REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed El-Bachir El-Ibrahimi de Bordj Bou Arreridj
Faculté des Mathématique et de l'Informatique
Département d'Informatique

MÉMOIRE

Présenté en vue de l'obtention du diplôme

Master en Informatique

Spécialité : Ingénierie de l'informatique décisionnelle

Thème

Un Système Intelligent D'analyse Avancée Des Fichiers Journaux (Logs) Web Pour La Découverte Des Cybermenaces

---

Réalisé par :

- Amel LOUDADJI
- Oussama AITEUR

Soutenu le : Samedi 28/06/2025, devant le jury composé de :

Monsieur Salah MAACHE	Enseignant Université de BBA	Président
Monsieur Oussama SENOUCI	Enseignant Université de BBA	Examinateur
Madame Meriem LAIFA	Enseignant Université de BBA	Examinatrice
Madame Hassina BENSEFIA	Enseignant Université de BBA	Encadrant
Madame Lila GHAMRAOUI	Attachée de Recherche au CERIST	Co-encadrant

Promotion 2024/2025

Remerciements

Avant tout, nous rendons grâce à ALLAH, DIEU le Tout-Puissant, pour nous avoir guidés et accordés la force, la patience et la clarté d'esprit nécessaires pour mener à bien ce travail. Sans Sa volonté, rien n'aurait été possible.

Nous tenons à exprimer toute notre reconnaissance à nos encadrantes **Madame Hassina BENSEFIA**, enseignante à l'université de Bordj Bou Arreridj, et **Madame Lila GUEMRAOUI**, attachée de recherche et chef d'équipe de recherche au CERIST pour leur accompagnement scientifique rigoureux, leur disponibilité constante, leur conseils éclairés, leur encouragement et surtout pour leur bienveillance humaine. Cette attention sincère a compté bien plus que des mots.

Nous tenons à adresser nos remerciements chaleureux aux membres du jury : **Monsieur Salah MAACHE**, **Monsieur Oussama SENOUCI** et **Madame Meriem LAIFA**, enseignants et maitres de conférences à l'université de Bordj Bou Arreridj, pour le temps qu'ils ont consacré à l'évaluation de ce mémoire, pour leurs observations pertinentes et leurs remarques constructives, que nous considèrerons comme une source précieuse d'apprentissage.

Nous exprimons nos vifs remerciements à l'honorable institution scientifique **Centre de Recherche sur l'Information Scientifique et Technique** (**CERIST**) pour leur accueil et accompagnement professionnels au sein de la division de recherche Cybersécurité. Nous sommes très particulièrement reconnaissants envers les attachées de recherche : **Madame Sihem BENSIMESSAOUD**, **Madame Amina ZEMMACHE** et **Madame Amina DJELLABIA**, l'équipe merveilleuse très professionnelle qui a accompagné nos encadrantes durant toute la période d'élaboration de ce travail avec leurs orientations stratégiques, conseils structurants, critiques constructives et encouragement constant.

Nous exprimons également notre profonde gratitude à l'ensemble des enseignants qui ont jalonné notre parcours universitaire. À chaque étape, nous avons eu la chance de croiser des enseignants passionnés, exigeants et engagés, qui ont su éveiller notre curiosité intellectuelle et nous aider à construire des bases solides. Une pensée particulière à **Monsieur le Professeur Rahmoune Azzedine**, ex. Doyen de la faculté MI de l'université de Bordj Bou Arreridj, pour sa vision et son engagement envers la qualité de notre formation.

Enfin, nous remercions toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce mémoire, par un mot, un conseil, un soutien moral ou une présence discrète : **merci du fond du cœur**.

Dédicace Oussama

À mes parents, Noreddine et Fatima,

Pour tout ce que vous êtes, et tout ce que vous m'avez transmis. Votre patience, vos sacrifices silencieux, votre éducation exigeante mais juste, ont forgé l'homme que je suis devenu. Vous avez toujours cru en moi, même lorsque moi je doutais. Je vous dois cette réussite, et bien plus encore.

À mon frère Imad,

Pour ta présence constante, rassurante et naturelle. Tu as toujours su trouver les bons mots quand il le fallait, avec calme et sincérité. Ta manière d'être là, toujours au bon moment, m'a profondément marqué. Tu n'es pas toujours dans l'expression, mais ta présence a toujours été là, et je l'ai toujours sentie. Tu es et restes un repère précieux dans ma vie.

À mon frère Dhirar,

Pour ton soutien immédiat, ta générosité naturelle et ta capacité à être disponible sans qu'on ait besoin de demander. Tu es de ceux sur qui on peut réellement compter. Ton implication dans ma réussite est réelle et profonde.

Peu de personnes ont la chance de croiser leurs idoles. Moi, j'ai eu la chance de grandir avec les miens. Vous étiez là, devant moi, éclairant mon chemin, chacun à votre manière.

À ma sœur Manel,

Pour ta sensibilité vive, ton attention aux détails, ta manière de toujours chercher à réconforter, même à distance. Tu es un appui émotionnel précieux, toujours présente avec douceur, sans jamais forcer.

À ma sœur Narimane,

Pour ton esprit posé, ta capacité à rester calme et lucide, et ton affection discrète mais constante. Tu fais partie de ces personnes qui donnent sans bruit, mais avec un réel impact. Ta manière de t'inquiéter sans le dire m'a toujours touché.

À mes belles-sœurs, Amel et Lydia,

Pour votre présence respectueuse, votre douceur naturelle et vos gestes pleins d'attention. Vous avez toujours su m'offrir une affection simple et sincère, sans jamais chercher à en faire trop, et cela a beaucoup compté pour moi.

À mes beaux-frères, Riad et Adel,

Pour votre fraternité sincère, votre esprit positif et votre simplicité dans les échanges. Vous avez toujours su créer un climat de confiance et de respect mutuel, et c'est quelque chose que j'apprécie profondément.

À mes neveux et nièces : Mahdi, Massine, Iyad, Élyne et Dina,

Vous êtes une partie précieuse de ma vie. Votre présence m'apporte une forme d'équilibre que seules les relations sincères peuvent offrir.

À Farouk et Samah,

Vous n'êtes pas simplement des amis. Vous êtes une continuité de ma famille. Dans les moments d'effort, d'échec ou de doute, vous avez été là, sans bruit, mais avec force. Votre fidélité est rare, et je la mesure pleinement.

Dédicace Amel

Au nom d'Allah, Le Tout Miséricordieux, Le Très Miséricordieux.

Louange à Dieu, source de savoir, de force et de patience, qui m'a guidée tout au long de ce parcours difficile et enrichissant.

Je dédie ce travail :

À mon père, pour son amour silencieux, ses sacrifices inestimables et sa confiance en moi qui ne s'est jamais éteinte.

À ma mère, mon pilier, pour ses prières, sa tendresse, sa force et son soutien inconditionnel dans chaque étape de ma vie.

À mes sœurs et à mon frère, pour leur présence, leurs encouragements et l'affection qu'ils m'ont toujours apportée.

À ma superviseure au centre de recherche, pour son encadrement précieux, sa rigueur scientifique et sa bienveillance tout au long de cette aventure.

À toutes ces personnes qui ont cru en moi, même dans les moments où moi-même je doutais.

RÉSUMÉ

La large diffusion des applications web et leur accessibilité universelle constituent un facteur de risque qui multiplie les vecteurs d'attaque potentiels. Malgré la diversité des mécanismes de protection disponibles, des systèmes de surveillance et d'alerte sophistiqués, ainsi qu'une capacité de réponse rapide aux incidents de sécurité sont nécessaires. Les fichiers journaux (logs) d'une application web fournissent des enregistrements détaillés des activités qui se produisent au niveau du serveur web. Leur analyse offre une surveillance constante permettant de détecter et de comprendre l'évolution des menaces, tout en garantissant une réaction rapide aux incidents de sécurité. L'intelligence artificielle, en particulier l'apprentissage automatique, est devenue une solution efficace pour l'analyse avancée des fichiers journaux web et ce pour améliorer significativement la capacité de détection des attaques web. Ce mémoire propose l'architecture d'un système intelligent d'analyse avancée des fichiers journaux web. Les composants clés de cette architecture sont le moteur d'analyse et le module de décision. Le moteur d'analyse utilise une approche hybride basée sur les techniques de machine d'apprentissage non supervisées et supervisées. Il consiste en ensemble de classifieurs diversifiés qui identifie les enregistrements journaux suspects puis fait leur classification détaillée en attaques. Le module de décision fusionne les résultats issus des classifieurs selon une heuristique proposée et présentent les résultats de classifications finaux. Ce qui donne un aspect scientifique à la classification des attaques et ajoute un deuxième caractère d'intelligence au système proposé. L'évaluation des performances de notre système montre des résultats très satisfaisants qui prouvent l'efficacité de notre système d'analyse, confirme notre orientation vers une approche hybride des techniques de machine d'apprentissage et montre la démarche scientifique et intelligente adoptée lors de l'étape de prise de décision pour la classification finale des résultats.

Mots-clés : Journaux web (logs web) ; apprentissage automatique ; machine d'apprentissage ; classification ; Attaque Web ; Prise de Décision.

Abstract

The widespread use of web applications and their universal accessibility constitute a risk factor that multiplies potential attack vectors. Despite the variety of available protection mechanisms, sophisticated monitoring and alert systems, as well as a rapid response capability to security incidents, are essential. Web application log files provide detailed records of activities occurring at the web server level. Their analysis enables continuous monitoring, allowing the detection and understanding of threat evolution while ensuring a quick response to security incidents. Artificial intelligence, particularly machine learning, has become an effective solution for advanced analysis of web log files, significantly improving the ability to detect web attacks. This thesis proposes the architecture of an intelligent system for advanced web log analysis. The key components of this architecture are the analysis engine and the decision module. The analysis engine uses a hybrid approach based on both unsupervised and supervised machine learning techniques. It consists of a set of diverse classifiers that first identify suspicious log records and then perform a detailed classification of the attacks. The decision module merges the outputs of the classifiers using a proposed heuristic and presents the final classification results. This gives a scientific dimension to attack classification and adds a second layer of intelligence to the proposed system. The performance evaluation of our system shows highly satisfactory results, proving its efficiency, confirming our choice of a hybrid machine learning approach, and demonstrating the scientific and intelligent methodology adopted in the final decision-making step.

