaques varie selon la qualité de dataset utilisé, et les attributs choisis, allant de 75% à 99,68%. Les travaux ont réussi à détecter des attaques avec une précision supérieure à 98% dans plusieurs cas. Les techniques de réduction de dimensionnalité, telles que PCA, ont été également utilisées pour améliorer la précision de détection.
Par conséquent, ces travaux ont démontré que les techniques ML sont très utiles pour la détection d'attaques web en analysant les fichiers log, et peuvent renforcer la sécurité des applications web. Cependant, ces travaux ont souligné que la performance des modèles ML dépend fortement de la qualité des données d'entraînement, la bonne sélection des attributs et de la capacité des algorithmes à généraliser les modèles pour détecter de nouvelles attaques.

2.6 Conclusion

Dans ce chapitre nous avons vu l'importance des fichiers log et leurs avantages dans le domaine de la cyber-sécurité notamment pour la découverte d'activités malveillantes. Leur analyse manuelle peut être fastidieuse et prend beaucoup de temps. Dans ce contexte, l'apprentissage automatique offre une solution prometteuse pour extraire des informations utiles à partir des fichiers logs de manière efficace et précise.
De ce fait, nous avons présenté un état de l'art sur l'apprentissage automatique, et comment les logs peuvent en profiter pour fournir un système efficace capable d'identifier les attaques possibles. Nous avons achevé ce chapitre par une étude comparative des techniques ML en cyber-sécurité, d'où nous avons conclu que ces algorithmes étant utilisés dans plusieurs travaux, représentent une solution efficace quant à l'analyse des fichiers logs.
Dans le prochain chapitre nous allons présenter notre contribution dans l'analyse des fichiers logs web et la méthode que nous avons suivie afin de créer un modèle basé sur les capacités d'apprentissage supervisé et non supervisé pour détecter les attaques web.

Chapitre 3

Conception d'un Système d'analyse Avancée des Fichiers logs

3.1 Introduction

Dans le chapitre précédent, nous avons vu que l'analyse manuelle des fichiers logs est presque impossible et très coûteuse. Cela a motivé les chercheurs à trouver des moyens plus efficaces et plus rapides pour résoudre ce problème, en particulier avec l'essor des technologies de l'apprentissage automatique.
D'après l'étude que nous avons faite, nous avons conclu que les algorithmes ML représentent une solution efficace quant à l'analyse des fichiers logs permettant de détecter de nombreuses anomalies et traces d'attaques, et ils peuvent être utilisés pour renforcer la sécurité des applications web.
Dans le présent chapitre, nous allons décrire notre solution d'analyse de fichiers log web en présentant l'architecture globale ainsi que ses principaux modules.

3.2 Objectif de notre travail

Ce présent travail s'inscrit dans le cadre de la protection des applications web. Son objectif consiste à la conception et la mise en œuvre d'une solution d'analyse des fichiers log web pour la détection des attaques et anomalies possibles. Notre solution est basée sur des techniques de ML et tente de :

- Améliorer la capacité de détection par la combinaison des algorithmes ML supervisés et non supervisés afin de détecter les indicateurs d'attaques les plus courantes et les comportements anormaux.
- Détecter les nouvelles formes d'attaques.
- Visualiser et transformer les évènements interceptés en des représentations visuelles claires pour une prise de décision au moment opportun.

3.3 Architecture globale de la solution

Pour atteindre nos objectifs et créer une solution capable de détecter les attaques Web à partir des journaux du serveur, nous avons combiné des techniques d'apprentissage supervisé pour la classification des attaques avec des techniques d'apprentissage non supervisé pour la détection d'anomalies et les nouvelles formes d'attaques.
Pour ce faire, nous avons proposé une architecture composée de trois principaux modules, présentée dans la figure 3.1 :

1. **Module de prétraitement :** Son rôle est de préparer et de transformer les données brutes en une forme appropriée pour l'entraînement des modèles de machine learning. Il comporte plusieurs étapes telles que le parsing, l'extraction des caractéristiques et la normalisation.
2. **Module d'analyse :** C'est le cœur de notre solution, il se compose de deux modules qui fonctionnent en parallèle :
   a. **Module de classification d'attaques :** c'est un classifieur automatique dont le rôle est de détecter les attaques avec leurs types. Il se base sur un ensemble de données étiquetées.
   b. **Module de détection d'anomalies :** qui peut améliorer les résultats du module précédent en étant capable de détecter des anomalies dans les fichiers journaux en apprenant l'état valide du serveur, et ceci en utilisant uniquement des journaux contenant des entrées valides de notre ensemble de données. Il se base sur des techniques non supervisées. Une entrée de journal qui est détectée comme une anomalie par ce module mais non détectée comme une attaque prédéfinie par le module de classification des attaques est susceptible d'être une nouvelle attaque inconnue ou pourrait même être une attaque zéro-day. La décision dans ce cas pourrait être prise dans le module de supervision par l'administrateur.
3. **Module de supervision et Affichage des résultats :** Enfin une interface de supervision est mise à la disposition de l'analyste ou l'opérateur de sécurité afin de consulter les résultats et les alertes provenant des différents modules et également de générer des statistiques pour construire une vue générale sur les attaques et les anomalies détectées à l'aide des graphes. Dans le cas d'une attaque/anomalie, l'administrateur aura alors la possibilité de supprimer les fausses alertes et mettre à jour les modèles si nécessaire.

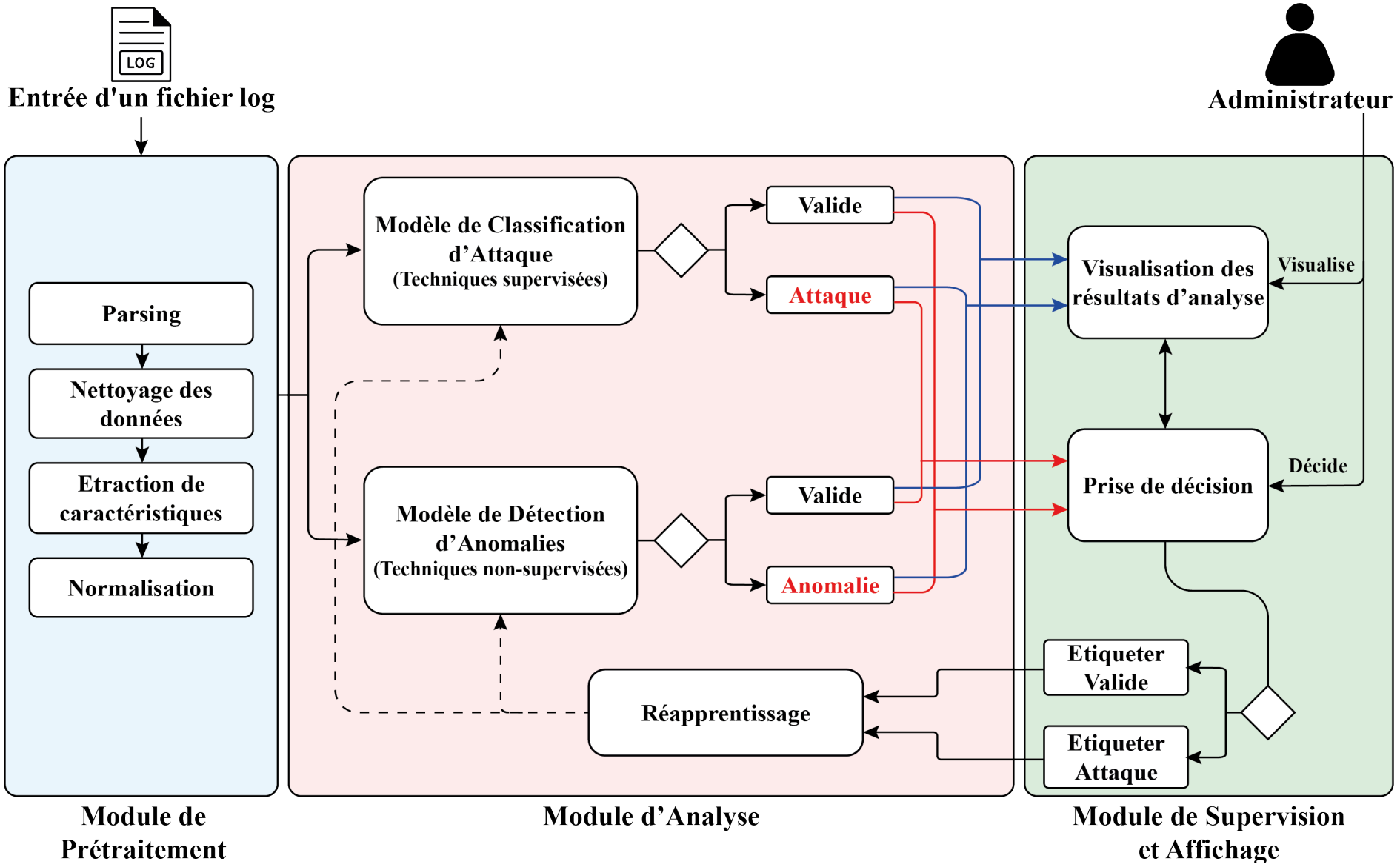

FIGURE 3.1 – Architecture globale de notre solution.

3.4 Génération de Datasets

3.4.1 Challenge et Problèmes d'Acquisition de Données pour la Détection d'Attaques

Dans le chapitre précédent, nous avons présenté certains travaux menés pour l'application des modèles ML au domaine de l'analyse des fichiers log pour la détection des attaques web. Tous ces travaux dépendent de la disponibilité d'un dataset approprié pour l'entraînement, l'évaluation et l'amélioration des modèles de machine learning. L'étude des datasets les plus utilisés révèle que la plupart d'entre eux présentent un certain nombre d'inconvénients :

- **Dataset non étiqueté :** Un problème courant est que les requêtes n'ont pas d'étiquette indiquant la classe à laquelle elles appartiennent. Les étiquettes sont nécessaires pour mesurer la capacité du système à classer les instances.
- **De nombreux ensembles de données ne sont pas accessibles au public :** En raison de la rareté des dataset, de nombreux chercheurs ont choisi de créer leurs propres datasets, dont beaucoup sont à usage privé (pour des raisons de confidentialité), ils ne sont pas utilisables par la communauté scientifique.
- **Ensembles de données non mis à jour :** Étant donné que de nouvelles attaques Web apparaissent constamment, il est important que le dataset soit à jour pour contenir les attaques récentes afin de tester l'efficacité des modèles de détection actuels. Par exemple, le dataset ECML / PKDD 2007 est obsolète et n'inclut pas la plupart des attaques modernes, ce qui le rend insuffisant pour évaluer les modèles sur les attaques actuelles.
- **Le trafic est anonymisé :** Les problèmes de confidentialité sont souvent une cause d'anonymisation des données. Ce processus peut conduire à une perte de réalisme et peut également affecter négativement la qualité des résultats de détection. Le dataset ECML / PKDD est un exemple de trafic anonyme à l'exception de la partie attaque.
- **La distribution d'attaques au sein des datasets est très déséquilibrée.** Certaines attaques Web sont largement sous-représentées.

Afin de surmonter ces inconvénients, en tant que première contribution de ce travail, nous avons généré un nouvel ensemble de données appelé « CERIST_2023 ».

3.4.2 Environnement de génération de notre Dataset CERIST_2023

La première contribution de notre travail consiste à la conception et l'implémentation d'un environnement capable de générer un ensemble de données (dataset) adaptées au contexte de notre solution d'analyse de fichiers log, pour aboutir à des données d'apprentissage de qualité. Cet environnement est composé d'un outil d'attaque, 4 serveurs Web non protégés avec 4 applications vulnérables. Les traces des requêtes malicieuses et légitimes

sont récupérées à partir du trafic web entre la machine de l'attaquant et les applications web. Pour se faire, nous avons :

1. Crée un environnement virtuel vulnérable : déploiement des machines virtuelles.
2. Simulé des attaques vers cet environnement.
3. Collecté les données réelles (trafic réseau) et généré les logs web.
4. Procédé à la classification et l'étiquetage des ensembles de données.

Le schéma de la figure 3.2 illustre le mode fonctionnel de l'environnement de génération de dataset que nous avons mis en place.

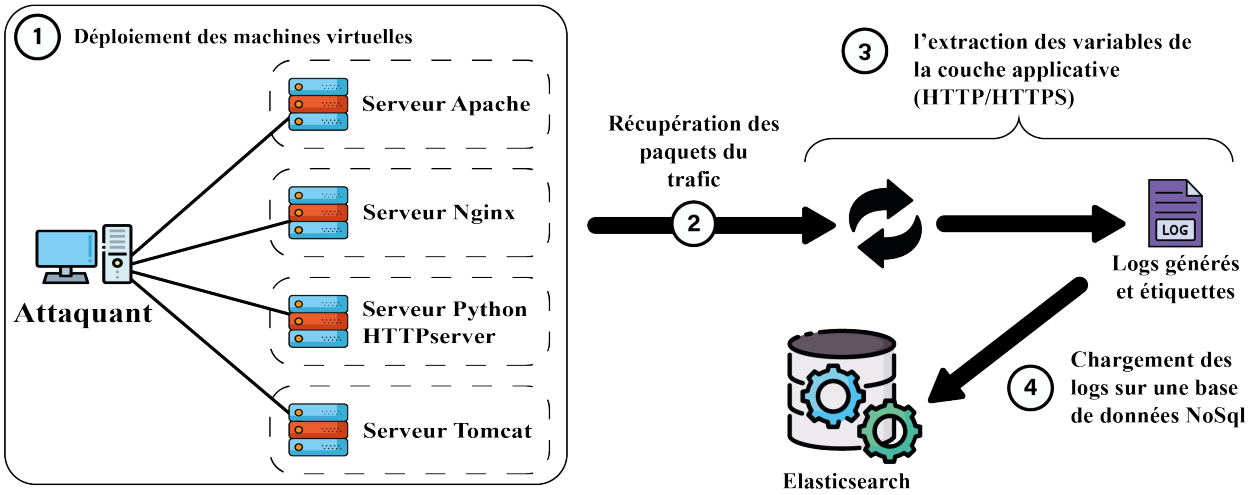

FIGURE 3.2 – Schéma général de génération du dataset.

3.4.2.1 Déploiement des machines virtuelles

La virtualisation permet de créer et de gérer plusieurs machines virtuelles sur un seul serveur physique ou bien machine, ce qui maximise l'utilisation des ressources matérielles et facilite la mise en place et la maintenance des infrastructures informatiques.
Nous avons déployé quatre machines virtuelles sous différents systèmes d'exploitation (Linux et Windows).
Chaque machine virtuelle contient un serveur web différents :

- Apache.
- Nginx.
- Python HTTP Servers.
- Tomcat.

Afin que notre environnement soit le plus proche possible du monde réel, nous avons développé des applications web vulnérables et déployé d'autres déjà prêtes et mises à disposition du public. Ces applications présentent plus de 10 vulnérabilités (TOP 10 OWASP).

3.4.2.2 Simulation des attaques

La simulation des attaques est une technique utilisée pour tester la sécurité informatique. Cela implique d'utiliser des outils spécialisés pour essayer de trouver des failles dans les

systèmes informatiques. Les tests de simulation d'attaques à l'aide de scripts sont souvent réalisés dans le cadre d'un processus d'audit de sécurité régulier. Dans ce travail, nous avons exploité ces scripts pour simuler des attaques sur nos applications vulnérables. Une variété d'attaques récentes et leurs variations ont été simulés dont nous citons :

- SQL Injection et ses variantes (blind SQLI, UNION based SQLI, TIME based SQLI, ERROR based SQLI, SECOND order SQL)
- Cross-Site Scripting (XSS) et ses variantes (XSS reflected, XSS stored, XSS dom based)
- LFI
- RFI
- Commande Injection

Pour cela, nous avons utilisé une machine attaquante (sous Kali linux et ParrotOS) qui exécute des scripts automatiques pour l'exploitation des services web.

3.4.2.3 Récupération des paquets

Cette étape consiste à capturer et analyser les paquets du trafic pour le sauvegarder dans un format PCAP. Pour cela il existe plusieurs outils d'interception de paquets HTTP/HTTPS comme Wireshark, Tshark, Ettercap et TCPdump, etc. Nous avons exploité ces outils afin de pouvoir récupérer tous les paquets qui circulent sur notre environnement virtuel.

3.4.2.4 Extraction des variables de la couche applicative pour former des lignes de logs

L'extraction des variables de la couche applicative, autrement dit, des protocoles HTTP/ HTTPS est une pratique courante en sécurité informatique. Les variables, telles que les entêtes et les corps des requêtes et réponses HTTP/HTTPS, contiennent des informations importantes comme les cookies, les informations d'identification et les données de session. L'extraction de ces variables permet de comprendre le comportement des applications Web. Après avoir terminé notre simulation et récupéré tous les paquets il est temps d'extraire les variable nécessaire à la génération de nos lignes de logs.

3.4.2.5 Étiquetage des données

L'étiquetage des données fait partie de l'étape de prétraitement lors du développement d'un modèle d'apprentissage automatique, qui nécessite l'identification des données brutes et l'ajout d'étiquettes à ces données pour indiquer leur contexte pour les modèles. Cela permet au modèle de faire des prédictions précises [28]. Vu que le dataset a pour but d'être utilisé a l'apprentissage automatique d'un modèle ML supervisé. Pour cela, l'étiquetage doit être effectué après que l'extraction des variables et la formation des lignes de logs est effectué. On aura des lignes de logs étiquetées avec leur type d'attaque.