Keywords: Web logs; Machine Learning; Classification; Web Attack; Decision Making ;Automatique Learning

الملخص

إن الانتشار الواسع للتطبيقات الويب وإمكانية الوصول إليها بشكل عالمي يُعتبر عامل خطر يزيد من احتمالات وجود نواقل للهجمات. ورغم تنوع آليات الحماية المتاحة، إلا أن الحاجة تبقى ملحّة إلى أنظمة مراقبة وتنبيه متطورة، بالإضافة إلى قدرة سريعة على الاستجابة لحوادث الأمان. توفّر ملفات السجلات (Logs) الخاصة بتطبيقات الويب تسجيلات مفصلة للأنشطة التي تحدث على مستوى خادم الويب. ويُتيح تحليلها مراقبة دائمة تساعد في اكتشاف وتتبع تطور التهديدات، مما يضمن استجابة سريعة وفعالة. وقد أصبحت تقنيات الذكاء الاصطناعي، ولا سيما التعلم الآلي، حلاً فعالاً في تحليل ملفات السجلات بشكل متقدم، مما يعزز بشكل كبير من قدرة اكتشاف الهجمات على تطبيقات الويب. يقترح هذا البحث معمارية لنظام ذكي للتحليل المتقدم لملفات سجلات الويب. وتتمثل المكونات الرئيسية لهذا النظام في محرّك التحليل ووحدة اتخاذ القرار. يستخدم محرّك التحليل نهجًا هجينًا يجمع بين تقنيات التعلم الآلي غير المُراقب والمُراقب، ويتكوّن من مجموعة من المصنّفات المتنوعة التي تقوم بتحديد السجلات المشبوهة أولاً، ثم تصنيفها بدقة حسب نوع الهجوم. أما وحدة اتخاذ القرار، فتقوم بدمج نتائج المصنّفات اعتمادًا على خوارزمية اقترحناها، وتعرض نتائج التصنيف النهائية. مما يُضفي طابعًا علميًا على تصنيف الهجمات، ويمنح النظام طابعًا ثانيًا من الذكاء. تُظهر نتائج تقييم أداء النظام فعاليته العالية، وتُؤكد جدوى استخدام المقاربة الهجينة لتقنيات التعلم الآلي، كما تُبرز النهج العلمي والذكي المُتبع في مرحلة اتخاذ القرار النهائي.

الكلمات المفتاحية :سجلات الويب؛ التعلم الآلي؛ الة التعلم؛ التصنيف؛ هجمات الويب؛ اتخاذ القرار.

Table Des Matières

Liste des figures

Liste des tableaux

INTRODUCTION GENERALE

CONTEXTE

La transformation numérique rapide et l'adoption généralisée des applications web ont considérablement accru la surface d'attaque des systèmes d'information, exposant les organisations à un éventail de cyber-menaces de plus en plus sophistiquées et persistantes.

Historiquement, la cybersécurité s'est appuyée sur des mécanismes de défense périphériques tels que les pare-feu (firewalls), les systèmes de détection d'intrusion (IDS) et des règles statiques. Bien que fondamentales, ces approches s'avèrent de plus en plus insuffisantes face à l'augmentation exponentielle du volume et de la complexité des attaques modernes.

Dans ce contexte, les fichiers log web, qui enregistrent chaque interaction avec le serveur, représentent une source de données inestimable. Ils offrent une traçabilité granulaire des activités et des incidents potentiels, constituant ainsi une ressource critique pour la détection proactive et réactive des menaces.

PROBLÉMATIQUE

Malgré leur richesse informationnelle, l'exploitation optimale des fichiers log web pour la cybersécurité reste souvent sous-utilisée ou limitée par les dispositifs classiques. L'analyse manuelle, ou celle basée sur des règles prédéfinies, atteint rapidement ses limites face à la volumétrie massive des données générées quotidiennement et à la diversité évolutive des menaces. La capacité à identifier des schémas d'attaque subtils, des anomalies "zero-day" ou des comportements malveillants complexes est gravement compromise par ces approches statiques, créant ainsi un écart significatif entre la quantité d'informations disponibles dans les logs et leur utilisation effective pour la détection des menaces.

Dans ce contexte, le potentiel avéré des techniques d'intelligence artificielle (IA) et d'apprentissage automatique (Apprentissage automatique) offre une voie prometteuse pour surmonter ces limitations en automatisant et en affinant la détection d'anomalies et d'attaques. Il devient alors crucial de déterminer la manière dont ces techniques peuvent être

mises à profit pour développer une approche optimale d'analyse avancée des fichiers log web. Une telle approche doit non seulement renforcer significativement la détection des incidents de sécurité et réduire efficacement les faux positifs, mais également assurer une synergie complémentaire avec les dispositifs de cybersécurité traditionnels, garantissant ainsi une sécurité globale plus robuste et adaptative.

OBJECTIFS

Ce mémoire vise à adresser la problématique susmentionnée à travers les objectifs spécifiques suivants :

- Valoriser et optimiser l'exploitation des fichiers log web en tant que source de données stratégique pour la détection précoce des comportements suspects et des activités malveillantes, en agissant comme un complément aux dispositifs de sécurité classiques plutôt qu'un remplacement.
- Intégrer et adapter des techniques d'intelligence artificielle et d'apprentissage automatique déjà établies pour automatiser et affiner l'analyse des logs, afin d'accroître la capacité de détection et la réactivité face aux menaces émergentes et sophistiquées.
- Minimiser le taux de faux positifs dans les alertes générées. Cet objectif est crucial pour maintenir la pertinence opérationnelle de détection des attaques, réduire la surcharge des équipes de sécurité et garantir une allocation efficace des ressources d'investigation.

ORGANISATION DU MÉMOIRE

Ce mémoire est structuré en deux parties principales :

Partie I : elle englobe l'état de l'art qui est réparti en trois chapitres

- **Chapitre 1** : intitulé **sécurité des applications web**, il présente les problèmes de sécurité des applications web et les stratégies de protection existantes. A la fin, il met l'accent sur le besoin d'analyse des fichiers journaux web comme une stratégie de protection prometteuse.

- I **Chapitre 2** : intitulé ‘’***Analyse des fichiers log web*“,** il introduit les différents types de fichiers log web, leurs formats, leur structure et leur importance fondamentale dans la surveillance des systèmes et la détection des incidents de sécurité.

- **Chapitre 3 :** Titré ‘**Etude des Approches Existantes d’Analyse des Journaux Web Basées sur le ML**’’, au début, il présente un état de l'art des techniques d'intelligence artificielle et d'apprentissage automatique puis il converge vers une étude approfondie des approches existantes d'analyse des logs web employant les techniques ML et l’apprentissage automatique.

Partie II : elle est consacrée à la contribution apportée et elle deux chapitres.

- **Chapitre 4 :** Il porte le titre ‘’ ***Conception du système d’analyse avancée des fichiers log web*** ‘’. Il aborde en détail l'architecture système d'analyse proposé en montrant l’approche hybride d’analyse basée sur les méthodes de apprentissage automatique et le module de décision aussi intelligent qui fusionne des résultats de l’analyse.
- **Chapitre 5 :** intitulé ‘’ ***Implémentation et Evaluation des performances*** ‘’, il présente les choix techniques réalisés pour l'implémentation du système, les jeux de données utilisés pour l'expérimentation, et une analyse rigoureuse des résultats obtenus, mettant en évidence les performances en termes de détection et classification des attaques et de minimisation des faux positifs.

Enfin, ce mémoire s’achève par une conclusion générale qui présente une synthèse des contributions apportées, une discussion des limites de l'approche proposée et des pistes pour de futures recherches.

CHAPITRE 1 : SECURITE DES APPLICATIONS WEB

I.1. INTRODUCTION

À l'ère numérique, les applications web sont devenues des piliers des échanges économiques et sociaux. Leur accessibilité mondiale, combinée à la sensibilité des données qu'elles manipulent (informations personnelles, transactions financières, etc.), représentent des cibles pour les attaquants. Selon un rapport IBM de 2023, le coût moyen d'une violation de données atteint 4,45 millions de dollars, soulignant ainsi l'impératif d'une sécurisation rigoureuse.

Les menaces telles que les injections SQL, les attaques XSS et les attaques DDoS peuvent engendrer des conséquences désastreuses : perte de données critiques, paralysie des opérations, ou atteinte durable à la réputation.

Ce chapitre examine les défis liés à la sécurité des applications web. Il analyse les vecteurs d'attaque courants et présente les stratégies de protection modernes, en mettant l'accent sur l'importance d'une analyse proactive pour anticiper et atténuer les risques.

I.2.INTRODUCTION AUX APPLICATIONS WEB

I.2.1. Définition

Une application web est un logiciel applicatif accessible via un réseau (Internet ou intranet) à l'aide d'un navigateur web. Contrairement aux applications traditionnelles qui sont installées directement sur un appareil, les applications web sont hébergées sur un serveur distant et accessibles depuis n'importe quel appareil disposant d'une connexion réseau et d'un navigateur [3].

I.2.2. Architecture De Base

L'architecture d'une application web repose généralement sur un modèle client-serveur [3], comme le présente la figure 1.1 :

- **Client (front end) :** Le navigateur web de l'utilisateur, qui envoie des requêtes au serveur et affiche les réponses.
- **Serveur (back end) :** Un hôte distant qui héberge l'application web et traite les requêtes des clients. Il est composé de trois éléments principaux :
    - **Serveur web :** Reçoit les requêtes des clients et renvoie les réponses (pages web, données, etc.).
    - **Serveur d'applications :** Exécute la logique métier de l'application et interagit avec la base de données.
    - **Base de données :** Stocke les données de l'application.

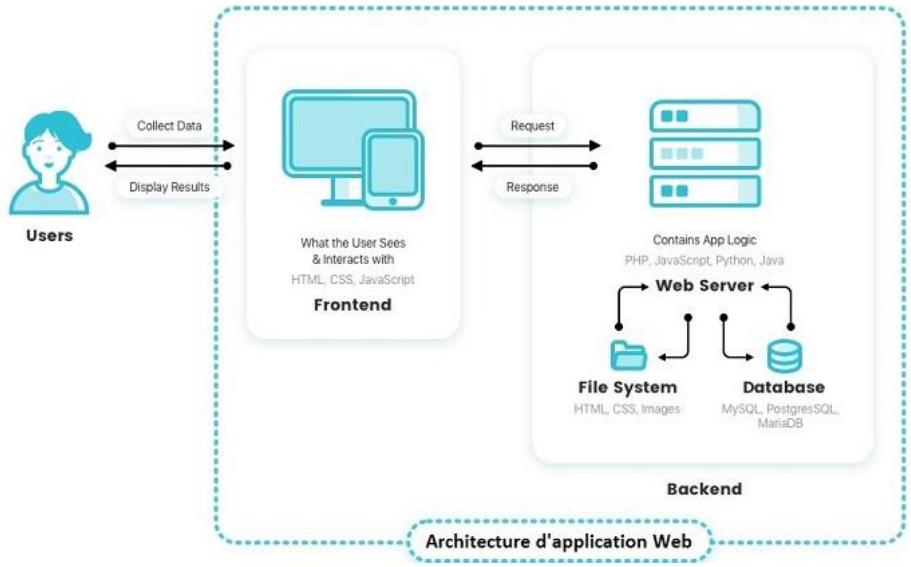


Figure I.1 : Architecture générale d'une application web

I.2.3. Types D'applications Web

Il existe de nombreux types d'applications web, classés en fonction de leurs fonctionnalités et de leur complexité [3] :

- **Applications web statiques :** elles sont composées de pages web HTML, CSS et JavaScript qui ne nécessitent pas de traitement côté serveur.

- **Applications web dynamiques :** elles génèrent du contenu personnalisé en fonction des requêtes des utilisateurs, en utilisant des langages de programmation côté serveur (PHP, Python, etc.) et des bases de données.
- **Applications mono-pages (Single Page Applications, SPA) :** elles chargent une seule page HTML et mettent à jour dynamiquement le contenu via JavaScript, offrant une expérience utilisateur fluide et réactive.
- **Applications web progressives (Progressive Web Applications, PWA) :** Ce sont des applications web qui utilisent les fonctionnalités modernes des navigateurs pour offrir une expérience similaire à celle des applications natives (fonctionnement hors ligne, notifications push, etc.).

I.2.4. Avantages Des Applications Web

- **Accessibilité :** elles sont accessibles depuis n'importe quel appareil disposant d'un navigateur et d'une connexion réseau.
- **Mises à jour centralisées :** Les mises à jour sont déployées sur le serveur, sans nécessité de mise à jour côté client.
- **Compatibilité multiplateforme :** elles fonctionnent sur tous les systèmes d'exploitation et appareils disposant d'un navigateur.
- **Coûts de développement réduits :** Un seul code base pour toutes les plateformes.

I.2.5. Limitations Des Applications Web

- **Dépendance à la connexion réseau : elles** nécessitent une connexion Internet pour fonctionner.
- **Performances :** elles peuvent être moins performantes que les applications natives dans certains cas.
- **Sécurité :** elles sont plus vulnérables aux attaques en raison de leur accessibilité via le réseau.

I.2.6 Facteurs De Risque Pour Les Applications Web

’ La diversité des applications web : une complexité synonyme de vulnérabilités’’

L'écosystème des applications web se caractérise par une grande variété d'architectures et de technologies. Cette hétérogénéité, bien que source d'innovation, engendre une complexité qui se traduit par des défis de sécurité significatifs. Les vulnérabilités potentielles se manifestent à différents niveaux :

- **Côté client :** Les failles de type Cross-Site Scripting (XSS) et Cross-Site Request Forgery (CSRF) représentent des menaces majeures.
- **Côté serveur :** Les injections SQL, les failles d'authentification et la gestion défaillante des sessions sont aussi de points faibles à exploiter.
- **Au niveau du réseau : M**an-in-**T**he-**M**iddle (MTM) compromettent la confidentialité, l’intégrité et l’authenticité des échanges.

De plus, la large diffusion des applications web les rend accessibles à un public étendu, ce qui accroît leur exposition aux attaques. Cette accessibilité universelle constitue un facteur de risque supplémentaire, car elle multiplie les vecteurs d'attaque potentiels.

La section suivante de ce chapitre explorera en détail les enjeux de sécurité spécifiques aux applications web, en mettant en lumière les mécanismes d'attaque et les contre-mesures appropriées.

I. 3. Cybersécurité : concepts fondamentaux

I. 3.1. DEFINITION DE LA CYBER-SECURITE

La cyber-sécurité est un ensemble de techniques, de pratiques et de mesures visant à protéger les actifs numériques d'une organisation, notamment les données, les logiciels, le matériel et les personnes. Elle dépasse la simple protection technique en intégrant des dimensions organisationnelles, juridiques et humaines. La cyber-sécurité englobe l'élaboration de politiques de sécurité, la sensibilisation des employés et la conformité aux réglementations en vigueur [1].

I. 3.2. Propriétés de la cybersécurité

Confidentialité : elle garantit r que seules les personnes autorisées ont accès aux informations sensibles. Cela nécessite des mécanismes d'authentification robustes et des contrôles d'accès appropriés.

- **Intégrité :** elle garantit que les données ne sont pas modifiées sans autorisation. Des techniques telles que les hachages et les signatures numériques sont utilisées pour vérifier l'intégrité des données.
- **Disponibilité :** elle garantit un accès continu et fiable aux données et aux ressources pour les utilisateurs autorisés. Cela implique la mise en place de solutions de redondance et de reprise après sinistre.
- **Authenticité :** elle permet de vérifier l'identité de la source d'une action. L'utilisation de certificats numériques et de l'authentification multifactorielle contribue à établir cette authenticité.

  Non-répudiation : Elle empêche la répudiation qui est le fait de nier avoir participé à une communication (échange de données : envoi ou réception) et ce grâce au mécanisme de signatures numériques.

I. 3.3. Concepts de base en cybersécurité

I. 3.3.1. Vulnérabilité

Une vulnérabilité est une faiblesse dans un système qui peut être exploitée par une menace [1,2], par exemples :

- **Faiblesses de conception :** Logiciels mal conçus qui ne tiennent pas compte des menaces potentielles.
- **Erreurs de configuration :** Paramètres de sécurité mal configurés qui créent des points d'entrée pour les attaquants.
- **Mots de passe faibles :** Utilisation de mots de passe faciles à deviner ou de mots de passe par défaut.

I.3.3.2. Menace

Une menace est une cause potentielle d'un incident de sécurité. Les menaces peuvent être classées en deux grandes catégories [1 ,2] :

- **Menaces internes :** elles proviennent de personnes au sein de l'organisation, qu'il s'agisse d'actes intentionnels (malveillance) ou accidentels (négligence).
  Exemples : Oubli de sauvegarde, erreurs de manipulation des informations, erreurs de conception des applications.
- **Menaces externes :** elles proviennent de sources extérieures à l'organisation, telles que des hackers ou des groupes cybercriminels.
  Exemples : Pirates informatiques, logiciels malveillants, attaques réseau.

I.3.3.3. Attaque

Une attaque est une action délibérée visant à compromettre la sécurité d'un système. Les attaques peuvent être classées de différentes manières [1] :

- **Ciblées :** Visent des individus ou des organisations spécifiques.
- **Non ciblées :** Vise un large éventail de cibles sans distinction.

- **Passives :** Surveillance des données sans interférer avec le système.
- **Actives :** Tentatives de modification ou de destruction de données.

LES ETAPES d'une ATTAQUE

1. **Reconnaissance :** Collecte d'informations sur la cible.
2. **Exploitation :** Utilisation de vulnérabilités pour accéder au système.
3. **Installation :** Mise en place d'une porte dérobée ou d'un logiciel malveillant.
4. **Maintien d'accès :** Assurer un accès continu au système compromis.
5. **Action :** Exécution d'actions malveillantes, comme le vol de données

✍ **Distinction entre menace et attaque**

La menace est une cause potentielle d'un incident pouvant endommager un système, tandis que l'attaque est l'action concrète qui vise à compromettre la sécurité de ce système en exploitant une vulnérabilité. En d'autres termes, la menace est le risque potentiel, et l'attaque est la réalisation de ce risque par une action délibérée [1].

I.3.3.4. Intrusion

Est une introduction par effraction dans un système qui peut compromettre le système en termes de confidentialité, d'intégrité et de disponibilité. C'est une attaque partiellement ou complètement réussie.