3.4.2.6 Stockage

La dernière étape de notre architecture consiste à stocker les logs générés. Pour cela, nous avons opté pour les bases de données NoSQL.
En effet, ces bases de données permettent de gérer une grande variété de type de données volumineuses, à haute vitesse. Ce qui répond parfaitement à nos besoins, où nous devons traiter une grande quantité de log , provenant de différents serveurs (plusieurs formats de log) et en temps réel (ou quasi réel).

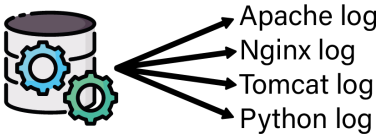

FIGURE 3.3 – Exemple sur le contenu de la base de donnée.

Notant que notre dataset est générique. Nous récupérons les paquets HTTP à partir du réseau pour les stocker sur une base de données, dans le but de générer quatre types de log web "Apache, NginX, Tomcat, et Python", où les données de chaque type de log sont structurées différemment.

3.5 Description des Principaux Modules de la Solution

Dans ce qui suit, nous allons détailler le fonctionnement des principaux modules de notre solution.

3.5.1 Module de Prétraitement

Le prétraitement des données est une étape essentielle dans toute architecture de machine learning. Il vise à améliorer la qualité des données, à les rendre adaptées à l'apprentissage automatique et à garantir des résultats plus fiables et robustes des modèles.
Cette étape consiste à convertir les enregistrements de journaux non structurés dans un format lisible par machine, puis à sélectionner les caractéristiques (Features) potentielles et informatives, ensuite, filtrer et supprimer le bruit et les caractéristiques inutiles, c'est-à-dire de ne conserver que les entrées susceptibles d'être intéressantes pour l'analyse .
Cette étape consiste à convertir les enregistrements de journaux non structurés dans un format lisible par machine, puis à sélectionner les caractéristiques (Features) potentielles et informatives, c'est-à-dire de ne conserver que les entrées susceptibles d'être intéressantes pour l'analyse. Cela inclut également le codage des caractéristiques dans des vecteurs de taille fixe, c'est parce que la plupart des modèles d'apprentissage automatique ne peuvent gérer que des valeurs numériques et les caractéristiques de type non numérique doivent être converties dans un format numérique unifié [42]. Le prétraitement comprend également le processus de normalisation des caractéristiques.

La figure 3.4 présente les étapes du prétraitement, que nous détaillerons dans les sections suivantes.

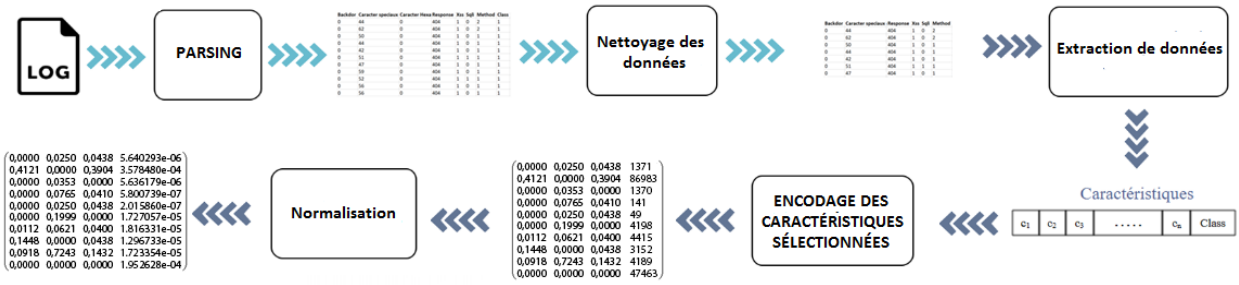

FIGURE 3.4 – Processus de prétraitement.

3.5.1.1 Parsing

Dans cette phase, nous convertissons les logs bruts en un format lisible par machine afin de pouvoir ensuite appliquer les étapes de prétraitement suivantes et créer nos modèles d'apprentissage automatique. La figure ci-dessous présente un exemple de ligne de log extraite de notre dataset :

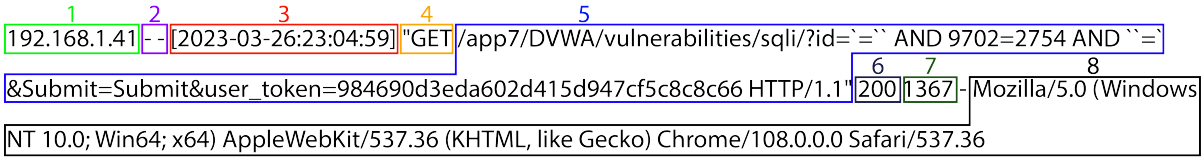

FIGURE 3.5 – Exemple d'une ligne de log extraite du Dataset CERIST_2023.

On peut distinguer dans cette ligne de log :

1. l'adresse IP de l'utilisateur.
2. L'identifiant utilisé pour identifier le client, et le nom d'utilisateur utilisé par le client pour l'authentification (représentés par - - pour indiquer une information manquante).
3. La date et l'heure de la requête, qui correspond au 26 mars 2023 à 04h59.
4. La méthode HTTP utilisée (GET).
5. La page requise ou l'URL avec la version du protocole d'échange.
6. Le code de la réponse ou status code.
7. La taille des données envoyés par le serveur ou response size.
8. Le user agent qui correspond à aux versions du navigateur et système d'exploitation du client.

Le résultat de cette phase sera un tableau structuré où les colonnes représentent les attributs de notre dataset et les lignes représentent les lignes de log.

3.5.1.2 Nettoyage des données

Cette phase consiste à nettoyer l'ensemble de données brutes, en supprimant les valeurs manquantes, éliminant les valeurs en double, et en décodant les urls pour aider à mieux détecter les attaques Web.

- **Suppression des valeurs manquantes :** Il existe de nombreuses raisons pour lesquelles des valeurs manquantes peuvent se produire, notamment des données corrompues, un échec de chargement ou une extraction incomplète. Les valeurs manquantes sont problématiques lors de l'utilisation d'un ensemble de données pour l'apprentissage automatique. Elles rendront la visualisation et l'analyse des données plus difficiles.
- **Suppression des doublons :** Les processus de nettoyage des données nécessitent souvent la détection de données en double. La présence de doublons peut affecter négativement les performances des modèles d'apprentissage automatique, car ils introduisent des biais et peuvent fausser les résultats.
- **Décodage d'URL :** Les données URL sont généralement encodées pour convertir le format des données. Une URL courte et claire peut être obtenue par décodage d'URL, ce qui est pratique par la suite pour l'entraînement du modèle sur des valeurs réelles. Voici un exemple d'URL avant et après le décodage :
  Avant décodage :
  http ://[domaine]/admin/users ?id=%27%20or%201%3D1%20and%20sleep%285%29
  Après décodage :
  http ://[domaine]/admin/users ?id=' or 1=1 and sleep(5)–

3.5.1.3 Extraction des Caractéristiques

L'extraction des caractéristiques est la transformation de données brutes en un nouvel ensemble de caractéristiques qui représentent de manière plus significative les informations contenues dans les données d'origine. Cette transformation vise à mettre en évidence les aspects pertinents des données et à réduire la complexité en extrayant les caractéristiques les plus pertinentes. Cette étape inclut des opérations telles que la sélection des attributs, la création de nouvelles caractéristiques à partir des données brutes et l'encodage des caractéristiques sélectionnées.
Dans ce qui suit, nous présentons le processus d'extraction de caractéristiques que nous avons utilisé :

1. **Sélection des Caractéristiques :**
   La sélection de caractéristiques (ou sélection d'attributs) consiste à identifier les caractéristiques les plus pertinentes parmi les attributs disponibles dans l'ensemble de départ, ce qui permet de réduire la complexité des données, d'améliorer la précision et la généralisation du modèle.
   D'après l'étude que nous avons mené sur des travaux récents, nous avons retenu des

caractéristiques suivantes :

- **URL :** Les URL jouent un rôle important dans la détection d'attaque, car ils peuvent être utilisés pour envoyer des payloads d'attaques.
- **Response Size :** La quantité de données renvoyées par un serveur en réponse à une requête HTTP. La taille de la réponse peut varier en fonction du type de ressource demandée, de la configuration du serveur et des paramètres du client, et même du type d'attaque en cas de requête présentant une attaque.
- **User Agent :** Est une séquence de caractères qui fournit des informations sur l'application, le système d'exploitation, le fournisseur et la version de l'entité du client. Il est inclus dans l'en-tête de la requête HTTP et envoyé par le navigateur web du client. Le serveur web l'utilise pour déterminer la réponse appropriée à la requête. Cet attribut peut être exploité par un attaquant pour envoyer des payload d'attaques.

2. Encodage des caractéristiques sélectionées :

Notre dataset contient uniquement des données textuelles qui doivent être transformées en valeurs numériques pour les utiliser comme entrée dans nos modèles. Ceci constitue une exigence car les modèles d'apprentissage automatique nécessitent des entrées dans des vecteurs dont les éléments constitutifs sont sous format numérique. Les caractéristiques sélectionnées sont divisées en deux catégories :

2.1 Données catégoriques :

C'est le type de données qui a un nombre limité de possibilités comme la colonne label (classe d'attaque) qui n'a que 6 valeurs, pour celà nous avons utilisé l'encodage ordinal qui consiste à attribuer aux différentes valeurs de la colonne des nombres entiers en fonction de leur relation ordinale les unes avec les autres.

2.2 Données non catégoriques :

Ce sont les données qui ne peuvent pas être limitées à certaines valeurs, comme les attributs **url** et **user-agent**, un processus pour convertir les caractéristiques textuelles, en représentation numérique/vectorielle s'avère donc nécessaire. Les journaux des serveurs contiennent beaucoup de données textuelles et d'entrées qui peuvent être fournies par l'utilisateur dans un format de langage naturel. Par conséquent, certaines techniques de traitement du langage naturel peuvent être appliquées à notre solution. Dans notre cas, nous avons utilisé la technique de traitement du langage naturel Term Frequency-Inverse Document Frequency (**TF-IDF**).

Cette méthode fonctionne en intégrant les données du texte sous forme de tokens. Cela nécessite que les lignes des journaux soient divisées en tokens individuels via le processus de tokenization, ensuite représenter les tokens résultants par leur poids tf-idf.

a. La tokenisation :

La tokenisation est probablement la partie la plus complexe du pipeline de

prétraitement. Elle consiste à découper un texte brut en unités plus petites appelées "tokens" qui peuvent être des mots, des caractères ou des sous-mots. C'est une tâche courante dans le traitement du langage naturel (NLP) [43].

- **Tokenisation par mots :** consiste à diviser un morceau de texte en mots, en fonction d'un certain délimiteur (espaces et signes de ponctuation), où chaque mot représente un token. L'inconvénient de cette méthode est la taille énorme du vocabulaire qui poserait des problèmes de mémoire et de performances lors de l'utilisation d'algorithmes ML, ainsi que le problème de traitement des mots hors vocabulaire qui font référence aux nouveaux mots rencontrés lors des tests.
- **Tokenisation par caractères :** divise le texte en un ensemble de caractères, cette méthode produit un très petit vocabulaire, par exemple un texte écrit en anglais aurait au plus un vocabulaire de taille 256 caractères (lettres, chiffres, caractères spéciaux), elle gère un mot hors vocabulaires de manière cohérente, elle décompose ce mot en caractères et représente le mot en fonction de ces caractères. Cette méthode souffre par contre d'une plus grande longueur de séquence et perd de sa signification sémantique car les caractères seuls ne peuvent pas exprimer le sens du texte.
- **Tokenisation par sous-mots :** se situe entre la tokenisation basée sur les mots et les caractères. Elle divise le texte en sous-mots (ou en caractères n-gram), le vocabulaire résultant sera plus petit que les mots mais plus grand qu'avec des caractères.
- **N-grams :** C'est une séquence de n éléments, ces éléments peuvent être des lettres ou des mots. Le n-gram décompose d'abord le texte en mots/-caractères, puis il construit N-gram de chaque mot/caractère de longueur spécifiée N.

Ces méthodes standard, présentent un certain nombre d'inconvénients pour être appliqués aux logs :

(a) Un simple texte peut être divisé simplement par des espaces blancs et d'autres caractères spéciaux. Cependant, pour les logs, il existe généralement beaucoup plus de caractères qui peuvent représenter des délimiteurs entre les tokens potentiels, tels que les crochets, les parenthèses, le signe égal, etc. Ce qui conduirait à une augmentation incontrôlable des tailles de vocabulaire, c'est-à-dire une cardinalité élevée de tokens.

(b) Les tokens résultants n'ont pas de sémantique.

(c) Le problème de généralisation à d'autres contextes d'application web : un classifieur conçu pour un jeu de données avec la tokenisation standard ne peut pas être utilisé avec un autre.

Afin que la sémantique soit préservée et que les tokens soient généralisables, un

ensemble de tokens qui caractérisent mieux les différentes attaques d'applications Web ont été définies. A cet effet, nous avons mené une étude approfondie sur la syntaxe des principales attaques web, et par la suite nous avons créé une liste qui contient plus de 250 tokens (mots et caractères spéciaux) contenus dans la majorité des payloads des attaques web. Enfin, nous avons traduit l'ensemble en une expression régulière, qui a été validée par un expert en sécurité informatique.

Cette expression régulière (REGEX) englobe tous les tokens (mots) définis par l'expert et contenus dans notre dataset web. La structure de notre REGEX est constituée des mots prédéfinis et séparer par un opérateur logique (OR) pour permettre la détection de tous les mots qui peuvent exister dans une ligne de log web, ces tokens reflètent des signatures des différentes attaques, par exemple, une attaque SQL injection est détectée avec ces tokens :

UNION, INSERT, UPDATE, SELECT, WHERE,

ou même avec des combinaisons de tokens tels que :

'OR 1=1 AND SELECT password FROM users limit 1–'

Nos tokens ont été minutieusement choisis pour couvrir tous les vecteurs d'attaque possible et ensuite appliquer une comparaison avec le contenu des deux attributs sélectionnés à savoir l'URL et l'user-agent.

b. Représentation des tokens (Vectorisation) :

Après avoir effectué la tokenisation, TF-IDF va être appliqué à notre solution. Nous calculons la fréquence TF-IDF de chaque token présent dans chaque ligne de log. La fréquence TF-IDF est une mesure statistique utilisée pour évaluer l'importance relative d'un mot pour un document dans une collection de documents. Il est souvent utilisé dans la recherche d'informations et l'exploration de texte. L'importance d'un mot augmente proportionnellement au nombre de fois où il apparaît dans le document mais elle est compensée par la fréquence du mot dans la collection (la collection de ligne de log dans notre cas), donnant plus de poids aux mots qui sont fréquents dans le document mais moins fréquents sur le reste de la collection [45]. Les résultats seront présentés dans le prochain chapitre.

Le TF-IDF pour un mot dans un document est calculé en multipliant deux mesures différentes [45] :

$$tf(t,d) = \frac{f(t,d)}{max\{f(w,d) : w \in d\}}$$

$$idf(t,D) = log\frac{N}{|d \in D : t \in d|}$$

$$TF - IDF(t,d,D) = tf(t,d) * idf(t,D)$$

- **tf (t,d) :** la fréquence de terme t (un mot ou un caractère) dans l'URL d.
- **f (t,d) :** le nombre d'occurrences de t dans l'URL d.
- **max {f(w,d) :w E d} :** est le nombre maximum de termes dans l'URL d.
- **idf (t,D) :** la fréquence inverse de terme t dans l'ensemble de tous les URL.
- **N :** est le nombre total d'URL.
- **D :** l'ensemble de tous les URL.

3.5.1.4 Normalisation

La normalisation permet de réduire la complexité des modèles, lorsque les caractéristiques sont à des échelles radicalement différentes. Étant donné que l'échelle de chaque caractéristique n'est pas nécessairement la même, lors de l'entraînement du modèle, les caractéristiques à plus grande échelle joueront un rôle décisif et les caractéristiques à plus petite échelle joueront un petit rôle. Dans l'étape de normalisation, nous contrôlons les caractéristiques de différentes échelles sous la même échelle standard, éliminons l'influence des différences d'échelle, et évitons que certaines caractéristiques aient un poids disproportionné dans le modèle. Par conséquent, l'étape de normalisation des caractéristiques est très nécessaire [46].
Dans notre cas, tous les attributs sont dans l'intervalle [0,1], sauf l'attribut response_size qui a été normalisé, en utilisant la méthode min-max scaler.

C'est la méthode la plus simple, qui met à l'échelle les données de manière à ce qu'elles soient bornées entre [0,1], tout en conservant les distances entre les valeurs. Elle consiste à soustraire aux données leur valeur minimale min(x) et à diviser le résultat par l'écart maximum rencontré (max(x)- min(x)) comme suit :

$$x_{scaled} = \frac{x - min(x)}{max(x) - min(x)}$$

A la fin de l'étape prétraitement, le résultat sera une matrice où les lignes représentent les lignes de log et les colonnes représentent les caractéristiques sélectionnées dans un format numérique comme illustré dans la figure 3.6.

	!	"	#	$	%	&	'	or 1=1	(	)	...	whoami	with	yum	zip	{	|	}	~	label	response_size
0	0.0	0.000000	0.0	0.000000	0.0	0.084697	0.337246	0.0	0.085350	0.128021	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	2.0	5.640293e-06
1	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.0	0.091858	0.091855	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	3.0	3.578480e-04
2	0.0	0.000000	0.0	0.114872	0.0	0.119878	0.636442	0.0	0.120803	0.120799	...	0.0	0.0	0.0	0.0	0.0	0.467771	0.0	0.0	2.0	5.636179e-06
3	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.0	0.089373	0.089371	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	3.0	5.800739e-07
4	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.0	0.093074	0.093071	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	3.0	2.015860e-07
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

FIGURE 3.6 – Jeux de données normalisé pour un classifieur supervisé.