I.4. Sécurisation des applications web

I.4.1. Attaques courantes sur les applications web (OWASP top 10, 2021)

Les dix principales attaques, illustrées dans la figure 1.2, recensées par (Open Worldwide Application Security Project, OWASP (www.OWASP.org)) incluent, entre autres [5] :

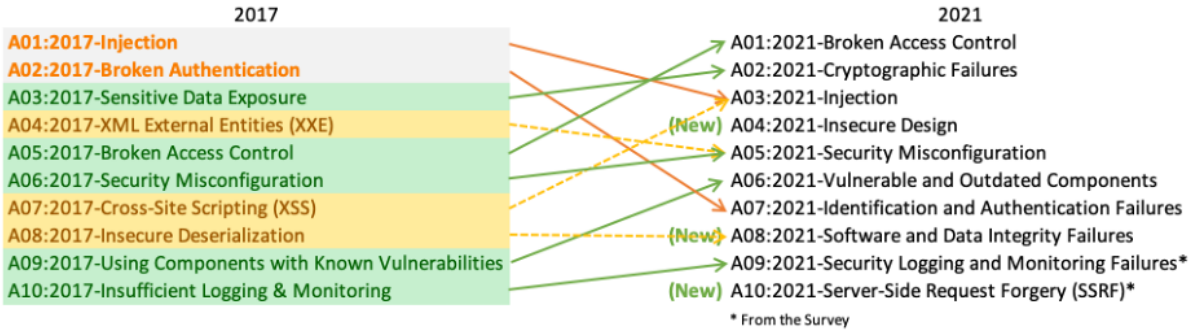


Figure I.2 : Les attaques web les plus fréquentes exploitant des vulnérabilités [5]

- **Contrôle d'accès brisé (Broken Access Control) :** Les restrictions sur ce que les utilisateurs authentifiés sont autorisés à faire sont mal appliquées. Les attaquants peuvent exploiter ces faiblesses pour accéder à des comptes d'autres utilisateurs, consulter des fichiers sensibles, modifier des données et modifier les droits d'accès.
- **Défaillances cryptographiques (Cryptographic Failures) :** Défauts liés au chiffrement des données sensibles, que ce soit au repos ou en transit. Cela peut inclure l'utilisation d'algorithmes de chiffrement faibles, un stockage incorrect des clés de chiffrement ou l'absence de chiffrement des données sensibles.
- **Injection (Injection) :** Les failles d'injection, telles que l'injection SQL, NoSQL, OS et LDAP, se produisent lorsque des données non fiables sont envoyées à un interpréteur

sous forme de commande ou de requête. L'attaquant peut ainsi exécuter des commandes inattendues ou accéder à des données sans autorisation appropriée.

- **Conception non sécurisée (Insecure Design) :** Catégorie large représentant les défauts de conception au niveau de l'architecture et du design de l'application, menant à des vulnérabilités. Nécessite une approche "security by design".
- **Mauvaise configuration de sécurité (Security Misconfiguration) :** Souvent le résultat de configurations par défaut non sécurisées, de configurations incomplètes, d'accès non contrôlé au stockage cloud, d'en-têtes HTTP mal configurés et de messages d'erreur détaillés contenant des informations sensibles.
- **Composants vulnérables et obsolètes (Vulnerable and Outdated Components) :** Les applications et les API utilisent souvent des bibliothèques, des frameworks et d'autres modules logiciels. Si ces composants sont vulnérables, l'application peut être compromise.
- **Défaillances d'identification et d'authentification (Identification and Authentication Failures) :** Liées à la mauvaise gestion des fonctions d'authentification et de gestion de session, permettant aux attaquants de compromettre les mots de passe, les clés ou les jetons de session, ou d'exploiter d'autres failles d'implémentation pour assumer l'identité d'autres utilisateurs.
- **Défaillances d'intégrité des données et des logiciels (Data and Software Integrity Failures) :** Cette catégorie englobe le code et l'infrastructure qui ne protègent pas contre les violations d'intégrité. En exemple, l'utilisation de plugins, de bibliothèques ou de modules provenant de sources non fiables.
- **Défaillances de journalisation et de surveillance de la sécurité (Security Logging and Monitoring Failures) :** Une journalisation et une surveillance insuffisantes permettent aux attaquants de rester inaperçus dans les systèmes, d'accéder à davantage de systèmes et de falsifier, d'extraire ou de détruire des données.
- **Falsification de requête côté serveur (Server-Side Request Forgery - SSRF) :** Les vulnérabilités SSRF se produisent lorsqu'une application web récupère une ressource distante sans valider l'URL fournie par l'utilisateur. Cela permet à un attaquant de forcer le serveur à envoyer des requêtes vers des destinations inattendues, même protégées par un pare-feu.

✍ Une description plus détaillée des attaques mentionnées ci-haut est fournie dans l'annexe 01.

I.4.2. Mécanismes et outils de protection

Pour sécuriser les applications web, plusieurs mécanismes et outils peuvent être déployés [4] :

- **Authentification et gestion de session sécurisées :** Utilisation de l'authentification multifactorielle (MFA) et de jetons sécurisés pour garantir que seuls les utilisateurs autorisés ont accès.
- **Chiffrement :** Mise en œuvre de HTTPS pour sécuriser la transmission des données entre le client et le serveur.
- **Validation et désinfection des entrées :** Assurer que toutes les données entrantes sont validées et nettoyées pour prévenir les injections.
- **Politique de sécurité du contenu (CSP) :** Mise en place de règles qui définissent quelles ressources peuvent être chargées sur une page web.
- **Outils d'analyse statique et dynamique (SAST, DAST) :** Utilisation d'outils pour analyser le code source et les applications en cours d'exécution à la recherche de vulnérabilités.
- **Outils de Linting :** Vérification de la qualité et de la sécurité du code durant le développement.
- **Bibliothèques de sécurité :** Intégration de bibliothèques éprouvées pour gérer les aspects de sécurité courants.
- **Surveillance et réponse aux incidents :** Mise en place de systèmes de surveillance pour détecter et répondre rapidement aux incidents de sécurité.

I.4.3. Limites des mécanismes de protection existants

Malgré la diversité des mécanismes de protection disponibles, plusieurs limites persistent et compromettent l'efficacité globale de la sécurité des applications web, dont nous citons principalement **[3]** :

- **Mises à jour insuffisantes :** Les nouvelles vulnérabilités et les nouveaux risques de sécurité apparaissent constamment, et les mécanismes de défense doivent être mis à jour régulièrement pour maintenir leur efficacité.
- **Erreur humaine :** Les erreurs humaines, telles que des mots de passe faibles, des configurations incorrectes et un manque de sensibilisation à la sécurité, peuvent compromettre l'efficacité des mécanismes de protection, même si ces mécanismes sont bien conçus et mis en œuvre.
- **Complexité technique :** La complexité croissante des applications Web et des infrastructures sous-jacentes rend difficile la conception, la mise en œuvre et la maintenance de mécanismes de protection efficaces.
- **Sécurité en temps réel :** Les mécanismes de protection doivent être capables de fonctionner en temps réel afin de détecter et de bloquer les attaques dès qu'elles se produisent. Cela nécessite des systèmes de surveillance et d'alerte sophistiqués, ainsi qu'une capacité de réponse rapide aux incidents de sécurité.
- **Gestion des vulnérabilités émergentes et des nouvelles attaques :** Les mécanismes de protection doivent être capables de s'adapter aux nouvelles vulnérabilités et aux nouvelles attaques dès qu'elles sont découvertes. Cela nécessite une surveillance continue des menaces, ainsi qu'une capacité de réponse rapide aux incidents de sécurité.

✍ Dans ce contexte, l'importance d'une analyse continue, en temps réel ou non, devient cruciale pour renforcer la posture de sécurité des applications web. Une surveillance constante permet de détecter et de comprendre l'évolution des menaces, tout en garantissant une réaction rapide aux incidents. Cela inclut non seulement l'analyse des alertes de sécurité, mais aussi une évaluation régulière des configurations et des comportements du système.

L'analyse des logs s'inscrit parfaitement dans cette démarche. En surveillant les journaux d'accès et d'erreurs, les organisations peuvent non seulement détecter des activités suspectes, mais aussi :

- **Identifier des tendances et des motifs d'attaques :** Une analyse approfondie des logs permet de dégager des modèles comportementaux des attaquants, offrant ainsi des insights précieux pour anticiper et neutraliser les menaces futures.
- **Améliorer la réponse aux incidents :** En examinant les logs post-incidents, les équipes de sécurité peuvent comprendre ce qui s'est passé, évaluer l'impact et ajuster leurs stratégies de protection en conséquence.
- **Affiner les mécanismes de protection :** L'analyse des logs aide à identifier les points faibles des systèmes de sécurité actuels, permettant ainsi d'apporter des améliorations ciblées et de renforcer la défense globale.

Ainsi, l'intégration de l'analyse continue, notamment à travers l'analyse des logs, permet de combler certaines lacunes des mécanismes de protection existants. Cette approche proactive et réactive est essentielle pour faire face aux menaces en constante évolution. Ce point sera approfondi dans le chapitre suivant, où nous explorerons les méthodologies et les outils d'analyse des logs pour améliorer la sécurité des applications web.

I.5. Synthèse et conclusion

La sécurité des applications web est un processus continu qui nécessite une approche multicouche. Comprendre les vulnérabilités et les attaques, tout en adoptant des mécanismes de protection robustes, est essentiel pour protéger les données et les utilisateurs. Cependant, au-delà des mesures techniques, l'analyse approfondie des journaux (logs) représente un outil crucial dans cette stratégie de défense.

L'analyse des logs permet non seulement de détecter des activités suspectes en temps réel, mais aussi d'identifier les tendances et les motifs d'attaques potentielles. En examinant minutieusement les données enregistrées, les organisations peuvent réagir rapidement aux incidents, mieux comprendre les tentatives d'intrusion et renforcer leurs défenses.

En somme, l'importance d'une analyse approfondie, notamment celle des logs, ne peut être sous-estimée. Elle constitue un pilier fondamental pour anticiper et résoudre les problèmes

de sécurité. Ce point sera abordé plus en détail dans le chapitre suivant, où nous explorerons les méthodologies et les outils d'analyse des logs pour améliorer la sécurité des applications web.

CHAPITRE 2 : ANALYSE DES FICHIERS LOG

II.1. Introduction

Dans l'environnement actuel de transformation numérique rapide, la sécurisation des systèmes d'information est une priorité absolue. Face à la fréquence croissante des cyberattaques et leur sophistication grandissante des menaces, il est crucial de disposer de mécanismes efficaces pour détecter, analyser et répondre aux incidents de sécurité. L'analyse des fichiers journaux joue un rôle central dans cette démarche, en fournissant des enregistrements détaillés des activités des systèmes, des applications et des réseaux.

Les fichiers journaux constituent une ressource précieuse pour la détection d'anomalies, l'investigation d'incidents et la garantie de la conformité aux réglementations en matière de cyber-sécurité. En analysant les fichiers log, les organisations peuvent identifier des comportements suspects, anticiper des attaques potentielles et renforcer la résilience de leurs infrastructures informatiques.

Ce chapitre vise à fournir une compréhension approfondie des fichiers journaux et de leur rôle essentiel dans la cyber-sécurité, en mettant l'accent sur une exploration détaillée des techniques d'analyse avancées.

II.2. Concepts de base

II.2.1. Définition des fichiers journaux

Les fichiers journaux sont des enregistrements chronologiques de toutes les activités qui se produisent dans un système, incluant des événements tels que les transactions, les erreurs et les intrusions. Ces données, transmises de diverses manières, peuvent être structurées, semi-structurées ou non structurées [6].

II.2.2. Types de fichiers journaux

Presque chaque composant d'un réseau génère un type de données différent, et chaque composant collecte ces données dans son propre log. Par conséquent, il existe de nombreux types de logs [6], citons principalement :

- **Logs de Serveur :** Ce sont des documents textuels contenant un enregistrement des activités liées à un serveur spécifique sur une période donnée.
- **Logs Système (Syslog) :** Ce sont des Enregistrements des événements du système d'exploitation, incluant les messages de démarrage, les changements système, les arrêts inattendus, les erreurs et avertissements, et d'autres processus importants.
- **Logs d'Autorisation et d'Accès :** Ils contiennent une liste des personnes ou des bots accédant à certaines applications ou certains fichiers.
- **Logs de Changements :** Ils Incluent une liste chronologique des modifications apportées à une application ou à un fichier.
- **Logs d'Erreur :** Ils sont utiles pour le dépannage, car ils fournissent des informations sur les erreurs, les exceptions et les avertissements générés par les logiciels et les systèmes.

II.2.3. Format des messages logs

Un format de log est une structure normalisée qui permet aux logs d'être lisibles par machine et facilement analysables. L'un des principaux défis des fichiers journaux réside dans le fait qu'ils sont généralement des données textuelles non structurées, ce qui rend difficile l'interrogation des logs pour obtenir des informations utiles. L'utilisation de logs structurés et d'un système de gestion de logs qui les prend en charge représente une avancée significative [7].

Il existe actuellement plusieurs formats de journaux, parmi lesquels nous pouvons citer [8] :

- Format de fichier de log étendu W3C (ELF)
- Log d'accès Apache
- Cisco SDEE/CIDEE
- Format d'événement commun ArcSight (CEF)
- Syslog (RFC3195 et RFC5424 plus récents)

- IDMEF, un format basé sur XML

II.2.4. Structure d'un message log web

Chaque entrée de journal comprend des champs spécifiques qui offrent des informations essentielles sur le comportement des utilisateurs et la performance du système. De manière générale, la structure de base d'un fichier de log se compose de [7] :

- **Horodatage :** Indique le moment précis où l'événement enregistré s'est produit.
- **Informations sur l'utilisateur :** Identifie l'utilisateur associé à l'événement.
- **Informations sur l'événement :** Décrit l'action qui a été effectuée.

Selon le type de source de log, le fichier peut également contenir une multitude de données pertinentes, telles que la page web référencée, le code d'état HTTP, les octets servis et les agents utilisateurs, dans le cas des logs de serveur.

Voici une entrée typique d'un serveur web Apache au format de log commun (CLF) :

Figure2.1 : entrée d'un serveur web Apache

```
192.168.1.10- - [12/Feb/2025:14:35:21 +0100] "GET / login.php HTTP/1.1" 200 2326
"https://example.com/home" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36''
```

Figure2.1 : entrée d'un serveur web Apache

Le tableau 2.1 présente en détail les différents champs d'un journal de serveur web.

Champ	Valeur dans l'Exemple	Interprétation
Adresse IP	192.168.1.10	Identifie l'adresse IP source de l'utilisateur accédant à l'application web.
Horodatage	[12/Feb/2025:14:35:21 +0100]	Indique la date et l'heure exactes d'accès, utile pour corréler avec d'autres événements de sécurité.
Méthode http	"GET"	Spécifie que l'utilisateur a effectué une requête GET (récupération d'une ressource).
Ressource Demandée	"/login.php"	L'utilisateur a accédé à la page de connexion (login.php), ce qui peut être pertinent pour détecter des tentatives de force brute.
Version http	"HTTP/1.1"	Spécifie la version du protocole utilisée pour la requête.
Code d'État http	200	Code de réponse du serveur : 200 signifie une requête réussie. Des erreurs comme 403 (Interdit) ou 500 (Erreur interne du serveur) pourraient indiquer des attaques.
Taille de la Réponse	2326	Nombre d'octets retournés au client. Des valeurs anormalement basses ou élevées pourraient indiquer une anomalie.
HTTP Referer	"https://example.com/home"	La page précédente visitée par l'utilisateur, utile pour détecter des redirections malveillantes ou des tentatives de phishing.
User-Agent	"Mozilla/5.0 (Windows NT 10.0; Win64; x64)..."	Fournit des détails sur le navigateur et le système d'exploitation utilisés par l'utilisateur. Peut aider à identifier des bots ou des requêtes suspectes.

Tableau 2.1 : différents champs d'un journal de serveur web.

II.3. Analyse des fichiers log

II.3.1. Importance de l'analyse des fichiers journaux pour la securite

L'analyse des logs est un composant crucial de la cyber-sécurité, permettant aux organisations de détecter et de répondre de manière proactive aux menaces de sécurité. Les logs enregistrent les activités des systèmes, les événements réseau et les comportements des utilisateurs, ce qui présente des ressources précieuses pour identifier les anomalies, les comportements malveillants et les tentatives d'attaque.

II.3.2. Cas d'utilisation de l'analyse logs

- **DETECTION D'ANOMALIES ET DE COMPORTEMENTS MALVEILLANTS**
    - **Activité système inhabituelle :** Les logs peuvent révéler des pics inattendus dans l'utilisation des ressources ou un accès non autorisé à des fichiers critiques.
    - **Tentatives de connexion suspectes :** Des échecs répétés de connexion peuvent signaler une attaque par force brute.
    - **Irrégularités réseau :** Des transferts de données inattendus ou des activités de balayage de ports anormales pourraient indiquer des tentatives de reconnaissance par des attaquants.[9]

- **Identification des tentatives d'attaque**
    - **Attaques par force brute :** Plusieurs tentatives d'authentification échouées peuvent indiquer qu'un attaquant tente d'accéder de manière non autorisée.
    - **Attaques par injection SQL et XSS :** Les logs des applications web aident à identifier les tentatives d'injection en analysant les requêtes anormales.
    - **Activité malveillante :** Les logs peuvent capturer des exécutions de commandes suspectes ou des modifications de fichiers non autorisées qui peuvent indiquer la présence de logiciels malveillants.[9]

- **Enquête sur les incidents et analyse forensique**
    - **Tentatives d'escalade de privilèges :** Une entrée de log montrant la création d'un nouvel utilisateur avec des privilèges root (UID=0) peut indiquer une compromission du système.
    - **Détection de l'exfiltration de données :** Les logs peuvent mettre en évidence de grands volumes de trafic sortant d'un système compromis.[9]

En tirant parti de l'analyse des logs, les organisations améliorent leur capacité à atténuer les incidents de sécurité, à prévenir les violations de données et à améliorer leur posture globale en matière de cyber-sécurité.

II.3.3. Processus d'analyse des fichiers log

Le processus d'analyse des fichiers log peut être décomposé en plusieurs étapes clés [8], illustrées dans le figure 2.1 :

Figure II.2 : processus d'analyse des fichiers log

II.3.3.1. Collecte et centralisation

La collecte des logs est la première étape du traitement des logs. Une fois que les logs sont générés par divers dispositifs et ordinateurs, il est recommandé de collecter tous les logs sur une infrastructure centrale appelée collecteur. Un collecteur est un système informatique, généralement un serveur Unix ou Windows, où les messages de log sont rassemblés.[8]

Il existe deux méthodes de collecte, illustrés dans le tableau 2.2 : la transmission (mécanisme push) et l'extraction (mécanisme pull).[8]

Tableau **II.**2 : Comparaison des méthodes de collecte des logs

Méthode de collecte	Description	Avantages	Inconvénients
Transmission (Push)	Les sources de logs envoient activement les données de log au collecteur.	Moins de charge sur le collecteur, transmission en temps réel des logs, adaptée aux environnements avec des exigences de sécurité strictes.	Peut entraîner une surcharge réseau si de nombreuses sources envoient des logs simultanément, nécessite une configuration sur chaque source de log.
Extraction (Pull)	Le collecteur interroge périodiquement les sources de logs pour récupérer les données.	Centralisation de la configuration, meilleure gestion des ressources du collecteur, adaptée aux environnements avec des ressources limitées sur les sources de logs.	Peut entraîner des retards dans la collecte des logs, charge plus importante sur le collecteur, nécessite des autorisations d'accès aux sources de logs.

Cette comparaison met en évidence les compromis entre les différentes méthodes de collecte de logs, démontrant que le choix optimal dépend de facteurs tels que la charge réseau, les exigences de sécurité et la complexité de l'infrastructure de logging.