Selon l'ordre des caractéristiques dans le vecteur des caractéristiques, cette ligne indique que dans la première ligne de log, il existe les tokens "%" , "'", "(", ")","where",...ect, avec leurs fréquences TF-IDF. L'avant dernière colonne représente la classe d'attaque pour guider le classifieur dans sa phase d'apprentissage.

3.5.2 Module d'analyse

Ce module permet d'analyser les logs prétraités, pour détecter d'éventuelles attaques et anomalies. Notre analyse consiste à combiner deux types de modèles d'apprentissage automatique qui fonctionnent en parallèle, le premier modèle supervisé permet de traiter les logs stockés pour détecter des attaques Web connues, le deuxième modèle non supervisé se base uniquement sur les comportements des utilisateurs pour détecter une variation par rapport aux comportements normaux qui représentent une anomalie.
Selon l'article [47], les algorithmes supervisés réalisent d'excellents résultats pour des attaques connues, ils sont meilleurs que les algorithmes non supervisés. Inversement, pour des attaques inconnues, les algorithmes supervisés voient une réduction drastique de leur capacité contrairement aux algorithmes non supervisés. Pour de meilleurs résultats, l'idéal est de combiner des algorithmes supervisés et non supervisés, pour avoir les avantages des deux techniques précédentes.
Ce module est donc scindé en deux modules : le module de classification des attaques, et le module de détection d'anomalies.

3.5.2.1 Module de classification des attaques

Ce module comprend l'identification des attaques Web connues, donc nous nous sommes orientés vers le choix des algorithmes d'apprentissage supervisé, ces algorithmes doivent être entraînés sur des données étiquetées avant de pouvoir être utilisés pour la prédiction.
Donc notre dataset doit être divisée en données d'entraînement (80%) et données de test (20%), comme illustré dans la figure 3.8.

- Les données d'entraînement sont utilisées pour former nos classificateurs afin d'obtenir le modèle de détection.
- Les données de test sont appliquées à travers le modèle de détection afin de l'évaluer et confirmer si notre modèle détecte correctement les attaques, avec une bonne précision. Les erreurs commises par ce même modèle lors de la phase de test servent pour évaluer les performances du classifieur, On parle de performances de généralisation quand il s'agit de prédire correctement la classe des nouvelles données.

Le choix du modèle de classification peut se justifier par les bonnes performances de ce modèle. Puisqu'il n'y a pas d'algorithme de classification standard pour les logs web, nous avons choisis de comparer plusieurs algorithmes d'apprentissage supervisés comme : « KNN »,« Naive Bayes », « SVM », « Decision Tree », « Random Forest », etc.
La comparaison se fait selon plusieurs critères d'évaluation (accuracy, précision, rappel et F1 score).

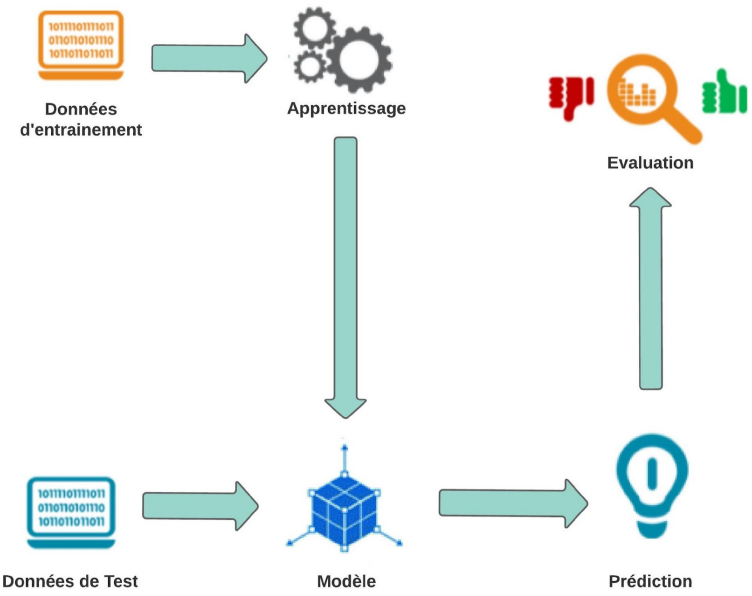

FIGURE 3.7 – Fonctionnement de l'algorithme d'apprentissage supervisé.

La construction de notre modèle de classification (illustré dans la figure 3.7) suit les étapes suivantes :

1. Présenter les fichiers logs étiquetés « ensemble d'entraînement » aux différents algorithmes pour entraîner ses classificateurs, ce qui permet de construire plusieurs modèles prédictifs.
2. Appliquer les modèles prédictifs obtenus à des fichiers log de test (sans étiquettes) pour faire de la prédiction.
3. Évaluer les modèles pour déterminer s'ils contribueront à prédire correctement les attaques : comme les instances futures ont des valeurs cibles inconnues, nous devons vérifier les métriques d'évaluation (cf. annexe A) du modèle d'apprentissage sur des données dont on connaît déjà la réponse cible (les données de test), puis utiliser cette évaluation comme indicateur de précision prédictive des données futures.
4. Afin de choisir l'algorithme idéal pour notre classification, nous avons comparé les métriques d'évaluation sur le même dataset.

A la fin, l'algorithme d'apprentissage qui donne la meilleure performance globale sera sélectionné pour la construction du modèle pour détecter les attaques web.

Notre modèle permet de reconnaître 4 types d'attaques Web parmi les plus courantes actuellement, dont nous citons :

- SQL Injection et ses variantes (blind SQLI, UNION based SQLI, TIME based SQLI, ERROR based SQLI, SECOND order SQL)
- Cross-Site Scripting (XSS) et ses variantes (XSS reflected, XSS stored, XSS dom based)
- LFI
- Commande Injection

3.5.2.2 Module de détection d'anomalies

Les anomalies dans les données du journal sont considérées comme des modèles ou des caractéristiques qui ne suivent pas le comportement moyen ou normal lors d'un fonctionnement parfait, de telles anomalies peuvent être déclenchées par des acteurs malveillants, des bugs au niveau du système ou un fonctionnement incorrect de l'utilisateur sont souvent symptômes d'une défaillance ou d'une violation imminente du système.
Le but de ce module est de détecter ces anomalies pour renforcer les lacunes des résultats du module de classification des attaques. Nous avons considéré parmis les lignes de logs suspectées par notre modèle comme anomalies, dont nous citons :

- Tentatives de demande de ressources cachées (ou inexistantes). Ces demandes incluent des fichiers obsolètes, des fichiers de configuration, des fichiers par défaut, etc.
- Requêtes anormales non intentionnelles : ces requêtes ne sont pas malveillantes, mais elles ne suivent pas le comportement normal de l'application web et n'ont pas la même structure que les valeurs de paramètres normales (par exemple, un numéro de téléphone composé de lettres).
- Présence d'un scanner de vulnérabilités qui est un programme conçu pour identifier des vulnérabilités dans une application web, ils peuvent être utilisés dans des objectifs illicites par les pirates informatiques pour trouver les failles dans les systèmes des entreprises pour les exploiter à leur avantage.
- Toutes les attaques connues (injection SQL, XSS, RFI, etc.) ou inconnues (attaques zéro day) sont considérés comme anomalies par rapport le comportement normal

Nous avons procédé de la même manière que dans le module de classification cité ci-dessus, sauf que c'est un modèle non supervisé, nous avons utilisé les mêmes métriques pour valider les résultats.

Dans l'annexe B, nous décrivons le fonctionnement des différents algorithmes non supervisés que nous avons testés.

3.5.3 Module de supervision

Ce module est la couche de visualisation, qui offre une visibilité étendue et continue, afin de mieux comprendre, communiquer et gérer les cyberattaques. Pour cela, nous l'avons doté de plusieurs fonctionnalités, dont nous citons principalement :

3.5.3.1 Visualisation des résultats d'analyse

Ce module est principalement basé sur des interfaces graphiques , il permet à l'administrateur de :

- Gérer les entrées de journal, visualiser et filtrer les fichiers journaux en fonction de différents critères.
- Visualiser nos données et les différentes statistiques sur les attaques et les anomalies détectées, en utilisant des graphiques.
- Exporter le résultat des analyses sous format d'un fichier CSV pour une utilisation ultérieure.
- Suivre et analyser toutes les anomalies ou les activités suspectes qui se présentent.
- Effectuer des recherches avancées selon certains critères, dont nous citons :
  - Adresse IP
  - Type d'attaque
  - Mot clé
  - Combinaison entre attaque et adresse IP

3.5.3.2 Module de Prise de Décision

Ce module aide l'administrateur à interagir avec les différents modules de notre système, principalement le module de classification des attaques et le module de détection d'anomalies, et de consulter les résultats de sortie de chacun des deux modules.
Si les deux modules de classification et de détection d'anomalies donnent les mêmes résultats i.e attaque/anomalie ou valide/valide, on affiche le résultat directement à l'administrateur. Si par contre ces deux modules donnent des résultats différents i.e attaque/valide ou valide/anomalie, cela nécessite l'intervention de l'administrateur comme suit :

a. Lorsque le module de classification d'attaque détecte une entrée de journal comme valide mais le module de détection d'anomalie la détecte comme anomalie, ce qui peut indiquer l'une des conclusions suivantes :

- L'entrée est une attaque connue par le module de classification mais il l'a classé comme valide (i.e. faux négatif) ou bien c'est une nouvelle attaque inconnue que le module de classification des attaques n'est pas formé pour la détecter. Ainsi, selon le cas, l'administrateur peut étiqueter l'entrée de journal comme une attaque prédéfinie ou nouvelle attaque selon son type, et l'envoyer vers un ensemble de données de réapprentissage qui peut être utilisé pour mettre à jour le modèle de classification des attaques.
- L'entrée du journal est valide mais le module de détection d'anomalie l'a classé comme anomalie. C'est le cas d'un faux positif ou un nouveau comportement normal , qui peut être envoyé à un nouvel ensemble de données en tant qu'entrée de journal valide pour mettre à jour le module de détection d'anomalies, afin qu'elle ne soit pas détecté comme anomalie la prochaine fois, ce qui améliorera ce module au fil du temps et réduira le nombre des faux positifs.
- L'entrée du journal est une anomalie mais pas une attaque, dans ce cas, nous l'ignorons.

b. Lorsque le module de classification d'attaque détecte une entrée de journal comme attaque mais le module de détection d'anomalie la détecte comme valide, cela peut indiquer l'un des cas suivants :

- L'entrée du log est une attaque connue que le module de classification des attaques a pu détecter, mais le module de détection d'anomalie l'a mal classée (faux négatif par rapport au modèle de détection d'anomalies).
- L'entrée du log est valide mais le module de classification l'a classé comme attaque. Dans ce cas, l'administrateur peut étiqueter cette entrée comme valide, et l'envoyer vers un nouvel ensemble de données qui peut être utilisé pour mettre à jour le module de classification des attaques.

3.6 Conclusion

Dans ce chapitre nous avons présenté l'architecture générale que nous avons proposée ainsi que ses différents modules. L'architecture proposée comporte toutes les étapes nécessaires à l'analyse d'un fichier log, à savoir : la conception du dataset, le prétraitement, l'analyse qui consiste à faire la détection des attaques/anomalies et enfin la présentation des résultats aux administrateurs de sécurité.
Nous avons également présenté et détaillé l'environnement de génération de datasets, qui constitue notre première contribution pour résoudre les problèmes des datasets existants.
Dans le chapitre suivant nous allons discuter les détails de l'implémentation de notre système d'analyse avancée des fichiers logs ainsi que les résultats des tests effectués.

Chapitre 4

Implémentation et Résultats

4.1 Introduction

Dans le présent chapitre, nous présentons notre environnement de travail en décrivant les langages de programmation utilisés ainsi que les différentes bibliothèques et tous les outils nécessaires tout en donnant la motivation de leurs utilisations. Aussi, nous verrons les principales étapes de mise en œuvre de notre solution, allant de la génération de notre dataset à l'implémentation des différents modules de notre architecture et la présentation des différents tests et résultats.

4.2 Environnement et outils de travail

4.2.1 Environnement Matériel

- **Serveurs :** quatre machines virtuelles (VM) ont été créées sur le serveur du CERIST, avec des systèmes d'exploitations différents (Ubuntu Server, windows) dotés de plusieurs applications web vulnérables. Voici les caractéristiques des machines virtuelles :
  - Deux machines ubuntu server dotées d'une RAM de 4 Go, une capacité de stockage de 40 Go et un CPU 4 cœurs.
  - Une machine ubuntu server dotée d'une RAM de 16 Go, une capacité de stockage de 100 Go et un CPU 20 cœurs.
  - Une machine Windows dotée d'une RAM de 4 Go, une capacité de stockage de 40 Go et un CPU 4 cœurs.
- **PC portable :** Utilisé comme machine attaquante afin d'exploiter les vulnérabilités des applications web déployées sur les 4 VMs. Pour cela nous avons créé deux VMs (Parot, Kali Linux) sur nos machines pour simuler les attaques web.
  - Machine 1 : équipée d'un processeur Intel i7 11eme, 16 Go de RAM et d'une carte graphique GeForce RTX 3070.

— Machine 2 : équipée d'un processeur Ryzen 7, 16 Go de RAM et d'une carte graphique GeForce RTX 3060.

- **Google colab :** collaboratory ou colab (en abrégé) développé par Google Research, un outil en ligne qui permet l'écriture et l'exécution du code python sans avoir besoin de l'installation sur un ordinateur local. C'est un environnement particulièrement bien adapté pour l'apprentissage automatique, l'analyse et la visualisation de données. Plus techniquement, colab est un service hébergé de notebooks Jupyter qui ne nécessite aucune configuration et permet d'accéder sans frais à des ressources informatiques, notamment aux GPU. Il utilise des CPU performants pour exécuter des opérations de traitement de données et de modélisation, et offre une allocation de RAM de 12 gigaoctets (Go) [49].

4.2.2 Langages de programmation et logiciels

- **PHP :** Un langage de programmation côté serveur utilisé pour créer des applications web dynamiques et interactives en combinaison avec des technologies web telles que HTML, CSS et JavaScript [50].
- **Python :** considéré comme un langage idéal pour les projets basés sur ML, car il dispose d'un grand nombre de frameworks et de bibliothèques pour aider le développeur à construire ses modèles de manière simple et rapide. Il est conçu pour être simple à lire et à écrire. Python est très populaire et largement utilisé par les professionnels de la cybersécurité et des développeurs en raison de sa facilité d'utilisation et de son efficacité [51]. Dans notre cas, il est utilisé dans la création des sites vulnérables, la transformation des fichiers PCAP en fichier JSON, l'extraction des données nécessaires, et dans la création des modèles ML.
- **TCPdump :** Un utilitaire de ligne de commande qui permet de collecter, filtrer et analyser le trafic réseau qui passe à travers le système. Il est souvent utilisé pour aider à résoudre les problèmes de réseau, ainsi qu'un outil de sécurité. Il est disponible sur plusieurs systèmes d'exploitation, tels que Linux, macOS et Windows [52].
- **Wfuzz :** Un outil destiné à l'attaque par force brute des applications Web. Il permet de découvrir diverses ressources, telles que des répertoires non liés, des servlets, des scripts, etc. De plus, il peut être utilisé pour effectuer des attaques par force brute sur les paramètres GET et POST afin de vérifier différents types d'injections (SQL, XSS, LDAP, etc.), d'essayer différentes combinaisons d'utilisateurs et de mots de passe dans les formulaires, et bien d'autres fonctionnalités [53].
- **SQLMap :** Un outil permettant d'exploiter automatiquement les vulnérabilités d'injection SQL. SQLMap peut être utilisé pour évaluer les sites Web et les bases de données pour les vulnérabilités, puis exploiter ces failles pour obtenir le contrôle de la base de données [54].

4.2.3 Bibliothèques

Les bibliothèques que nous avons importées dans notre projet sont définies dans ce qui suit :

- **Numpy :** est une bibliothèque python qui fournit un objet tableau pour faciliter l'opération avec les listes, il peut être 50 fois plus rapide que la liste python standard. Puisque Numpy rend l'opération sur les tableaux beaucoup plus facile, ceci aidera beaucoup avec les grandes structures de données. L'objet tableau dans NumPy est appelé ndarray, il fournit beaucoup de fonctions qui rendent le travail très facile [55].
- **Pandas :** Pandas est une bibliothèque python à code source ouvert, utilisée pour analyser les données, dotée de nombreuses fonctions d'analyse, de nettoyage et de manipulation des données. Les outils de Pandas sont très utiles pour l'analyse et les structures de données. Le nom "Pandas" fait référence à la fois à "Panel Data" et à "Python Data Analysis" [56].
- **Scikit-Learn :** Scikit-Learn est également appelé "sklearn", il s'agit d'une bibliothèque python, qui propose de nombreux algorithmes de classification et de régression pour l'apprentissage supervisé, tels que la forêt aléatoire et les arbres de décision, ainsi que des regroupements pour l'apprentissage non supervisé, tels que k-means [57].
- **joblib :** est une bibliothèque python qui permet de sérialiser les objets python, c'est-à-dire les sauvegarder dans un format spécial pour pouvoir les recharger après en les désérialisant. Ceci est très utile pour sauvegarder les modèles machine learning entraînés et les utiliser plus tard dans la pratique [58].
- **PCAPkit :** PCAPkit est une bibliothèque Python open-source qui fournit des utilitaires pour manipuler et analyser les fichiers PCAP (packet capture) [59].
- **Parse :** Il s'agit d'une bibliothèque légère et flexible, elle fournit diverses fonctionnalités permettant d'analyser et de manipuler des chaînes de caractères selon des modèles spécifiques [61].