II.3.3.2. Prétraitement

Les fichiers log sont par défaut des données textuelles non structurées. Pour les transformer en logs structurés, ils doivent subir un prétraitement, généralement effectué en trois phases : l'analyse syntaxique (parsing), la normalisation et le filtrage.[8]

- **Analyse Syntaxique (Parsing) :** Ce processus consiste à diviser les données en blocs d'informations plus faciles à manipuler et à stocker. L'objectif est de reconnaître et de regrouper ces données de manière significative.
- **Normalisation :** Cette étape implique de convertir les logs dans le même format pour garantir que les entrées de journaux de différents types et sources expriment les informations de manière uniforme. Par exemple, si les logs de diverses sources contiennent des formats de date et d'heure différents, ils doivent être normalisés avant de continuer.
- **Filtrage :** Cette phase consiste à ignorer les entrées qui ne sont pas utiles pour l'analyse des logs et à conserver uniquement celles susceptibles d'être d'intérêt pour l'analyse. Cela réduit la quantité de données à traiter.

II.3.3.3. Stockage et indexation

Après le prétraitement, les logs doivent être disponibles avec une faible latence et stockés dans une base de données permettant la recherche. Les formats de stockage les plus couramment utilisés sont les formats texte, binaire et base de données.

L'indexation implique la création d'une structure de données appelée index, qui permet des recherches plus rapides en utilisant des mots-clés et/ou des opérateurs logiques.[8]

II.3.3.4. Analyse

L'analyse des logs est l'une des parties les plus importantes du processus de traitement ; c'est la capacité à effectuer une série de traitements automatiques, de manière itérative et de plus en plus précise et ciblée. Cette partie sera détaillée dans la section suivante.

II.3.3.5. Supervision et alerte

La supervision repose sur trois actions essentielles :[8]

- **Visualisation :** Présenter les résultats de l'analyse sur une interface conviviale et dans un format plus compréhensible pour un humain, comme des graphiques et des diagrammes, ce qui facilite la détection des problèmes et la recherche de leurs causes.
- **Rapports :** Générer des rapports analytiques pour résumer les activités au sein du système ou fournir des informations détaillées sur un événement spécifique. Les rapports sont utiles pour présenter les résultats à des personnes qui n'ont pas de connaissances techniques.
- **Alertes :** Fournir des alertes en temps réel lors de la détection d'un événement significatif (tel qu'une intrusion) pour attirer l'attention de l'équipe de surveillance. Les alertes peuvent être envoyées sous forme de courriel ou de message texte.

II.4. Méthodes d'analyses des fichiers journaux

II.4.1. Analyse manuelle des fichiers log

L'analyse manuelle des fichiers log est réalisée par un humain interprétant les logs, parfois avec l'aide d'outils informatiques. Les logiciels peuvent aider à organiser de grands ensembles de données et à rechercher des motifs spécifiques. Cette approche est souvent complétée par des enquêtes et des entretiens pour enrichir la compréhension.

II.4.1.1. Techniques

- **Correspondance de Modèles :** Recherche de motifs de signatures d'attaque connues ou d'activités suspectes, telles que des tentatives de connexion échouées répétées ou un comportement système anormal.
- **Examen Contextuel :** Révision des logs dans leur contexte, où l'expert possède une connaissance approfondie des opérations du système et peut identifier des comportements qui s'écartent de la norme.

II.4.1.2. Avantages

- **Enquête Détaillée :** L'analyse manuelle permet un examen approfondi des données log, crucial pour comprendre des attaques complexes ou sophistiquées.
- **Perspicacité d'Expert :** Les experts humains peuvent appliquer leur expérience et leur intuition, identifiant des motifs que les systèmes automatisés pourraient manquer, notamment dans des cas complexes ou uniques.
- **Flexibilité :** Le processus peut être adapté en fonction des besoins spécifiques ou des exigences de l'enquête de sécurité.

II.4.1.3. Défis

- **Problèmes d'Évolutivité :** La révision manuelle des logs ne s'adapte pas à l'augmentation des tailles de fichiers log. Alors qu'analyser quelques lignes est gérable, les environnements d'entreprise génèrent des millions de logs, parfois à un rythme de 20 000 enregistrements par seconde. À cette échelle, l'analyse manuelle devient inefficace et peu pratique.[8]
- **Difficulté avec les Logs Complexes :** La révision manuelle fonctionne bien pour des logs simples qui expriment clairement des erreurs. Cependant, pour des formats de logs obscurs, non documentés ou complexes, l'interprétation manuelle devient chronophage et peu fiable. Les analystes peuvent passer des heures à rechercher une seule entrée de log. [8]
- **Manque de Vue d'Ensemble :** Les outils simples et la révision manuelle sont utiles pour inspecter des entrées de log individuelles, mais ne fournissent pas une vue complète du comportement global du système. Des outils plus avancés peuvent corréler plusieurs sources de logs pour générer des informations significatives.[8]
- **Défis de Corrélation :** Les logs provenant de différentes sources doivent souvent être analysés ensemble pour détecter des motifs ou des menaces à la sécurité. Bien que cela puisse être fait manuellement, l'effort requis augmente considérablement, rendant l'automatisation une solution plus efficace.[8]

- **Fatigue et Ennui des Analystes :** L'analyse des logs est souvent répétitive et fastidieuse. La révision manuelle peut entraîner de la fatigue, augmentant la probabilité d'erreurs humaines. De plus, l'ennui peut réduire l'efficacité et la concentration d'un analyste au fil du temps. [8]

II.4.2. Analyse automatisée des fichiers log

L'analyse automatisée des fichiers log est un programme informatique qui traduit les entrées de log en métriques spécifiées sans intervention humaine. Elle se base sur des outils et des techniques avancées. [10]

II.4.2.1. Techniques

- **Analyse Basée sur des Règles :** Utilisation de règles et de motifs prédéfinis (par exemple, signatures d'attaques connues) pour identifier des menaces spécifiques dans les logs.[8]
- **Détection d'Anomalies :** Utilisation de méthodes statistiques ou d'algorithmes d'apprentissage automatique pour identifier des écarts par rapport au comportement normal, signalant des menaces potentielles ne correspondant pas à des motifs connus.[8]
- **Agrégation et Centralisation des Logs :** Collecte des logs provenant de plusieurs sources, normalisation et centralisation des données pour une analyse plus facile et efficace.[8]

II.4.2.2. Outils populaires

- **Splunk [8] :** Une plateforme puissante pour rechercher, surveiller et analyser de grandes quantités de données générées par des machines via une interface de type web. Elle est largement utilisée pour sa capacité à indexer et analyser de grands volumes de logs et pour son support analytique en temps réel.

- **ELK Stack (Elasticsearch, Logstash, Kibana) [8] :** Un ensemble d'outils open-source populaire utilisé pour l'analyse des logs en temps réel. Elasticsearch stocke les logs, Logstash les traite et les normalise, et Kibana fournit des visualisations des données log.
- **Graylog [8]:** Une plateforme open-source qui se concentre sur la gestion et l'analyse des logs, offrant une journalisation centralisée avec des capacités de recherche et d'alerte puissantes.

II.4.2.3. Avantages

- **Efficacité :** L'analyse automatisée peut traiter de grands volumes de données log rapidement, réduisant significativement le temps nécessaire pour identifier des menaces.
- **Évolutivité :** Les systèmes automatisés peuvent gérer des logs provenant de systèmes grands et complexes, les rendant adaptés aux environnements avec d'énormes quantités de données.
- **Surveillance en Temps Réel :** Les systèmes automatisés peuvent effectuer une analyse en temps réel et générer des alertes instantanées lorsqu'une activité suspecte est détectée, permettant des temps de réponse plus rapides.
- **Réduction des Erreurs Humaines :** L'automatisation réduit le risque de négliger des détails importants ou de commettre des erreurs, améliorant ainsi la précision de l'analyse.

II.4.2.4. Défis

- **Faux Positifs :** Les systèmes automatisés, en particulier ceux utilisant une analyse basée sur des règles, peuvent générer un nombre élevé de fausses alertes, nécessitant une intervention manuelle pour vérifier les alertes.
- **Complexité :** La configuration et la mise en place de systèmes d'analyse des logs automatisés peuvent être complexes et nécessiter des connaissances ou une formation spécialisée.

- **Manque de Contexte :** Les systèmes automatisés peuvent avoir du mal à comprendre le contexte plus large de certaines entrées de log, ce qui peut entraîner des interprétations erronées des données ou des menaces manquées.
- **Dépendance Excessive aux Règles :** Les systèmes basés sur des règles sont limités à la détection de motifs d'attaques connus et sont inefficaces pour identifier de nouvelles menaces sans données d'apprentissage suffisantes.

Le tableau **II**.3 : Comparaison des deux méthodes d'analyse des logs [11][12]

Caractéristique	Analyse Manuelle	Analyse Automatisée
Description	Interprétation humaine des logs, parfois avec des outils assistés par ordinateur.	Les programmes informatiques traduisent les logs en métriques sans intervention humaine.
Cas d'Utilisation	Analyse qualitative, enquêtes complexes, recherche.	Analyse quantitative, surveillance à grande échelle, détection des menaces en temps réel.
Techniques	Correspondance de Modèles, Examen Contextuel	Analyse Basée sur des Règles, Détection d'Anomalies, ML, Agrégation et Centralisation des Logs
Outils/Technologies	Tableurs, éditeurs de texte, outils de recherche basiques, cadres théoriques (par exemple, théorie ancrée).	Splunk, ELK Stack (Elasticsearch, Logstash, Kibana), Graylog, langages de script (par exemple, Perl).
Avantages	Enquête détaillée sur des attaques complexes, perspicacité d'expert, flexibilité pour s'adapter aux besoins spécifiques.	Efficacité dans le traitement de grands volumes de données, évolutivité pour des systèmes complexes, surveillance en temps réel, réduction des erreurs humaines.
Inconvénients	Problèmes d'évolutivité pour les grands ensembles de données, difficulté avec les logs complexes, manque de vue d'ensemble, défis de corrélation, fatigue des analystes.	Possibilité de faux positifs, complexité dans la configuration, manque de contexte, dépendance excessive aux règles.

En résumé, bien que l'analyse manuelle des logs offre des informations précieuses pour des enquêtes spécifiques, ses limitations en matière d'évolutivité et d'efficacité nécessitent l'adoption d'approches automatisées pour l'analyse des logs à grande échelle et en temps réel

dans les systèmes modernes. Les compromis entre ces méthodes soulignent l'importance de sélectionner la stratégie la plus appropriée en fonction du contexte spécifique et des objectifs de l'analyse.

II.4.3. Défis actuels de l'analyse des fichiers log

- **Volume de Données Élevé et Gestion des Big Data :** Le volume de données générées par les logs a considérablement augmenté avec l'avènement de l'informatique en nuage, des réseaux hybrides et de la transformation numérique. Chaque aspect des systèmes modernes produit des données log, rendant difficile pour les organisations de gérer et d'analyser efficacement cette vaste quantité d'informations. Ce volume énorme de données log nécessite des systèmes avancés pour la collecte, le traitement et l'analyse, capables d'identifier rapidement des informations précieuses.[10]
- **Complexité des Formats de Log et Besoin d'Outils Appropriés :** Un autre défi est le manque de standardisation des formats de fichiers log. Les logs peuvent être structurés, semi-structurés ou non structurés, en fonction du système ou de l'application qui les génère. Cette incohérence rend difficile l'extraction d'informations significatives des données en temps réel. Pour une gestion et une analyse efficace des logs, la normalisation des données log est essentielle. Sans standardisation, les organisations ont du mal à interpréter les logs provenant de sources diverses, nécessitant des outils spécialisés pour rendre les données facilement lisibles et exploitables.[12]
- **Analyse en Temps Réel et Gestion des Faux Positifs :** L'analyse des logs en temps réel pose des défis importants, surtout lorsqu'il s'agit de traiter d'énormes quantités de données. La détection immédiate des menaces ou des anomalies est cruciale, mais la complexité des données log et la fréquence des faux positifs compliquent cette tâche. L'analyse en temps réel des logs nécessite des systèmes avancés capables de réduire de tels erreurs tout en améliorant la précision de la détection.[12]
- **Diversité des Sources de Log et Difficultés d'Intégration des Données :** Les données log proviennent de plusieurs sources, souvent hétérogènes. Celles-ci peuvent inclure des systèmes, des applications, des réseaux et des dispositifs, chacun générant des logs dans des formats et des structures différentes. L'intégration de ces logs dans un

système cohérent est un défi. Le besoin d'une plateforme centralisée et standardisée pour la gestion des logs devient évident alors que les organisations s'efforcent de rassembler des données provenant de diverses sources, y compris des infrastructures en nuage et sur site. Le processus d'intégration nécessite un système sophistiqué capable de gérer et de corréler les logs provenant d'environnements divers.[12]

- **Analyse Manuelle des Logs dans de Grands Systèmes :** Dans les grands systèmes, l'analyse manuelle des logs devient très complexe. Un expert ou un administrateur doit comprendre le comportement du système et les logs qu'il génère. Cependant, avec de nombreuses applications et des formats de logs divers, il devient de plus en plus difficile de trouver un expert possédant toutes les connaissances nécessaires. De plus, l'analyse manuelle est chronophage et inefficace. Par conséquent, l'analyse automatisée des logs est fortement souhaitable, car elle peut traiter rapidement et avec précision de grands volumes de données, sans nécessiter une expertise étendue.[13]
- **Défis liés à la Programmation Explicite :** Les systèmes à grande échelle se composent souvent de plusieurs composants construits avec différents langages de programmation. Les logs de ces composants peuvent varier considérablement, et il n'existe pas de structure de sortie standard pour les fichiers log. La mise en œuvre d'un analyseur de logs unique pour un système aussi diversifié via une programmation explicite est un défi majeur. Le manque de standardisation dans les structures de fichiers log rend difficile la création d'un analyseur de logs universel, en particulier dans les systèmes à grande échelle.[13]

L'Apprentissage Automatique comme Solution : Étant donné que l'analyse manuelle et la programmation explicite ne suffisent pas à gérer la complexité des fichiers journaux, des techniques d'apprentissage automatique s’imposent actuellement pour automatiser le processus. L'apprentissage automatique permet aux systèmes d'apprendre à partir de jeux de données d'entraînement et de découvrir des problèmes cachés au sein des logs, même lorsque les formats et les contenus diffèrent. Cette approche ne nécessite pas de programmation explicite ni de connaissances spécialisées en analyse de logs, ce qui en fait un outil puissant pour automatiser l'analyse des logs et découvrir des menaces potentielles.[13]

L'intelligence artificielle, en particulier l'apprentissage automatique, est devenu une solution efficace pour l'analyse avancée des fichiers log. Contrairement aux systèmes basés sur des règles, les modèles d'IA peuvent apprendre à détecter des anomalies sans nécessiter de signatures prédéfinies.

II.5. Synthèses et conclusion

Dans ce chapitre, nous avons exploré les principes fondamentaux de l'analyse des fichiers log, en nous concentrant sur les différents types et structures de fichiers log couramment trouvés dans les systèmes de cyber-sécurité. Les fichiers log servent de ressource précieuse pour la surveillance et la détection d'événements, offrant des perspectives sur le comportement des systèmes et des applications. Nous avons examiné les approches clés de l'analyse des logs, en commençant par les méthodes basées sur des règles et des signatures. Ces méthodes, malgré leur utilisation répandue présentent des limites en termes d'efficacité. La dépendance aux signatures et motifs prédéfinis les rend vulnérables aux menaces émergentes, en particulier aux attaques zero-day, et nécessite des mises à jour régulières de la base de données pour maintenir leur pertinence. De plus, l'effort manuel impliqué dans la maintenance de ces systèmes ajoute une couche supplémentaire de complexité et d'inefficacité potentielle.

Bien que l'analyse basée sur des règles offre un cadre structuré pour la détection des menaces, il est clair que des solutions plus dynamiques et adaptatives sont nécessaires pour faire face à la sophistication croissante des menaces cybernétiques. À mesure que les systèmes évoluent, les méthodes traditionnelles doivent être complétées par des technologies plus avancées capables d'apprendre des motifs de données et de s'adapter à de nouveaux vecteurs d'attaque.

Cela nous amène naturellement à la phase suivante de la recherche, qui explore le rôle de l'intelligence artificielle (IA) dans l'amélioration de l'analyse des fichiers log. En particulier, l'IA offre la possibilité de dépasser la détection statique basée sur des règles en s'appuyant sur des algorithmes d'apprentissage automatique pour la détection d'anomalies, l'analyse des comportements et l'apprentissage automatisé.

Le chapitre suivant approfondira les méthodes basées sur l'IA, explorant comment elles peuvent répondre aux limitations des approches traditionnelles et fournir des solutions plus robustes et adaptatives pour détecter et atténuer les menaces cybernétiques.

CHAPITRE 3 : ETUDE DES APPROCHES EXISTANTES D'ANALYSE DES JOURNAUX WEB BASEES SUR LE ML

III.1. Introduction

Les logs, en tant que référentiels d'informations critiques concernant l'opérationnalisation des systèmes informatiques et les interactions des utilisateurs, constituent un élément fondamental pour l'identification et l'analyse des incidents de sécurité. Néanmoins, l'accroissement exponentiel du volume de ces données pose des défis significatifs au regard des méthodologies de traitement conventionnelles, fréquemment basées sur des systèmes de règles statiques. Bien que pertinentes dans des contextes simplifiés, ces techniques démontrent une efficacité limitée face à la complexification croissante des vecteurs d'attaque contemporains.

Afin de pallier ces contraintes, l'intégration de l'intelligence artificielle (IA) et de l'apprentissage automatique (ML) apparaît comme une nécessité. Ces approches offrent la possibilité d'une analyse des journaux plus rapide, plus précise et intrinsèquement adaptative, permettant ainsi une détection des attaques de manière plus proactive et prédictive

Ce chapitre explore l'application des techniques d'intelligence artificielle à l'analyse des journaux Web. Il débute par une introduction aux concepts fondamentaux de l'IA et de l'apprentissage automatique, propose une étude comparative des approches actuelles, puis conclut par une synthèse des différentes approches analysées.

III.2. Intelligence artificielle et l'apprentissage automatique

III.2.1. Définitions et concepts de base

III.2.1.1. Intelligence artificielle (IA)

L'intelligence artificielle (IA) est un domaine de l'informatique qui vise à concevoir des systèmes capables d'effectuer des tâches qui nécessitent normalement l'intelligence

humaine. Ces tâches incluent la reconnaissance vocale, la prise de décision et l'analyse de données complexes. L'IA s'appuie sur des modèles et des algorithmes qui simulent des processus cognitifs humains, tels que l'apprentissage et la résolution de problèmes [15].

➤ Dans le contexte de la sécurité informatique, l'IA est utilisée pour la détection d'intrusions, la surveillance automatisée et la gestion des menaces. Les systèmes d'IA peuvent analyser les flux de données en temps réel pour identifier les comportements suspects ou les attaques potentielles, ce qui réduit le temps de réponse aux incidents de sécurité.

III.2.1.2. APPRENTISSAGE AUTOMATIQUE (Apprentissage Automatique, ML)

L'apprentissage automatique est une branche de l'IA qui permet aux machines d'apprendre à partir de données, sans être explicitement programmées pour chaque tâche. Le ML utilise des algorithmes pour analyser les données, identifier des motifs ou des tendances, et effectuer des prédictions ou des classifications basées sur ces observations [15].

➤ En sécurité informatique, le ML est utilisé pour améliorer la détection des anomalies, prédire les attaques et identifier les logiciels malveillants. Des modèles de ML peuvent être entraînés sur de vastes ensembles de données pour détecter de nouvelles menaces de cybersécurité, en analysant les comportements du réseau ou les fichiers suspects.