4.3 Déploiement de l'environnement de génération des DATASETS

Dans cette partie, nous allons détailler les principales étapes effectuées pour mettre en œuvre l'architecture de génération du dataset déjà présentée dans le chapitre précédent.

4.3.1 Déploiement des machines virtuelles

Nous avons suivi les étapes ci-dessous :

1. Installer les machines virtuelles avec les différents systèmes d'exploitation : ubuntu serveur, kali linux, parrot et windows.

2. Installer les services nécessaires pour les 4 VMs (Apache, Nginx, Tomcat, Python) avec le langage PHP et le système de gestion de base de données Mysql.
3. Déployer les applications que nous avons développées et les applications vulnérables mises au public, dont nous citons l'application Damn Vulnerable Web Application (DVWA), qui est une application Web développée avec PHP et MySQL. Elle est construite à la base vulnérable afin d'aider les professionnels de la sécurité à tester leurs compétences et leurs outils, aider les développeurs Web à mieux comprendre les processus de sécurisation des applications Web. Son objectif est de pratiquer certaines vulnérabilités Web les plus courantes, avec différents niveaux de difficulté, avec une interface simple[60].
   La commande suivante nous a permi de cloner DVWA du github :
   user@serveur :-$ git clone https ://github.com/digininja/DVWA.git
   Une fois téléchargée, nous avons suivi les étapes indiquées par le créateur de l'application web, pour son installation et sa mise en marche.

4.3.2 Récupération des paquets

Dans cette étape nous avons installé TCPDUMP, afin de capturer les paquets de données échangés entre la machine attaquante et les quatres machines virtuelles hébergeant les applications web vulnérables, pour enfin les stocker dans un fichier PCAP.
- La commande suivante permet d'installer TCPDUMP sur chaque machine qui héberge nos applications web :
user@serveur :-$ apt install tcpdump

- Afin de commencer le "Listenning", chaque instance doit lancer l'outil en spécifiant le nom de l'interface par laquelle passe le trafic réseaux pour ensuite le stocker dans un fichier pcap :
user@serveur :-$ tcpdump -i [interface] -A -s 0 'tcp port 80 and (((ip[2 :2]-((ip[0]&0xf)«2)) - ((tcp[12]&0xf0)»2)) != 0)' -w [nom du fichier].pcap

La figure 4.1 donne un aperçu d'un fichier .PCAP capturé avec TCPDUMP :

FIGURE 4.1 – Extrait d'un fichier .PCAP

4.3.3 Simulation des attaques

Pour la simulation des attaques, nous nous somme contenter des scripts open source mis en œuvre par des spécialistes en cybersécurité pour exploiter les vulnérabilités des applications web que nous avons déployées. Afin d'exécuter notre simulateur nous avons utilisé deux outils : WFUZZ et SQLmap, qui sont populaires dans le domaine des tests de sécurité des applications Web.
Pour cela nous avons exécuté les étapes suivantes :

1. Installer les deux outils sur la machine attaquante, à l'aide de ces deux lignes de commandes :
   user@attaquer :-$ apt install wfuzz
   user@attaquer :-$ apt install sqlmap
2. Commencer la simulation juste après avoir lancé le capteur de paquets (TCPDUMP) sur les serveurs web. Cette étape se base principalement sur les deux commandes suivantes :
   - Simulation d'attaque avec la méthode (GET) à l'aide de l'outil WFUZZ :
     user@attaquer :-$ wfuzz -z file,[payload].txt -u http ://exemple.com/path/ ?input=FUZZ
   - Simulation à l'aide de l'outil SQLmap sur une base de donnée de notre application vulnérable :
     user@attaquer :-$ sqlmap -u http ://exemple.com/path/ ?id=1&Submit=Submit –level 4 –risk 3 –batch

4.3.4 Extraction des variables de la couche applicative pour former les lignes de logs

L'extraction des variables est une étape essentielle. Après avoir terminé notre simulation, il est temps de commencer à former nos lignes de logs comme suit :

1. Convertir le fichier .PCAP généré par TCPDUMP en un fichier structuré sous format JSON qui contient toutes les informations des paquets (Requête/Réponse) afin de pouvoir par la suite parcourir les champs de la requête facilement. Cette transformation a été réalisée à l'aide de la bibliothèque PCAPkit, qui est pratique pour extraire des informations à partir de captures de paquets réseau. La figure 4.2 montre un extrait d'un fichier JSON créé.

```
"http": {
    "receipt": {
        "type": "request",
        "method": "GET",
        "uri": "/app7/DVWA/vulnerabilities/sqli/?id=&Submit=Submit&user_token=984690d3eda602d415d947cf5c8c8c66",
        "version": "1.1"
    },
    "header": {
        "Cache-Control": "no-cache",
        "User-Agent": "sqlmap/1.6.4#stable (https://sqlmap.org)",
        "Referer": "http://192.168.1.38:80/app7/DVWA/vulnerabilities/sqli/?id=&Submit=Submit&user_token=984690d3eda602d415d947cf5c8c8c66",
        "Host": "192.168.1.38",
        "Accept": "*/*",
        "Accept-Encoding": "gzip,deflate",
        "Connection": "close"
    },
    "body": null
}
```


FIGURE 4.2 – Extrait d'un fichier JSON.

2. Pour former nos lignes de log, nous avons développé un script qui va parcourir les attributs de notre fichier JSON afin de récupérer leur valeur, à la fin d'exécution de ce script, nous aurons des lignes de logs enregistrées dans un fichier texte, comme le montre la figure 4.3 :


```
192.168.1.41 - - [2023-03-26:23:01:12] "GET /app7/DVWA/vulnerabilities/sqli/?id=&Submit=Submit&user_token=
984690d3eda602d415d947cf5c8c8c66 HTTP/1.1" 200 1370 - Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15
(KHTML, like Gecko) Version/16.1 Safari/605.1.15
192.168.1.41 - - [2023-03-26:23:01:16] "GET /app7/DVWA/vulnerabilities/sqli/?id=&Submit=Submit&user_token=
984690d3eda602d415d947cf5c8c8c66 HTTP/1.1" 200 1372 - Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/109.0.0.0 Safari/537.36
192.168.1.41 - - [2023-03-26:23:04:52] "GET /app7/DVWA/vulnerabilities/sqli/?id=&Submit=Submit&user_token=
984690d3eda602d415d947cf5c8c8c66 HTTP/1.1" 200 1370 - sqlmap/1.6.4#stable (https://sqlmap.org)
192.168.1.41 - - [2023-03-26:23:04:54] "POST /app7/DVWA/vulnerabilities/sqli/?id=&Submit=Submit&user_token=
984690d3eda602d415d947cf5c8c8c66 HTTP/1.1" 200 1370 - Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/108.0.0.0 Safari/537.36
192.168.1.41 - - [2023-03-26:23:04:54] "GET /app7/DVWA/vulnerabilities/sqli/?id=&Submit=Submit&user_token=
984690d3eda602d415d947cf5c8c8c66 HTTP/1.1" 200 1369 - Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/109.0.0.0 Safari/537.36
192.168.1.41 - - [2023-03-26:23:04:57] "GET /app7/DVWA/vulnerabilities/sqli/?id=4211&Submit=Submit&user_token=
984690d3eda602d415d947cf5c8c8c66 HTTP/1.1" 200 1368 - Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/109.0.0.0 Safari/537.36
192.168.1.41 - - [2023-03-26:23:04:57] "GET /app7/DVWA/vulnerabilities/sqli/?id=',)..))(("&Submit=Submit&user_token=
984690d3eda602d415d947cf5c8c8c66 HTTP/1.1" 200 1367 - Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/109.0.0.0 Safari/537.36
192.168.1.41 - - [2023-03-26:23:04:57] "GET /app7/DVWA/vulnerabilities/sqli/?id='VSJBNt<'">OlTXYh&Submit=Submit&user_token=
984690d3eda602d415d947cf5c8c8c66 HTTP/1.1" 200 1367 - sqlmap/1.6.4#stable (https://sqlmap.org)
192.168.1.41 - - [2023-03-26:23:04:57] "GET /app7/DVWA/vulnerabilities/sqli/?id=) AND 4211=1323--
bvsn&Submit=Submit&user_token=984690d3eda602d415d947cf5c8c8c66 HTTP/1.1" 200 1371 - Mozilla/5.0 (Macintosh; Intel Mac OS X
10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
```


FIGURE 4.3 – Lignes de log générées.

Tous les scripts python que nous avons développé pour former nos lignes de log sont présentés dans l'annexe C.

4.3.5 Étiquetage des données

Dans cette étape, nous devons étiqueter chaque ligne de log avec le type d'attaque qui lui convient. Pour ce faire, nous avons utilisé la bibliothèque "parse" pour parser la ligne de log pour ensuite lui étiqueter son attaque. Dans ce qui suit le code développé :

```
noneLabledLogs = open('sql_injection.txt','r')
labledLogs = open('sql_injection_etiq.txt','w')
template = '{ip} {someVar} {user} {date} "{method} {url}" {status_code} 4 {reponse_size} "{refrer}" "{user_agent}"'
for line in noneLabledLogs.readlines():
    line = line.strip('\n')
    out = parse(template,line)
    if (out == None):
        line = line + " \"-\" \"-\""
    labledLine = line + " (sql injection)"
    labledLogs.write(labledLine+"\n")
    print(labledLine+"\n")
noneLabledLogs.close()
labledLogs.close()
```

FIGURE 4.4 – Script pour l'étiquetage de donnée.

Après l'exécution du script nous aurons des lignes de log étiquetées comme illustré sur la figure 4.5 :

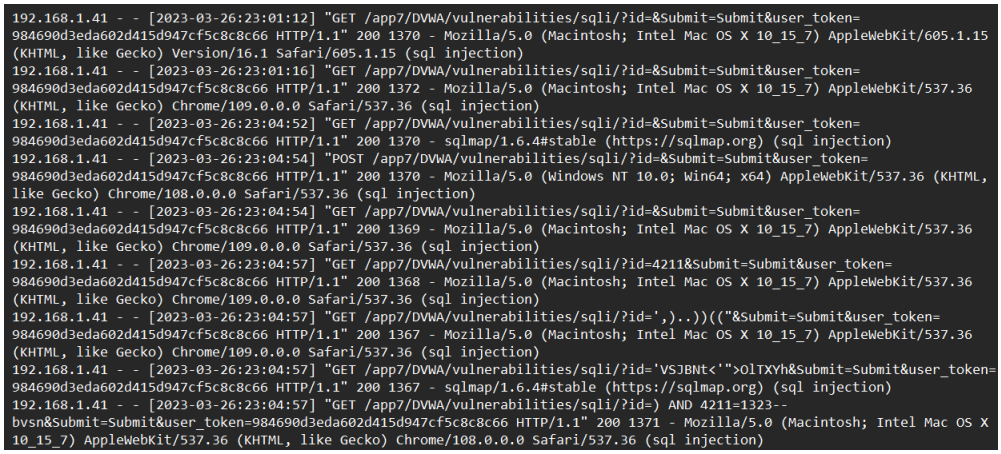

FIGURE 4.5 – lignes de log étiquetées avec leur type d'attaque.

4.3.6 Stockage des données

Après avoir formé nos lignes de log à partir des fichiers JSON et les avoirs étiqueté il faudra les stocker sur une base de données NoSql. Dans notre cas, nous avons opté pour ElasticSearch, qui offre une grande capacité de stockage, de recherche et d'analyse de données structurées et non structurées.
Dans le but d'automatiser le stockage, et ajouter des données dans elasticsearch, chaque ligne de log doit être précédé par une ligne de code de ce type :

{ "create" : { "_index" : "Logs", "_type" : "attaque", "_id" : 0} }

Le script python qui transforme le fichier log en un fichier JSON afin de le préparer pour son importation dans elasticsearch est détaillé dans l'annexe D.
A la fin nous aurons un dataset réel qui contient une variété de lignes de log, bien étiquetées par leur types d'attaques, dans le but de bien entraîner nos modèles.

4.3.7 Apports du dataset généré

Le dataset que nous avons généré constitue notre première contribution , il a été conçu dans le but de surmonter les inconvénients des datasets existants. L'environnement de génération est basé sur une topologie virtuelle contenant 4 machines virtuelles sous différents OS (Linux et Windows). Le point fort de notre Dataset est que chaque machine virtuelle contient un serveur web différent : Apache, Nginx, Python HTTP Servers et Tomcat, d'où la diversité lors de la collecte des données.
Le dataset ainsi généré, se compose de 528 535 lignes de logs , réparties sur 9 classes qui représentent la classe valide et les différents types d'attaques web, dont nous citons : SQL Injection, XSS, LFI, RFI, commande Injection, SSTI, etc.

Le manque de ressources nécessaires pour l'entrainement des modèles , nous a contraint de restreindre notre dataset lors de l'evaluation de la solution.

Le dataset ainsi considéré, contient 239 429 lignes de logs (soit 74,2 Mo), répartis sur 5 classes qui représentent la classe valide et 4 types d'attaques et ses variantes, à savoir :

- SQL Injection et ses variantes (blind SQLI, UNION based SQLI, TIME based SQLI, ERROR based SQLI, SECOND order SQL)
- Cross-Site Scripting (XSS) et ses variantes (XSS reflected, XSS stored, XSS dom based)
- LFI
- Commande Injection
- Valide

La figure ci-dessous présente la répartition des classes dans l'ensemble de données que nous avons considéré :

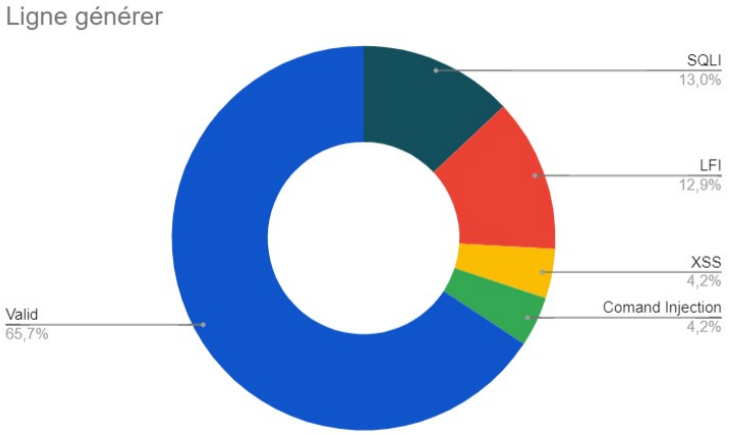

FIGURE 4.6 – Répartition des classes dans le dataset "CERIST_2023" considéré.

4.4 Implémentation des modules de la solution

Dans cette partie, nous allons expliquer comment nous avons implémenté les principaux modules de notre solution (module de prétraitement, analyse, supervision et affichage des résultats), et les paramètres utilisés pour chacun d'eux, ainsi que les résultats obtenus.

4.4.1 Module de prétraitement

4.4.1.1 Parsing des journaux

Dans cette phase, nous transformons les journaux d'une représentation non structurée en une représentation structurée comme le montre la figure 4.7.
Nous avons développé un parser, qui utilise la bibliothèque python "Parse", qui nous a permis de parcourir les champs de la ligne de log pour les séparer en colonnes dans un tableau. C'est

un tableau structuré où les colonnes représentent les attributs de notre dataset et les lignes représentent les lignes de log, sachant que le dernier attribut représente la classe qui peut avoir comme valeur 'normal' ou bien 'type d'attaque'.

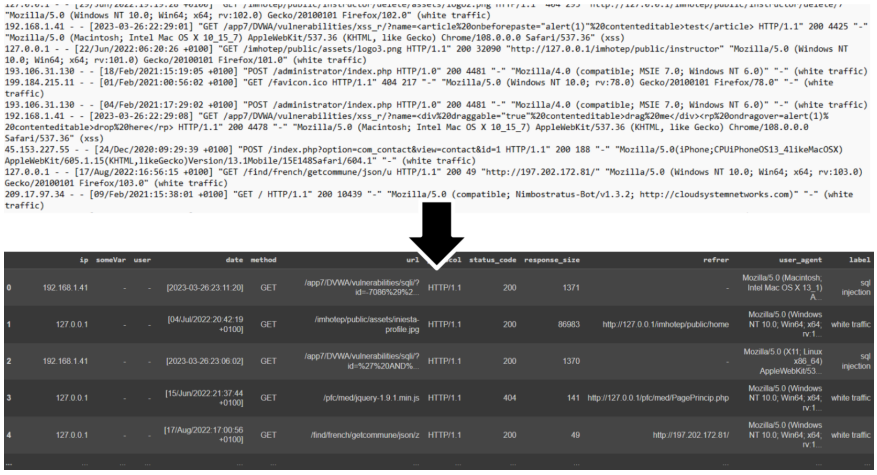

FIGURE 4.7 – Affichage des lignes de log parsés dans un tableau.

4.4.1.2 Nettoyage des données

Dans cette étape, nous avons nettoyé nos données en supprimant les lignes redondantes, et les attributs avec des valeurs manquantes comme 'le referer'. Les journaux peuvent ensuite être stockés dans un fichier csv (valeurs séparées par des virgules) à importer facilement la prochaine fois que nous en aurons besoin.

4.4.1.3 Extraction des caractéristiques

Les caractéristiques que nous avons jugées les plus pertinentes dans notre analyse, sont les suivantes :'url', 'response_size', 'user-agent', car la plupart des attaques envoient des requêtes malveillantes dans ces champs.

Quant à la numérisation, elle se porte sur les champs non numériques, qui sont : l'url et le user-agent. A partir des lignes de log nous obtenons les données d'apprentissages, qui sont des données comprises par les différents algorithmes ML. Pour cela nous avons développé un script python qui implémente la tokenisation par expression régulière et la codification TF-IDF. On aura comme résultat une matrice contenant la fréquence de tous les tokens, ce qui nous donne une meilleure signification sémantique. La figure 4.8 présente les deux fonctions développées avec une petite partie de l'expression régulière .

```
def tokenize(text):

tokens=re.findall(r'add|all|alter|and|select|between|by|case|check
|column|constraint|boot.ini|create|database|default|delete|desc|di
stinct)\b', text)

    return tokens

def tfidfCalculator(df):
    vectorizer =
TfidfVectorizer(stop_words=None,analyser=tokenize)
    vectorizer.fit(df['decoded_url'].tolist())
    X = vectorizer.transform(df['decoded_url'].tolist())
    ## Convertir la matrice sparse résultante en un DataFrame
    return pd.DataFrame(X.toarray(),columns=vectorizer.
    get_feature_names_out())
```

FIGURE 4.8 – Fonctions d'extraction et de numérisation des caractéristiques.

Après l'exécution des deux fonctions nous obtenons un tableaux de tokens, illustré dans la figure suivante.

	!	"	#	$	%	&	'	' or 1=1	(	)	...	void	where	whoami	with	yum	zip	{	|	}	~
0	0.0	0.000000	0.0	0.000000	0.0	0.056226	0.223888	0.0	0.055306	0.105875	...	0.0	0.047462	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
1	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.0	0.000000	0.000000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
2	0.0	0.000000	0.0	0.120381	0.0	0.125627	0.666989	0.0	0.123572	0.118280	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.490204	0.0	0.0
3	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.0	0.000000	0.000000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
4	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.0	0.000000	0.000000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
109551	0.0	0.000000	0.0	0.444327	0.0	0.347768	0.000000	0.0	0.342079	0.327429	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
109552	0.0	0.377083	0.0	0.000000	0.0	0.000000	0.000000	0.0	0.132292	0.126627	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
109553	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.0	0.000000	0.000000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
109554	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.0	0.000000	0.000000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
109555	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.0	0.000000	0.000000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0

FIGURE 4.9 – Tableau des tokens.

4.4.1.4 Normalisation

La normalisation a été effectuée pour l'attribut "Responce_size". Pour chaque valeur de cet attribut, nous avons calculé sa valeur normalisée et remplacé la valeur initiale par celle normalisée. Comme le présente le pseudo code de la figure 4.10 .

```
def normalize(df):
    scaler = MinMaxScaler(feature_range=(0, 1))
    df['response_size'] = scaler.fit_transform(df[['response_size']])

    return df
```

FIGURE 4.10 – Script de la normalisation.

A la fin du processus de prétraitement nous aurons une matrice d'apprentissage qui constitue l'entrée de nos modèles.

4.4.2 Module d'analyse

Les deux modèles d'analyse (classification des attaques, détection des anomalies) de notre système fonctionnent en parallèle, de cette façon les faux positifs du premier modèle n'affecteront pas les résultats du second modèle et vise versa, et à la fin de la prédiction des deux modèles nous comparons leurs résultats. Nous aurons donc trois scénarios :

- **Valide**
  Les entrées de journal détectées comme valides dans le modèle de classification des attaques et valides dans le modèle de détection des anomalies sont considérées comme des entrées de logs valides.
- **Attaque**
  Les entrées de journal détectées comme des attaques dans le modèle de classification des attaques et comme des anomalies dans le modèle de détection des anomalies sont considérées comme des attaques.
- **Anomalie**
  Les entrées de journal détectées comme des attaques dans le modèle de classification des attaques et comme valides dans le modèle de détection des anomalies, ou bien les entrées de journal détectées comme valides dans le modèle de classification des attaques et comme anomalies dans le second modèle sont considérés comme anomalies.

4.4.2.1 Module de classification d'attaques

Pour implémenter le module de classification, nous avons testé plusieurs algorithmes supervisés sur notre dataset, afin de comparer leurs performances et prendre le meilleur modèle pour la classification des attaques.

Les algorithmes d'apprentissage testés sont les suivants : SVM, KNN, Decision Tree, et RandomForest.

Dans l'annexe B, nous détaillons le fonctionnement des différents algorithmes d'apprentissage supervisé que nous avons testés.

Tests et expérimentations

Les étapes suivies pour l'entrainement et la validation de chaque modèle sous python sont les suivantes :

1. Diviser l'ensemble de log prétraités en ensemble d'entraînement (80%) et ensemble de test (20%).
2. Instancier l'algorithme à tester, et lancer l'entraînement avec l'ensemble d'entraînement.
3. Lancer le modèle créé sur l'ensemble de test.
4. Calculer les mesures de performance du modèle créé.

5. Construire la matrice de confusion pour chaque algorithme testé, pour montrer les résultats de prédictions.

Les figures 4.11, 4.12, 4.13 et 4.14 présentent respectivement les matrices de confusion des algorithmes "SVM", "KNN", "Arbre de décision" et "Forêt aléatoire".

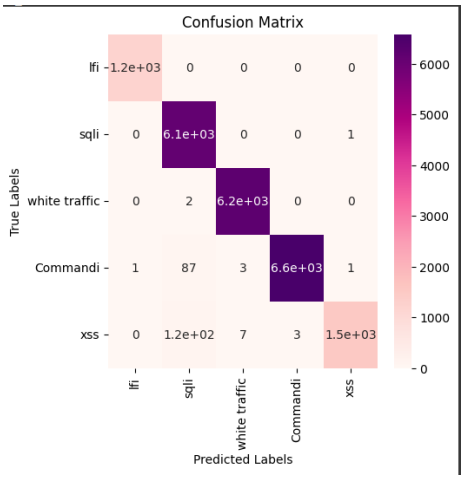

FIGURE 4.11 – Matrice de confusion du SVM.

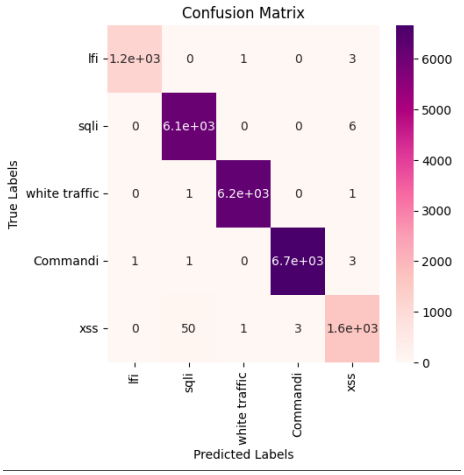

FIGURE 4.12 – Matrice de confusion du KNN.

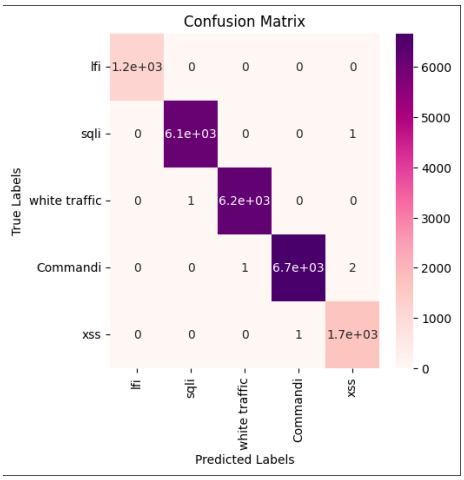

FIGURE 4.13 – Matrice de confusion de l'arbre de décision.

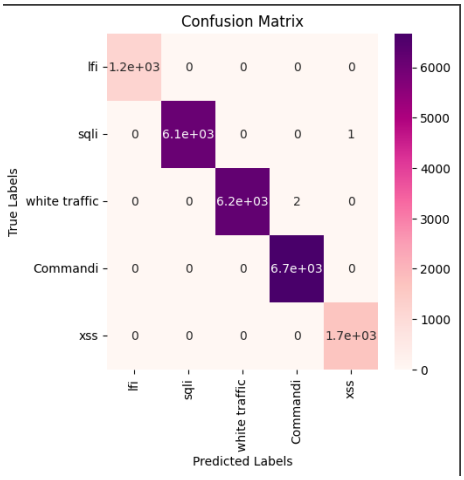

FIGURE 4.14 – Matrice de confusion de la forêt aléatoire.

Les résultats des différentes mesures de performance de nos modèles sont récapitulés dans le tableau 4.1 .

	SVM	**Decision Tree**	**KNN**	**Random Forest**
Accuracy	89%	95%	90%	98%
Précision	86%	96%	91%	99%
Rappel	82%	94%	87%	98%
F-measure	83%	94.5%	89%	98%

TABLE 4.1 – Tableaux de comparaison de performance des 4 modèles.

Pour avoir une idée plus précise sur les performances du modèle Random Forest généré, nous avons mis le modèles dans un environnement de production après la sérialisation. Il sera testé sur un nouveau log web qu'il n'a jamais rencontré auparavant, ceci est dans le

but de tester si il n'est pas affectés par le sur-apprentissage. Le sur-apprentissage se produit lorsque le modèle prédictif pourra donner de très bonnes prédictions sur les données d'entrainement (les données qu'il a déjà vu et auxquelles il s'y est adapté), mais il prédira mal sur des données qu'il n'a jamais vu lors de sa phase d'apprentissage.

Dans le tableau suivant nous présentons les résultats de prédictions sur un nouveau fichier log contenant 10 000 lignes. La taille de chaque modèle sérialisé est aussi mentionnée

	Random Forest
Fausse prediction	10 / 10 000
Prediction juste	9 990 / 10 000

TABLE 4.2 – Résultats de prédictions.

	Random Forest	**KNN**	**SVM**	**Decision Tree**
La taille du modèle sérialisé	6,38 Mo	299,96 Mo	11 Mo	5.44 Ko

TABLE 4.3 – Taille de chaque modèle sérialisé.

Discussion

Selon les résultats obtenus, nous constatons que le Random forest a mieux classé les entrées du log par rapport aux autres algorithmes. En effet, le taux de faux positif produit par le modèle "Random forest" est le plus faible. Par ailleurs, la taille du fichier du modèle "Random forest" après sérialisation est à 6.38 MB, optimal par rapport aux autres. De plus, le modèle "Random Forest" a donné de meilleurs résultats de prédictions sur les nouvelles entrées de log, ce qui prouve qu'il généralise mieux sur de nouvelles données.

Pour ces raisons, nous avons choisi l'algorithme de " RandomForest" comme modèle de classification des attaques.

Sauvegarde du modèle choisi " Random Forest"

Un modèle s'interpréte sous forme d'un fichier ".joblib" généré par l'algorithme après avoir été entraîné sur le dataset. Afin que notre modèle puisse être mis en production et testé sur les nouvelles données, il doit d'abord être sauvegardé. Cela a été fait grâce à l'utilisation de la bibliothèque "joblib" qui sérialise l'objet modèle et le sauvegarde dans un fichier ".joblib". La fonction qui le sauvegarde suit la structure suivante :

```
import joblib
# serialization of model
joblib.dump(ModelObject,'model.joblib')
# deserialization of model
saved_model = joblib.load('model.joblib')
```

FIGURE 4.15 – Fonction de sauvegarde du modèle.

Grâce à cette fonction, avec le fichier de sauvegarde "model.joblib", n'importe quel code python peut charger le modèle et après avoir appliqué le tokenisation et l'encodage des données, le modèle sera prêt à prédire et détecter.

4.4.2.2 Module de détection d'anomalies

Ce module sert à détecter les attaques inconnues et les anomalies au sein du système dont le premier module (le module de classification d'attaques) ne détecte pas. Pour ce faire, nous avons testé trois algorithmes non supervisés à savoir : "DBSCAN" , "LOF" et "AutoEncoder". Nous avons comparé leurs performances afin de choisir le meilleur entre eux.

Tests et expérimentations

Pour cette partie, nous avons utilisé le dataset valide, que nous avons répartis en deux ensembles : 80% ensemble d'entraînement et 20% ensemble de test.

Dans la phase d'entraînement, nous avons pris que les entrées de journal "Valide", pour modéliser le comportement normal. Par contre, dans la phase de test, nous avons pris les deux type de logs "valide" et "Anomalie".

1. **DBSCAN :**

   L'algorithme a été sélectionné car il peut étiqueter les données de bruit par lui-même. Une donnée de bruit est un point qui n'appartient à aucun cluster et dans notre cas, nous étiqueterons les données de bruit sélectionnées comme des anomalies. Le nombre de clusters est également déterminé par l'algorithme. Cependant, nous devons définir les valeurs de deux paramètres, eps et min_sample.

   Après avoir testé plusieurs valeurs de ces deux paramètres (Tunning), et comparé les résultats de chacun, les meilleurs résultats de précision ont été obtenus avec les valeurs suivantes :

   eps = 1.3 : L'augmentation de la valeur eps permet d'inclure dans le même voisinage des points plus éloignés les uns des autres, ce qui se traduit par des clusters plus dispersés. Si la valeur eps est diminuée, les clusters résultants seront plus denses et plus compactes. Le paramètre "eps" joue un rôle important dans la détermination de la taille d'un voisinage et des clusters qui en résultent. Dans notre cas, après plusieurs tests,la valeur 1,3 nous a donné une bonne précision.

   min_samples = 5 : DBSCAN considère les clusters comme des zones de haute densité séparées par des zones de faible densité. Les zones à faible densité sont indiquées comme potentiellement des points de bruit ou abberrantes. Le paramètre min_sample définit formellement ce que nous entendons par dense. Il représente le nombre minimal d'échatillons qui peuvent exister dans un cluter, Une valeur élevé de min_sample indique une densité plus élevée nécessaire pour former un cluster. Ce paramètre contrôle principalement la tolérance de l'algorithme au bruit. Dans notre cas, il a été fixé à 5 par manque de ressources après avoir testé plusieurs valeurs.

   Le tableau 4.2 montre les résultats des différentes mesures d'évaluation du modèle "DBSCAN".

Taux d'erreur	3/5309 lignes
Taux de détection	5306/5309 lignes

TABLE 4.4 – Mesure des performances pour le DBSCAN

2. **LOF :**
   En effectuant des tests sur divers paramètres pour l'algorithme LOF et en comparant les résultats de chacun, il a été déterminé que les résultats les plus efficaces ont été produits en utilisant le paramètre suivant :
   n_neighbors = 5 : C'est le nombre de voisins pris en compte lors du calcul de la densité locale d'un point de données. pour notre cas on a fixer cette valeur à 5 c'est la valeur la plus optimale.
   Le tableau 4.3 montre les résultats des différentes mesures d'évaluation du modèle "LOF".

Taux d'erreur	954/5337 lignes
Taux de détection	4383/5337 lignes

TABLE 4.5 – Mesure des performances pour le LOF

3. **L'Auto-encodeur :**
   L'auto-encodeur est un type spécifique de réseaux de neurones qui essaie de créer les valeurs d'entrée dans la sortie, tout en minimisant l'erreur de reconstruction. Cet avantage peut être utilisé pour la détection d'anomalies en suivant les étapes suivantes :
   — Entraîner l'auto-codeur en utilisant uniquement des données valides, et calculer la perte d'entraînement de chaque donnée.
   — Il est important de définir un seuil, car il est impossible d'entraîner le modèle avec toutes les données valides. Nous l'avons calculé comme suit [71] :
     (a) Calculer la moyenne de la perte d'entraînement de tous l'ensemble de données d'entraînement.
     (b) Calculer les écarts-types de la perte d'entraînement .
     (c) Le seuil est la somme de la moyenne de la perte d'entraînement et des écarts-types : threshold = np.mean(train_loss) + np.std(train_loss)

   Le modèle peut utiliser maintenant le seuil pour identifier les données "valide" des données "d'anomalie" comme suit :
   — Encoder et décoder les nouvelles données en utilisant l'encodeur et le décodeur de l'auto-encodeur.
   — Comparer l'erreur de recontruction résultante avec le seuil :
     Si erreur $<$ seuil : Valide
     Sinon erreur $>$ seuil : Anomalie

   Nous avons testé plusieurs paramètres de cet algorithme tels que : le nombre de couches, le nombre de neurones par couche, la fonction d'activation...etc. Le tableau

suivant nous indique les meilleurs valeurs des paramètres de l'algorithme "Auto-encodeur", que nous avons adoptés :

seuil	0.0004695349
epochs	10
optimiseur	Adam
fonction de perte	MSE (Mean Square Error)

TABLE 4.6 – Meilleurs paramètres de l'Auto-encodeur

L'auto-encodeur ne peut pas bien codifier l'anomalie, car Il a appris à représenter que les données valide (comportement normal). Lorsque nous essayons de reconstruire les données d'anomalies à partir de leur représentation courte, la reconstruction ne ressemblera pas aux données d'origine. Ce qui aide ainsi à détecter les anomalies (perte de reconstruction $>$ seuil)

Le tableau suivant montre les résultats des différentes mesures d'évaluation pour l'auto-encodeur.

Accuracy	91,03%
Précision	93,31%
Rappel	91,01%
F-measure	92,66%

TABLE 4.7 – Mesures d'évaluation de l'algorithme Auto-encodeur

Etude comparative

Le tableau suivant présente un récapitulatif de tous les résultats des différentes mesures d'évaluation pour les trois méthodes de détection d'anomalies que nous avons testées.

	DBSCAN	**LOF**	**Auto-Encoder**
Taux d'erreur	3/5309 lignes	954/5337 lignes	5/10000 lignes
Taux de détection	5306/5309 lignes	4383/5337 lignes	9995/10000 lignes
Précision	92%	82,12%	93,31%

TABLE 4.8 – Mesures des performances des trois algorithmes.

Selon le tableau comparatif, nous avons constaté que l'algorithme "Auto-encoder" donne de meilleur résultat. Cela signifie que l'algorithme "Auto-encoder" est meilleur lorsqu'il s'agit de détecter des anomalies à partir de journaux, et qu'il a moins de faux positifs que les deux autres algorithmes.

Pour ces raisons, nous avons choisis l'algorithme "Auto-encodeur" comme modèle de détection d'anomalies.

4.4.3 Module de supervision et affichage des résultats avec la plateforme MDP

Pour permettre à l'administrateur d'interagir avec les modules déjà présentés ci-dessus, nous avons développé notre propre plate-forme MDP "Monitoring Detection Platform" ou plateforme de détection et de supervision. La figure 4.16 présente l'architecture de notre application.

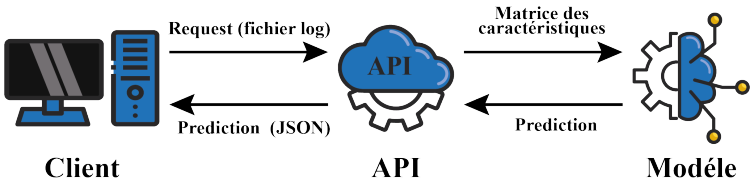

FIGURE 4.16 – Architecture de la plateforme MDP.

MDP est une application web, déstinée à un client web (l'administrateur dans notre cas) qui se connecte à un serveur web via une API REST. Le serveur web gère les requêtes provenant du client et les transmet à nos modèles d'analyse, et ensuite affiche les résultats.
L'interface MDP permet de :

- Uploader un fichier log.
- Lancer l'analyse d'un fichier log.
- Choisir le fichier log pour lancer l'analyse.
- Afficher les entrées de journal avec les attaques et les anomalies détectées.
- Afficher les statistiques.
- Télécharger le fichier analysé sous format CSV (Comma Separated Values).

Une description complète de toutes les interfaces fournies dans notre plateforme seront présentées dans les sections suivantes.

4.4.3.1 Interface d'authentification

L'interface présentée dans la figure 4.17, permet l'accés à la plateforme après une authentification réussie.

FIGURE 4.17 – Interface d'authentification.

4.4.3.2 Interface d'accueil

Depuis l'interface illustrée dans la figure 4.18, l'administrateur peut uploader un fichier journal brut pour être analyser. Les fichiers uploadés seront sauvegardés dans notre serveur. Plusieurs formats de journal sont pris en charge par notre système.

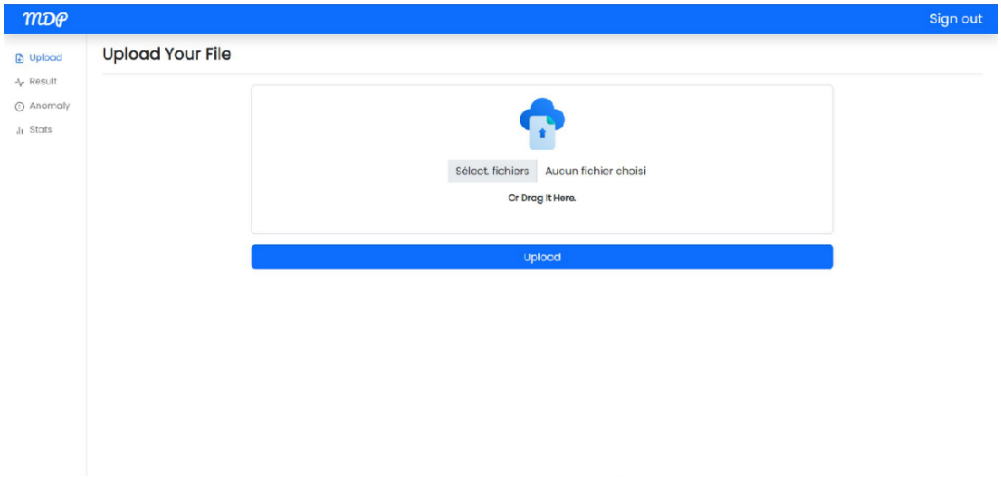

FIGURE 4.18 – Interface d'accueil.

4.4.3.3 Interface d'analyse et de visualisation des résultats

Dans l'interface de la figure 4.19, l'administrateur peut choisir un fichier log pour l'analyser (parmi ceux dèjà uploadés). C'est dans cette phase que nos modèles d'analyse s'exécutent. Une fois l'analyse achevée, l'aministrateur peut consulter et examiner les résultats qui s'affichent sur l'interface. Il pourra sauvegarder ces résultats sous format CSV pour une utilisation ultérieure.

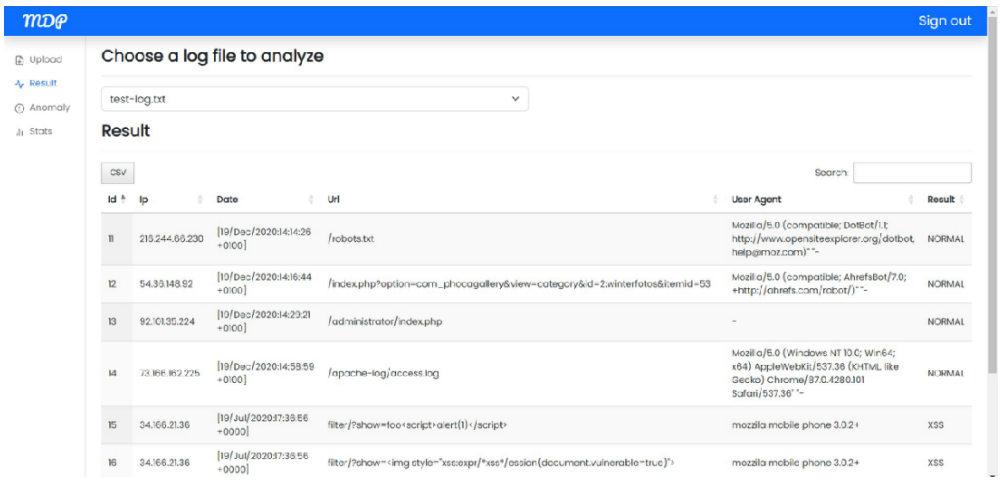

FIGURE 4.19 – Interface d'analyse et de visualisation des résultats

L'interface offre à l'administrateur un ensemble de fonctionnalités avancées de recherche (par adresse IP, type d'attaque, etc.).

4.4.3.4 Interface de gestion des conflits de prédictions

Cette interface (figure 4.20) permet la consultation des conflits de prédictions entre les deux modèles (classification et détection d'anomalies).
Pour la même ligne de log, nous affichons les résultats des deux modèles, ce qui permet à l'administrateur de prendre la bonne décision selon le cas observé :

1. Une nouvelle attaque ou attaque prédéfinie : La ligne de log sera étiquetée par type d'attaque appropriée et sauvgardée dans l'ensemble de réapprentissage du modèle de classification d'attaque.
2. Une entrée de log valide : la ligne de log est étiquetée valide et sauvgardée dans l'ensemble de réapprentissage du modèle de détection d'anomalies.

De plus, cette interface offre à l'administrateur un ensemble de fonctionnalités avancées de recherche et de filtrage.

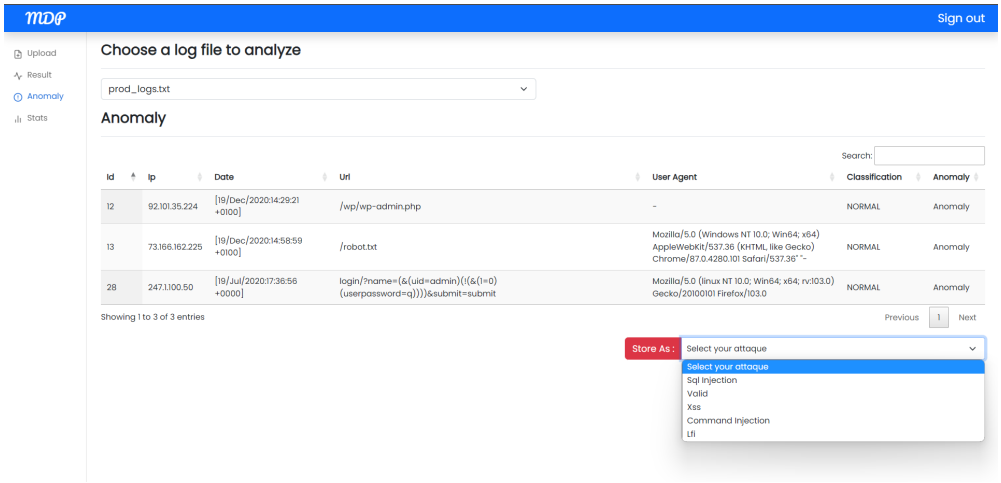

FIGURE 4.20 – Interface de gestion des anomalies

4.4.3.5 Interface des statistiques

L'interface de statistiques offre différentes représentations graphiques des statistiques concernant la répartition des attaques , des anomalies détectées, etc.

En fonction de ces statistiques, l'administrateur de sécurité peut prendre les décisions adéquates pour prévenir les cyber-menaces.

La figure 4.21 et la figure 4.22 montre le graphique qui donne le pourcentage de chaque type d'attaque, et le pourcentage des anomalies détectées.

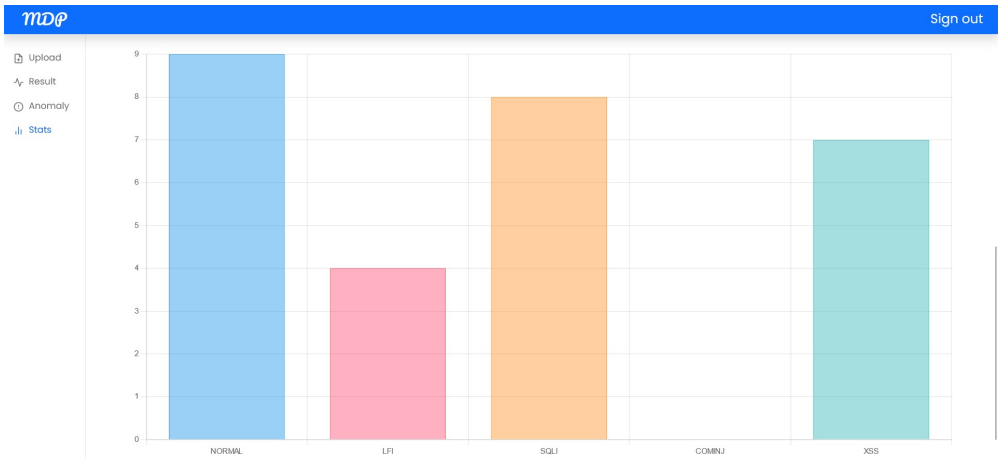

FIGURE 4.21 – Interface des statistiques

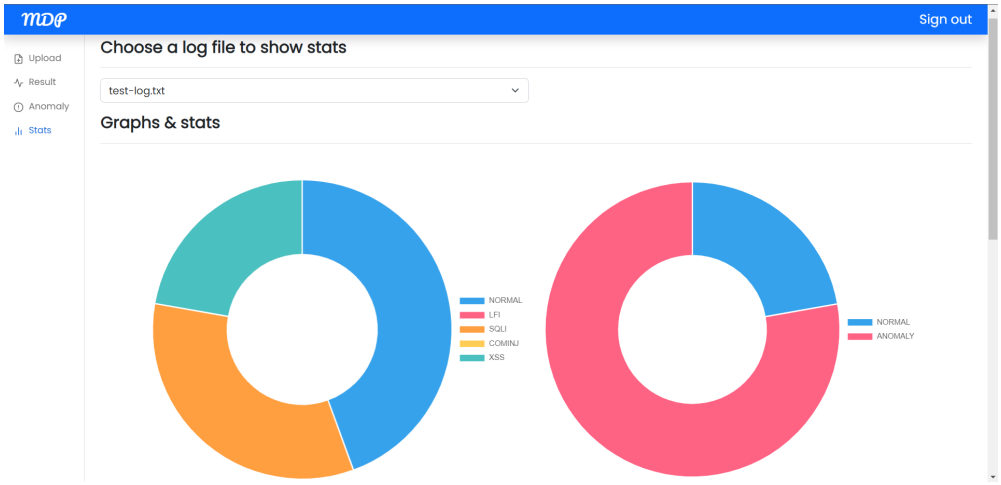

FIGURE 4.22 – Interface des statistiques

4.5 Conclusion

Dans ce chapitre, nous avons expliqué les fonctionnalités des différentes parties de notre architecture et leur mise en œuvre. Nous avons testé chaque partie séparément, de la classification des attaques à la détection des anomalies en passant par le parsing, et l'extraction des caractéristiques. À la fin du chapitre, nous avons exposé l'interface graphique qui offre plusieurs fonctionnalités à l'expert de sécurité.

Conclusion générale

La complexité croissante des cyber-attaques et leur évolution de plus en plus rapide ont motivé un intérêt accru pour améliorer les techniques et les méthodes de réduction des cyber-risques qui devient extrêmement difficile.

En effet, les mesures traditionnelles de cyber-sécurité sont précieuses, surtout lorsqu'elles se cumulent. Mais bien que ces techniques soient efficaces contre les menaces déjà connues, elles restent insuffisantes face à l'évolution quotidienne des risques, et les nombreuses failles découvertes et exploitées chaque jour : il faut effectivement traiter d'énormes quantités d'évènements variant dans le temps. Par conséquent, analyser et améliorer la posture de cyber-sécurité n'est plus une tâche à l'échelle humaine. La cyber-sécurité doit être renforcée par de nouvelles formes de protection numérique.
C'est dans cette optique, que nous avons proposé une solution qui se fonde sur l'utilisation des technologies de l'IA principalement le Machine Learning (ML) et le traitement du langage naturel pour : l'analyse des données, la détection d'attaques/anomalies et la supervision dans le but d'identifier et d'explorer les incidents de sécurité ou les menaces potentielles.

La solution envisagée repose sur l'exploitation des logs, qui constituent une source très précieuse d'informations, souvent sous-estimés. Cependant, leur traitement et leur analyse permettent de découvrir des problèmes non visibles dans l'utilisation quotidienne.

- SYNTHÈSE DU TRAVAIL RÉALISÉ

Les majeures contributions de ce mémoire peuvent se résumer :

- **Etude théorique :** Nous avons mené une étude approfondie sur des travaux récents qui ont souligné que la performance des modèles ML dépend fortement de la qualité des données d'entraînement, la bonne sélection des attributs et de la capacité des algorithmes à généraliser les modèles pour détecter de nouvelles attaques. Des critères que nous avons pris en considération lors de la conception de notre solution.
- **Génération du DataSet (CERIST_2023) :** L'un des plus grands défis auxquels nous avons été confrontés était le manque de datasets de logs web récents, étiquetés et accessibles au public. Ce qui nous a conduit à implémenter un environnement capable de générer un nouveau dataset adaptés au contexte de notre solution.
- **Fonctions avancées de prétraitement des données :** Nous avons porté un intérêt particulier aux différentes étapes du processus de prétraitement des données,

notamment la tokenisation, où nous avons mené une étude approfondie sur la syntaxe des principales attaques web, et par la suite nous avons créé une liste qui contient plus de 250 tokens (mots et caractères spéciaux) contenus dans la majorité des payloads des attaques web, qui a été traduite en une expression régulière. Cette étape nous a permi de préserver la sémantique et la généralisabilité des tokens.

- **Module d'analyse :** Notre analyse consiste à combiner deux types de modèles d'apprentissage automatique qui fonctionnent en parallèle, le premier modèle supervisé (modèle de classification d'attaques)permet de détecter des attaques Web connues, il est limité à la détection d'attaques déjà connues dans la phase d'entrainement. Le deuxième modèle non supervisé (modèle de détection d'anomalies) se base uniquement sur les comportements des utilisateurs pour détecter une variation par rapport aux comportements normaux qui représentent une anomalie, ce qui permet de détecter de nouvelles attaques. Une entrée de log qui est détectée comme une anomalie par ce modèle mais détectée comme valide par le modèle de classification est susceptible d'être une nouvelle attaque ou pourrait même être une attaque zéro-day.
- **Module de réapprentissage :** Sachant que le comportement des cyber-attaques évolue continuellement, et afin d'éviter que nos modèles d'apprentissage automatique deviennent obsolètes au fil du temps. Nous avons proposé ce module qui permet de réentraîner et redéployer par la suite, notre modèle dés qu'une nouvelle forme d'attaque est reconnus par l'administrateur.
- **Module de supervision :** Représente la couche de visualisation, qui offre une visibilité continue, afin de mieux comprendre, interagir et gérer les cyberattaques. Pour cela, une interface graphique qui offre plusieurs fonctionnalités a été développée.

- PERSPECTIVES

En perspective, il serait intéressant d'apporter des améliorations aux performances de notre système comme :

— Améliorer la précision du système, et minimiser la détection des faux positifs.
— Déploiement de la solution sur des serveurs en production (serveur du CERIST).
— Entraîner le système à détecter de nouvelles formes attaques, autres que celles qu'il peut détecter maintenant.

Annexe A

Critères d'évaluation des modèles ML

Faces aux diverses méthodes d'apprentissage, plusieurs critères d'évaluation et de comparaison des performances des modèles ont été mis en œuvre. Cette évaluation consiste à une évaluation empirique des modèles.
Pour valider ou tester un classifieur, l'ensemble de données est généralement divisé en deux sous ensemble : l'ensemble d'apprentissage sur lequel le classificateur fait son apprentissage et l'ensemble de test sur lequel nous pouvons évaluer sa performance. La performance du classifieur peut être analysée en considérant les erreurs commises sur l'ensemble de test. Car ce dernier contient les données dont on connaît à l'avance les classes auxquelles elles devraient appartenir. Nous pourrons donc comparer les décisions prises par le classificateur automatique à celles des experts humains et calculer un score de performance. Dans ce contexte, nous définissons les métriques que nous avons utilisées pour évaluer nos modèles :

1. **Matrice de confusion**
   Il s'agit d'une matrice qui est souvent utilisée pour décrire le résultat des prédictions d'un modèle d'IA (classificateur qui impliquent deux classes ou plus en sortie) dans l'ensemble de test, avec des détails sur chaque classe en utilisant les paramètres suivants [48] :
   - **Vrais positifs (VP) :** prédits positifs et réellement positifs.
   - **Faux positifs (FP) :** prédits positifs et réellement négatifs.
   - **Vrais négatifs (VN) :** prédits négatifs et réellement négatifs.
   - **Faux négatifs (FN) :** Prédits négatifs et en réalité positifs.

   La matrice indique le nombre de prédictions correctes et incorrectes pour chaque classe en fonction de la classe prédite. Le tableau 4 montre un exemple de matrice de confusion pour la classification binaire.

Classe/Prédits	Positive	Négative
Positive	Vrai positif (VP)	Faux positif (FP)
Négative	Faux négatif (FN)	Vrai négatif (VN)

Table 9 – Matrice de confusion.

2. **Accuraccy**

   L'accuraccy, est la fréquence à laquelle le classificateur (le modèle) fait la bonne prédiction, dans l'ensemble de test en utilisant l'ensemble d'entrainement (c'est-à-dire que le modèle peut généraliser de nouvelles données en utilisant seulement les données d'entrainement) [48].

   Il s'agit de la proportion de classifications correctes par rapport au nombre total de classifications effectuées.

$$Accuracy[48] = \frac{VP + VN}{VP + FP + FN + VN}$$

3. **Precision**

   Elle évalue le pouvoir prédictif d'un modèle d'IA en estimant la valeur prédictive d'une étiquette, autrement dit, la capacité d'un modèle à ne pas identifier de faux positifs. C'est le rapport entre les instances positives (négatives) et le total des instances positives (négatives) prédites [48].

$$Precision[48] = \frac{VP}{VP + FP}$$

4. **Rappel**

   Évalue l'efficacité du modèle dans une classe spécifique en calculant la probabilité que l'étiquette positive soit vraie [48]. Autrement dit c'est la capacité du modèle à ne pas manquer de vrais positifs. C'est le rapport entre les instances positives et le total des instances positives réelles :

$$Rappel[48] = \frac{VP}{VP + FN}$$

5. **Le Score F_1**

   Il prend en compte la contribution de la précision et du rappel, de sorte que plus la valeur F1 est élevée, meilleur est le résultat [48]. Le modèle obtient un F1 élevé lorsque les prédictions positives sont réellement positives (précision) et que le modèle ne rate pas beaucoup de valeurs positives (rappel).

$$ScoreF_1[48] = \frac{2VP}{2VP + FP + FN} = \frac{2 * prcision * rappel}{prcision + rappel}$$

Annexe B

Algorithme d'apprentissage automatique

Arbre de Décision "Decision Tree"

L'arbre de décision (Decision tree ou DT) est un modèle prédictif populaire pour la classification qui utilise les caractéristiques de l'ensemble de données d'apprentissage pour créer un ensemble de règles de décision. Chaque nœud dans l'arbre représente une caractéristique et chaque branche représente une valeur possible de cette caractéristique. Les mesures de gain d'information et d'impureté de Gini sont utilisées pour sélectionner les meilleurs nœuds à chaque étape de la construction de l'arbre. L'arbre CART est un algorithme d'arbre de décision couramment utilisé [69]. Une définition plus formelle des DT est la suivante : un arbre décisionnel est un graphique sans cycles dans lequel les nœuds représentent une question, les arcs représentent les réponses et les feuilles représentent des conclusions ou des terminaux de classes [63].

K-Nearest Neighbor (KNN)

En raison de sa facilité d'implémentation, l'approche KNN est fréquemment utilisée dans les applications d'extraction de données et d'apprentissage automatique. [64]. L'idée fondatrice est qu'une donnée de classe inconnue est comparée à toutes les données stockées pour choisir la classe majoritaire parmi ses K plus proches voisins. Le principe de cette méthode est de faire voter les plus proches voisins d'une observation. La classe de x est déterminée en fonction de la classe majoritaire parmi les k plus proches voisins de l'observation x. En d'autres termes, en ayant un ensemble de données d'apprentissage D, une fonction de distance d et un entier k, pour tout nouveau point de test x, l'algorithme recherche dans D les k points les plus proches de x au sens de la distance d (Euclidienne, Manhattan, Minkowsky, Hamming) et attribue à x la classe qui est la plus fréquente et majoritaire parmi ces k voisins [70].

Forêt aléatoire ou "Random Forest"

La Forêt aléatoire (Random Forest) est l'un des algorithmes d'apprentissage automatique supervisé les plus populaires et les plus puissants. Le foret aléatoire se base sur les arbres de décision (vu dans la section précédente). Elle consiste à créer plu-

sieurs arbres de décision indépendants et d'agréger leurs résultats pour prédire une réponse. Les arbres de décision sont rendus différents en sélectionnant aléatoirement un sous-ensemble de caractéristiques à chaque nœud, ce qui permet d'éviter le sur-apprentissage. La réponse finale est déterminée par un vote majoritaire des arbres de décision [65].

Support Vector Machine (SVM)

La proposition initiale de la méthode du vecteur support a été faite en 1965 par V. Vapnik. Le but était de traiter les problèmes liés à la reconnaissance de formes. Par la suite, Kimeldorf a introduit l'idée de construire un espace noyau avec des vecteurs de support en 1971. Dans les années 1990, V. Vapnik a officiellement introduit la méthode de la machine à vecteurs de support (SVM) pour l'apprentissage statistique. Depuis lors, les SVM ont été largement utilisées dans divers domaines, notamment le traitement du langage naturel et la reconnaissance de formes. Il est utilisé pour la classification et la régression. Il trouve un hyperplan qui catégorise précisément les données en maximisant la distance entre l'hyperplan et les points de données les plus proches, appelée marge. L'hyperplan est ensuite utilisé pour classer de nouvelles données non étiquetées [66].

DBSCAN :

DensityBased Spatial Clustering of Applications with Noise (DBSCAN) est un algorithme de clustering basé sur la densité. Les clusters sont décrits comme des zones contiguës présentant une forte densité de points de données. Lorsque la densité est inférieure à un certain niveau, les données sont considérées comme un bruit. L'algorithme DBSCAN a été construit dans le but de découvrir dans l'ensemble de données des clusters de taille arbitraire et un ensemble de points de bruit. Il nécessite de définir manuellement deux paramètres eps et MinPts. Le paramètre eps définit le rayon d'un voisinage autour d'un point de données et le paramètre MinPts définit le nombre minimum de voisins dans le rayon eps (ou seuil de densité). Pour localiser les clusters, DBSCAN commence par initialiser un point arbitraire et continue à former des clusters jusqu'à ce que les critères eps et MinPts soient satisfaits. Les points qui ne sont pas dans un cluster seront classés comme bruit. [67].

LOF : "Local Outlier Factor" ou Facteur local aberrant :

Le facteur local aberrant est une méthode de détection d'anomalies non supervisée qui calcule l'écart de densité local d'un point de données par rapport à ses voisins (abbération). Elle est basée sur la technique de densité et sur la base de la valeur de LOF et du seuil fixé, pour décider si le point est une anomalie ou un point valide.
Le LOF est utilisé pour de nombreuses applications telles que la détection des fraudes, les soins médicaux et le traitement des images.

Il existe deux types de valeurs aberrantes : les valeurs aberrantes locales et les valeurs aberrantes globales.

1. **Les valeurs aberrantes globales :** sont les points qui sont très différents de l'ensemble des données.
2. **Les valeurs aberrantes locales :** sont des points qui ne sont pas éloignés de l'ensemble des données mais qui sont éloignés des clusters.

Le LOF est utilisé pour de nombreuses applications telles que la détection des fraudes, les soins médicaux et le traitement des images. [68]

Auto-encodeur :

est un type de modèle d'apprentissage automatique non supervisés à base de réseaux de neurones artificiels (RNN exactement), utilisé pour coder les données d'entrée (il peut s'agir d'une image, d'un vecteur ou de tout autre type de données) en une représentation plus petite. À partir de cette représentation, il essaie de les décoder pour recréer les données d'entrée avec une perte de reconstruction minimale, en actualisant les poids chaque fois que l'erreur est supérieure à un seuil, et ceci en utilisant la rétropropagation. Les auto-encodeurs sont des algorithmes non supervisés, car ils ne nécessitent pas des données étiquetées, et auto-supervisés, car ils comparent les données de sortie aux données d'entrée pour calculer la perte et mettre à jour les poids.
Les auto-encodeurs peuvent être utilisés pour de nombreuses applications, telles que l'extraction de caractéristiques, la suppression de bruit dans les données, et la détection des anomalies.

Les auto-encodeurs se composent de deux éléments principaux : l'encodeur et le décodeur.

Il y a 4 hyperparamètres que nous devons définir pour entraîner l'auto-encodeur :

1. **La taille du code :** Nombre de neurones dans la couche intermédiaire.
2. **Le nombre de couches :** Nous pouvons définir n'importe quel nombre de couches pour les parties encodeur et décodeur.
3. **Le nombre de neurones pour chaque couche :** Le nombre de neurones dépend des types d'auto-encodeurs avec lesquels nous travaillons, par exemple pour les auto-encodeurs empilés le nombre de neurones diminue dans l'encodeur et augmente dans le décodeur.
4. **La fonction de perte :** Qui nous donne la différence entre les données d'entrée et les données de sortie, nous utilisons soit l'erreur carrée moyenne (mean square error (mse)) ou l'entropie croisée binaire.

Un auto-encodeur peut être entraîné en suivant les étapes suivantes :

1. Définir les 4 hyperparamètres, et sélectionner les données d'entrée qui seront utilisées pour entraîner le modèle.
2. L'auto-encodeur lit une ligne de données et l'insère dans l'encodeur et essaie de créer une représentation courte de ces données dans le goulot d'étranglement.
3. L'auto-encodeur insère la représentation créée dans le décodeur et tente de créer à nouveau les données insérées.
4. calculer la fonction de perte entre les données de sortie et les données d'entrée.
5. Mettre à jour les poids en utilisant la rétro-propagation.
6. Répéter les étapes de 2 à 5 avec toutes les données.

Annexe C

Extraction des variables de la couche applicative pour former les lignes de logs

1. Convertir le fichier .PCAP généré par TCPDUMP en un fichier structuré sous format JSON qui contient toutes les informations des paquets (Requête / Réponse) afin de pouvoir par la suite parcourir les champs de la requête facilement. Le script python montré sur la figure ci-dessous, nous a permis de faire la transformation en utilisant la bibliothèque PCAPkit, qui est pratique pour extraire des informations à partir de captures de paquets réseau.

```python
from pcapkit import extract
import json as j
import zlib

json = extract(fin='wfuzz_brut.pcap', fout='wfuzz_brut.json', format='json', extension=False)
```

FIGURE 23 – Script de transformation du fichier PCAP.

Le fichier JSON est créé. comme montré sur la figure suivante :


```json
"http": {
    "receipt": {
        "type": "request",
        "method": "GET",
        "uri": "/app7/DVWA/vulnerabilities/sqli/?id=&Submit=Submit&user_token=984690d3eda602d415d947cf5c8c8c66",
        "version": "1.1"
    },
    "header": {
        "Cache-Control": "no-cache",
        "User-Agent": "sqlmap/1.6.4#stable (https://sqlmap.org)",
        "Referer": "http://192.168.1.38:80/app7/DVWA/vulnerabilities/sqli/?id=&Submit=Submit&user_token=984690d3eda602d415d947cf5c8c8c66",
        "Host": "192.168.1.38",
        "Accept": "*/*",
        "Accept-Encoding": "gzip,deflate",
        "Connection": "close"
    },
    "body": null
}
```


FIGURE 24 – Extrait d'un fichier JSON.

Pour former nos lignes de log, nous avons développé un script qui va parcourir les attributs de notre fichier JSON afin de récupérer leur valeur comme le montre le code suivant :

```
try:
    if (jsonFile[packet]['ethernet']['ipv4']['tcp']['http']['receipt']['type'] == "request"):
        it+=1
        self.uri = jsonFile[packet]['ethernet']['ipv4']['tcp']['http']['receipt']['uri']
        self.host = jsonFile[packet]['ethernet']['ipv4']['tcp']['http']['header']['Host']
        self.method = jsonFile[packet]['ethernet']['ipv4']['tcp']['http']['receipt']['method']
        self.userAgent = jsonFile[packet]['ethernet']['ipv4']['tcp']['http']['header']['User-Agent']
        self.ua.append(self.userAgent)
        data.update({'srcIp': self.srcIp})
        data.update({'destIp': self.destIp})
        data.update({'srcPort': self.srcPort})
        data.update({'destPort': self.destPort})
        data.update({'ack': self.ack})
        data.update({'uri': "http://"+self.host+self.uri})
        data.update({'method': self.method})
        data.update({'requestType': 'Request'})
        serverResponse = self.findCorrectPacketResponse(jsonFile,self.ack)
        data.update({'Response': serverResponse })
```

FIGURE 25 – Script d'extraction des valeurs depuis le fichier JSON.

Pour former une ligne de log correspondante à ce paquet, voici la partie essentiel du code qu'on a utilisé :

```
if hasattr(self,'res'):
    print("response from malformed packet")
logLine = "{} - - [{}:{}] \"{} {} HTTP/1.1\" {} {} \"-\" \"{}\" {}\n".format(self.srcIp,self.date
print(logLine)
self.writeLog(logLine)
```

FIGURE 26 – Script pour former nos lignes de logs.

Annexe D

Stockage des logs générés sous ElasticSearch

Pour ajouter des données dans elasticsearch chaque ligne de log doit être précédé par une une ligne de code de ce type :

{ "create" : { "_index" : "Logs", "_type" : "attaque", "_id" : 0} }

Ci dessous le script python qui transforme le fichier log en un fichier JSON afin de le préparer pour son importation dans elasticsearch :

```
data = open('logs.txt','r')
data_new = open('logs_elasticsearch.json','w')
data_ligne = data.readlines()
e = 0
for i in data_ligne:
    data_new.write('{ "create" : { "_index" : "Logs", "_type" : "attaque", "_id": '+ str(e) +'} }\n')
    data_new.write(i)
    e += 1
data.close()
data_new.close()
```

FIGURE 27 – Script de transformation en fichier JSON avant l'importation dans elasticsearch.

A la fin de l'exécution du code nos données seront sous le format suivant :

```
{ "create" : { "_index" : "Logs", "_type" : "attaque", "_id": 0} }
192.168.1.41 - - [2023-03-26:23:01:12] "GET /app7/DVWA/vulnerabilities/sqli/?id=&Submit=Submit&user_token=
984690d3eda602d415d947cf5c8c8c66 HTTP/1.1" 200 1370 - Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15
(KHTML, like Gecko) Version/16.1 Safari/605.1.15 (sql injection)
{ "create" : { "_index" : "Logs", "_type" : "attaque", "_id": 1} }
192.168.1.41 - - [2023-03-26:23:01:16] "GET /app7/DVWA/vulnerabilities/sqli/?id=&Submit=Submit&user_token=
984690d3eda602d415d947cf5c8c8c66 HTTP/1.1" 200 1372 - Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/109.0.0.0 Safari/537.36 (sql injection)
{ "create" : { "_index" : "Logs", "_type" : "attaque", "_id": 2} }
192.168.1.41 - - [2023-03-26:23:04:52] "GET /app7/DVWA/vulnerabilities/sqli/?id=&Submit=Submit&user_token=
984690d3eda602d415d947cf5c8c8c66 HTTP/1.1" 200 1370 - sqlmap/1.6.4#stable (https://sqlmap.org) (sql injection)
{ "create" : { "_index" : "Logs", "_type" : "attaque", "_id": 3} }
192.168.1.41 - - [2023-03-26:23:04:54] "POST /app7/DVWA/vulnerabilities/sqli/?id=&Submit=Submit&user_token=
984690d3eda602d415d947cf5c8c8c66 HTTP/1.1" 200 1370 - Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/108.0.0.0 Safari/537.36 (sql injection)
{ "create" : { "_index" : "Logs", "_type" : "attaque", "_id": 4} }
192.168.1.41 - - [2023-03-26:23:04:54] "GET /app7/DVWA/vulnerabilities/sqli/?id=&Submit=Submit&user_token=
984690d3eda602d415d947cf5c8c8c66 HTTP/1.1" 200 1369 - Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/109.0.0.0 Safari/537.36 (sql injection)
{ "create" : { "_index" : "Logs", "_type" : "attaque", "_id": 5} }
192.168.1.41 - - [2023-03-26:23:04:57] "GET /app7/DVWA/vulnerabilities/sqli/?id=4211&Submit=Submit&user_token=
984690d3eda602d415d947cf5c8c8c66 HTTP/1.1" 200 1368 - Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/109.0.0.0 Safari/537.36 (sql injection)
```

FIGURE 28 – Ajout de l'entête pour charger les données sur elasticsearch.

Une fois que nos données sont prêtes, nous pouvons les intégrer à Elasticsearch en utilisant une opération de "bulk", on doit intégrer le fichier JSON vers notre index pour pouvoir stocker nos logs avec un script python :

```python
import requests
url = "http://localhost:9200/_bulk"
header = {"Content-Type":"application/json"}
data = open("logs_elasticsearch.json", "rb")
data = data.read()
response = requests.post(url,headers=header,data=data)
print(response.status_code)
print(response.content)
```

FIGURE 29 – Script de chargerment des données dans ElasticSearch.

La figure qui suit montre un aperçu de nos lignes de log et comment elles sont stockées sur Elasticsearch :

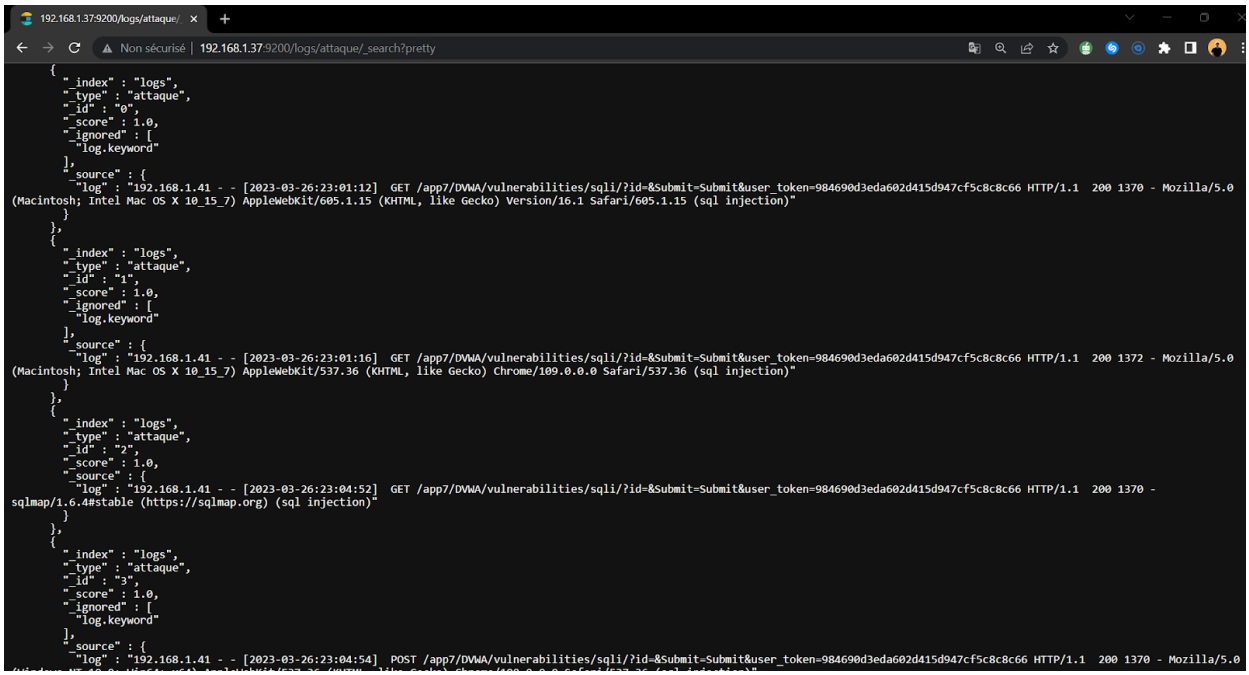

FIGURE 30 – Aperçu des logs chargés sur elasticsearch.

Bibliographie

[1] SecurityAffairs.Co. Croatian Phone Carrier A1 Hrvatska Has Disclosed a Data Breach that Has Impacted Roughly 200,000 Customers ; SecurityAffairs.Co. : New York, NY, USA, 2022

[2] "CVE, Common Vulnerabilities and Exposures", [En ligne], Consulter le : 10/05/2023. Disponible : `https://www.cve.org/`

[3] "NVD, National Vulnerability Database", [En ligne], Consulter le : 10/05/2023. Disponible : `https://nvd.nist.gov/`

[4] "VERACODE, Vulnerability Penetration testing", [En ligne], Consulter le : 10/05/2023. Disponible : `https://www.veracode.com/`

[5] "OWASP", [En ligne], Consulter le : 10/05/2023. Disponible : `https://owasp.org/`

[6] "WASC, Web Application Security", [En ligne], Consulter le : 10/05/2023. Disponible : `http://www.webappsec.org/`

[7] "OWASP, OWASP Top Ten", [En ligne], Consulter le : 10/05/2023. Disponible : `https://owasp.org/www-project-top-ten/`

[8] Prandl, S., Lazarescu, M., Pham, D. S. (2015). A study of web application firewall solutions. In Information Systems Security : 11th International Conference, ICISS 2015, Kolkata, India, December 16-20, 2015. Proceedings 11 (pp. 501-510). Springer International Publishing.

[9] Sabahi, F., Movaghar, A. (2008, October). Intrusion detection : A survey. In 2008 Third International Conference on Systems and Networks Communications (pp. 23-26). IEEE.

[10] "Constantin, L. What are vulnerability scanners and how do they work ?", [En ligne], Consulter le : 11/05/2023. Disponible : `https://www.csoonline.com/article/3537230/what-are-vulnerability-scanners-and-how-do-they-work.html`

[11] Zhao, C., Si, S., Tu, T., Shi, Y., Qin, S. (2022). Deep-Learning Based Injection Attacks Detection Method for HTTP. Mathematics, 10(16), 2914.

[12] Chuvakin, A., Schmidt, K., Phillips, C. (2012). Logging and log management : the authoritative guide to understanding the concepts surrounding logging and log management. Newnes.

[13] "Sematext, What is log management ?", [En ligne], Consulter le : 12/05/2023. Disponible : https://sematext.com/guides/log-management/

[14] "Crowdstrike, 6 Common Log File Formats", [En ligne], Consulter le : 12/05/2023. Disponible : https://www.crowdstrike.com/cybersecurity-101/observability/log-file-formats/

[15] "Microsoft, IIS Logging", [En ligne], Consulter le : 12/05/2023. Disponible : https://learn.microsoft.com/en-us/windows/win32/http/iis-logging

[16] Bensefia, H. (2005). Fichiers logs : preuves judiciaires et composant vital pour forensics. Revue d'Information Scientifique et Technique, 15(1-2).

[17] Li, W. (2013). Automatic log analysis using machine learning : awesome automatic log analysis version 2.0.

[18] "Medium, Introduction to Machine Learning", [En ligne], Consulter le : 14/05/2023. Disponible : https://medium.com/analytics-vidhya/introduction-to-machine-learning-e1b9c055039c

[19] "CNIL, Commission nationale de l'informatique et des libertés", [En ligne], Consulter le : 10/05/2023. Disponible : https://cnil.fr/fr/definition/apprentissage-automatique

[20] Bouchard, G. (2005). Les modèles génératifs en classification supervisée et applications à la catégorisation d'images et à la fiabilité industrielle (Doctoral dissertation, Université Joseph-Fourier-Grenoble I).

[21] "datafranca, Apprentissage semi-supervisé", [En ligne], Consulter le : 14/05/2023. Disponible : https://datafranca.org/wiki/Apprentissage_semi-supervis%C3%A9

[22] "talend, Tout savoir sur le machine learning", [En ligne], Consulter le : 15/05/2023. Disponible : https://www.talend.com/fr/resources/what-is-machine-learning

[23] "Menzli,A. A Machine Learning Approach to Log Analytics : How to Analyze Logs ?", [En ligne], Consulter le : 15/05/2023. Disponible : https://neptune.ai/blog/machine-learning-approach-to-log-analyticsg

[24] Bensefia, H. (2005). Fichiers logs : preuves judiciaires et composant vital pour forensics. Revue d'Information Scientifique et Technique, 15(1-2).

[25] Wirehed, A., Suhren Gustafsson, A. (2021). Log Classification using NLP Techniques Data-Driven Fault Categorization of Multimodal Logs using Natural Language Processing Techniques.

[26] Betarte, G., Pardo, Á., Martínez, R. (2018, December). Web application attacks detection using machine learning techniques. In 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 1065-1072). IEEE.

[27] Eunaicy, J. C., Suguna, S. (2022). Web attack detection using deep learning models. Materials Today : Proceedings, 62, 4806-4813.

[28] "IBM, Qu'est-ce que l'étiquetage des données ?" [En ligne], Consulter le : 29/05/2023. Disponible : https://www.ibm.com/fr-fr/topics/data-labeling

[29] Saleem, S., Sheeraz, M., Hanif, M., Farooq, U. (2020, October). Web server attack detection using machine learning. In 2020 International Conference on Cyber Warfare and Security (ICCWS) (pp. 1-7). IEEE.

[30] Hoang, X. D. (2021). Detecting common web attacks based on machine learning using web log. In Advances in Engineering Research and Application : Proceedings of the International Conference on Engineering Research and Applications, ICERA 2020 (pp. 311-318). Springer International Publishing.

[31] Gao, Y., Ma, Y., Li, D. (2017, October). Anomaly detection of malicious users' behaviors for web applications based on web logs. In 2017 IEEE 17th International Conference on Communication Technology (ICCT) (pp. 1352-1355). IEEE.

[32] Mandagondi, L. G. (2021). Anomaly detection in log files using machine learning techniques.

[33] Björnerud, P. (2021). Anomaly Detection in Log Files Using Machine Learning.

[34] "Cloudflare, Global network designed to make everything you connect to the Internet secure", [En ligne], Consulter le : 15/05/2023. Disponible : https://www.cloudflare.com/fr-fr/learning/ddos/glossary/web-application-firewall-waf/

[35] "Kaspersky, Qu'est-ce qu'une injection SQL ? Définition et explication", [En ligne], Consulter le : 29/05/2023. Disponible : https://www.kaspersky.fr/resource-center/definitions/sql-injection

[36] "OWASP, Blind SQL Injection", [En ligne], Consulter le : 29/05/2023. Disponible : https://owasp.org/www-community/attacks/Blind_SQL_Injection

[37] "HTTPCS,Faille File Inclusion (LFI / RFI)", [En ligne], Consulter le : 29/05/2023. Disponible : https://www.httpcs.com/fr/faille-file-inclusion

[38] "OWASP, Cross Site Scripting(XSS)", [En ligne], Consulter le : 29/05/2023. Disponible : https://owasp.org/www-community/attacks/xss/

[39] "OWASP, Command Injection", [En ligne], Consulter le : 29/05/2023. Disponible : https://owasp.org/www-community/attacks/Command_Injection/

[40] "lirmm, Analyzing Web Traffic ECML/PKDD 2007 discovery challenge", Consulter le : 20/05/2023. Disponible : https://www.lirmm.fr/pkdd2007-challenge/

[41] "ITEFI, HTTP DATASET CSIC 2010", [En ligne], Consulter le : 20/05/2023. Disponible : https://www.tic.itefi.csic.es/dataset/

[42] Moustafa, N., Hu, J., Slay, J. (2019). A holistic review of network anomaly detection systems : A comprehensive survey. Journal of Network and Computer Applications, 128, 33-55.

[43] Schutze, H., Manning, C. D., Raghavan, P. (2008). Introduction to information retrieval. Cambridge University Press.

[44] Yasinsac, A., Manzano, Y. (2001, June). Policies to enhance computer and network forensics. In Proceedings of the 2001 IEEE workshop on information assurance and security (pp. 289-295).

[45] Rajaraman, J. L. A., Ullman, J. D. Data mining (recommended subject) textbook.

[46] Zhao, C., Si, S., Tu, T., Shi, Y., Qin, S. (2022). Deep-Learning Based Injection Attacks Detection Method for HTTP. Mathematics, 10(16), 2914.

[47] Laskov, P., Düssel, P., Schäfer, C., Rieck, K. (2005). Learning intrusion detection : supervised or unsupervised ?. In Image Analysis and Processing–ICIAP 2005 : 13th International Conference, Cagliari, Italy, September 6-8, 2005. Proceedings 13 (pp. 50-57). Springer Berlin Heidelberg.

[48] Sokolova, M., Japkowicz, N., Szpakowicz, S. (2006). Beyond accuracy, F-score and ROC : a family of discriminant measures for performance evaluation. In AI 2006 : Advances in Artificial Intelligence : 19th Australian Joint Conference on Artificial Intelligence, Hobart, Australia, December 4-8, 2006. Proceedings 19 (pp. 1015-1021). Springer Berlin Heidelberg.

[49] "Google, Colaboratory FAQ", [En ligne], Consulter le : 01/06/2023. Disponible : https://research.google.com/colaboratory/faq.html?hl=fr

[50] "JDN, PHP (Hypertext Preprocessor) : définition", [En ligne], Consulté le : 27/05/2023. Disponible : https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203597-php-hypertext-preprocessor-definition/

[51] "JDN, Python : définition et utilisation de ce langage informatique" [En ligne]. Consulté le : 27/05/2023. Disponible : https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1445304-python-definition-et-utilisation-de-ce-langage-informatique/

[52] "Opensource.com, An introduction to using tcpdump at the Linux command line" [En ligne]. Consulté le : 27/05/2023. Disponible : https://opensource.com/article/18/10/introduction-tcpdump

[53] "Kali, Tool Documentation : wfuzz Usage Example" [En ligne]. Consulté le : 27/05/2023. Disponible : https://www.kali.org/tools/wfuzz/#:~:text=Wfuzz%20is%20a%20tool%20designed,Password

[54] "FreeCodeCamp, SQL Injection Attacks – How to Use SQLMap to Find Database Vulnerabilities" [En ligne]. Consulté le : 27/05/2023. Disponible : https://www.freecodecamp.org/news/how-to-protect-against-sql-injection-attacks/#:~:text=SQLMap%20is%20a%20tool%20used,is%20vulnerable%20to%20SQL%20injection.

[55] "NumPy", [En ligne]. Consulté le : 27/05/2023. Disponible : https://numpy.org.

[56] "pandas", [En ligne]. Consulté le : 27/05/2023. Disponible : https://pandas.pydata.org/.

[57] "sklearn", [En ligne]. Consulté le : 27/05/2023. Disponible : https://scikit-learn.org.

[58] "python simple, joblib", [En ligne]. Consulté le : 28/05/2023. Disponible : http://www.python-simple.com/python-autres-modules-non-standards/joblib.php

[59] "PyPI, pypcapkit 1.0.2.post5", [En ligne]. Consulté le : 28/05/2023. Disponible : https://pypi.org/project/pypcapkit/

[60] "Github, digininja/DVWA", [En ligne]. Consulté le : 29/05/2023. Disponible : https://github.com/digininja/DVWA

[61] "PyPI, parse 1.19.0", [En ligne]. Consulté le : 28/05/2023. Disponible : https://pypi.org/project/parse/

[62] "TensorFlow, Introduction aux encodeurs automatiques", [En ligne]. Consulté le : 30/05/2023. Disponible : https://www.tensorflow.org/tutorials/generative/autoencoder?hl=fr

[63] Osório, F. S. (1998). INSS : un système hybride neuro-symbolique pour l'apprentissage automatique constructif (Doctoral dissertation, Institut National Polytechnique de Grenoble-INPG).

[64] Zhang, S., Li, X., Zong, M., Zhu, X., Cheng, D. (2017). Learning k for knn classification. ACM Transactions on Intelligent Systems and Technology (TIST), 8(3), 1-19.

[65] N.K. Sangani, H. Zarger, “Machine Learning in Application Security,” Book chapter in "Advances in Security in Computing and Communications", IntechOpen, 2017

[66] Wang, Q. (2022, June). Support Vector Machine Algorithm in Machine Learning. In 2022 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA) (pp. 750-756). IEEE.

[67] Arlia, D., Coppola, M. (2001). Experiments in parallel clustering with DBSCAN. In Euro-Par 2001 Parallel Processing : 7th International Euro-Par Conference Manchester, UK, August 28–31, 2001 Proceedings 7 (pp. 326-331). Springer Berlin Heidelberg.

[68] "Outlier detection with Local Outlier Factor (LOF)", [En ligne], Consulter le : 13/06/2023. Disponible : `https://scikit-learn.org/stable/auto_examples/neighbors/plot_lof_outlier_detection.html`

[69] N.K. Sangani, H. Zarger, “Machine Learning in Application Security,” Book chapter in "Advances in Security in Computing and Communications", IntechOpen, 2017

[70] "What is the k-nearest neighbors algorithm ?", [en ligne], Consulté le 10/06/2023 Disponible : `https://www.ibm.com/topics/knn`

[71] "Intro to autoencoders", [en ligne], Consulté le 13/06/2023 Disponible : `https://www.tensorflow.org/tutorials/generative/autoencoder.`

[72] "Hubspot, Qu’est-ce qu’un fichier log en informatique et à quoi sert-il ?", [En ligne]. Consulté le : 27/06/2023. Disponible : `https://blog.hubspot.fr/marketing/fichier-log`