III.2.2. Types d'apprentissage en ml

Le ML regroupe plusieurs types d'apprentissage, chacun adapté à des contextes et objectifs spécifiques. Voici les principales catégories **[14]**, illustrées dans la figure 3.1 **:**

III.2.2.1. Apprentissage supervisé

L'apprentissage supervisé consiste à entraîner un modèle à partir de données étiquetées, c'est-à-dire que chaque exemple d'entrée est associé à une sortie attendue (étiquette). Le modèle apprend à prédire cette sortie en établissant une relation statistique entre les variables d'entrée et les résultats cibles. Parmi les algorithmes couramment utilisés, on trouve notamment les arbres de décision, les machines à vecteurs de support (SVM) et les réseaux neuronaux.

Ces techniques sont particulièrement efficaces pour des tâches telles que :

- La classification binaire (ex. : spam/non-spam),
- La classification multiple (ex. : reconnaissance d'images en plusieurs catégories)
- La régression (prédiction de valeurs numériques continues, comme le prix d'un bien immobilier).

III.2.2.2. Apprentissage non supervisé

À la différence de l'apprentissage supervisé, l'apprentissage non supervisé est utilisé lorsque les données ne sont pas étiquetées. L'objectif est alors de découvrir des structures, des motifs ou des regroupements cachés dans les données, sans connaissance préalable du résultat attendu.Parmi les principaux algorithmes non supervisés, nous citons l'algorithme des k-means qui regroupe les données similaires en différents clusters.
Une application importante de ces techniques est la détection d'anomalies, qui vise à repérer des observations atypiques ou suspectes dans un jeu de données. Cela s'avère particulièrement utile pour l'analyse comportementale des utilisateurs, ou la détection d'activités anormales dans un système.

III.2.2.3. Apprentissage par renforcement

Dans l'apprentissage par renforcement, un agent intelligent apprend en interagissant avec un environnement, recevant des récompenses ou pénalités selon les actions entreprises. L'objectif est de maximiser les gains cumulés au fil du temps.

Cette approche est utilisée dans des domaines tels que les jeux vidéo, la robotique, ou encore la réponse automatisée aux incidents de cybersécurité, où l'agent apprend à prendre des décisions optimales face à des menaces évolutives.

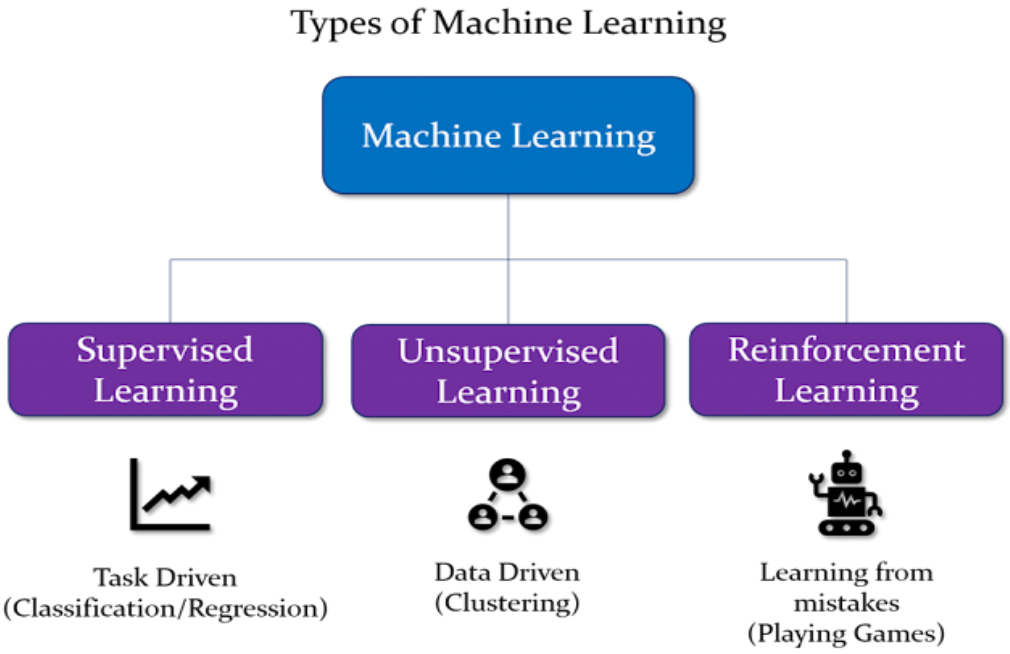


Figure III.1 : Types d'apprentissage automatique

III.3. Étapes clés du processus d'apprentissage automatique

La mise en œuvre d'un pipeline d'apprentissage automatique pour l'analyse des journaux web se décompose en six phases interdépendantes, illustrées dans la figure 2.2, chacune conditionne la qualité, la robustesse et l'opérabilité du système en production **[14].**

Cette section offre un aperçu synthétique des étapes clés du processus, illustré par quelques techniques, modèles et algorithmes, qui seront décrites en détail en annexe et développées dans les chapitres suivants.

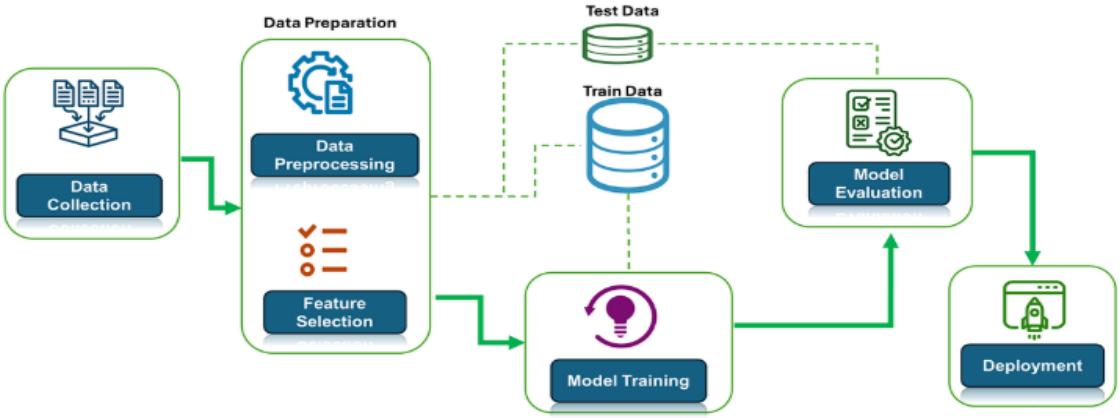


Figure III.2: Processus d'apprentissage automatique

III.3.1. Collecte et préparation des données

La première phase consiste à rassembler les fichiers log émis par les serveurs web, les applications et les dispositifs de sécurité, puis à effectuer un prétraitement rigoureux, qui englobe plusieurs sous-étapes, dont nous citons :

- **Prétraitement :** Cette étape comprend diverses opérations de base sur les données brutes, telles que le formatage (conversion des données dans un format uniforme comme CSV, JSON ou texte brut), l'analyse syntaxique (parsing) des champs des journaux pour extraire des informations pertinentes, et la conversion des types de données**.**

- **Nettoyage :** traitement des valeurs manquantes (suppression ou imputation), élimination des doublons et filtrage des enregistrements corrompus pour éviter l'introduction de biais ou de bruit dans l'apprentissage.
- **Extraction de caractéristiques :** création de variables discriminantes à partir des requêtes HTTP (fréquence de requêtes, motifs d'URL, intervalles temporels) et génération de features liées à la session utilisateur (durée, nombre d'erreurs) **.**
- **Transformation** : normalisation des variables numériques (MinMaxScaler, StandardScaler), encodage des attributs catégoriels (One-Hot Encoding) ou textuels (TF-IDF, Word2Vec) et, si nécessaire, réduction de dimension **PCA** (Principal

Component Analysis) et **t-SNE** (t-distributed Stochastic Neighbor Embedding) pour optimiser l'espace de travail.

III.3.2. Sélection du modèle

Le choix de l'algorithme d'apprentissage dépend de la nature de la tâche et des contraintes opérationnelles **[14]**:

- **Tâches supervisées : elles sont** destinées principalement à la classification, elles privilégient des modèles tels que les forêts aléatoires, les machines à vecteurs de support (SVM) ou les arbres de décision, reconnus pour leur interprétabilité et leur aptitude à traiter des données hétérogènes.
- **Tâches non supervisées :** elles sontdédiées généralement à la détection d'anomalies, elles reposent sur des méthodes comme Isolation Forest, Density-Based Spatial Clustering of Applications with Noise (DBSCAN)
- ou d'autres algorithmes de clustering, permettant de repérer des observations atypiques sans nécessiter d'étiquetage préalable.
- **Contraintes opérationnelles :** le choix des algorithmes est guidé par la latence d'inférence requise (préférence pour des modèles légers en temps réel) et par les ressources de calcul disponibles (architecture GPU vs CPU) **.**

III.3.3. Entraînement

- **Optimisation :** ajustement des paramètres internes via des procédures tels que la descente de gradient ou de recherche en grille (Grid Search) pour minimiser la fonction de coût, tout en appliquant des techniques de régularisation (L1, L2, dropout) pour prévenir le surapprentissage**.**
- **Suivi :** monitoring des courbes d'apprentissage (train vs validation) et utilisation d'un jeu de validation pour détecter précocement les phénomènes de dérive ou de surapprentissage**.**

III.3.4. Validation et réglage des hyperparamètres

- **Méthodes de validation :** Pour évaluer la capacité de généralisation du modèle, on recourt à des méthodes de validation telles que le split hold-out simple ou la validation croisée k-fold**.**
- **Métriques :** Pour quantifier la performance selon la nature de la tâche, on utilise des métriques adaptées : précision, rappel, F1-score et AUC-ROC en classification ; MSE ou MAE en régression**.**
- **Ajustement :** Pour optimiser les paramètres critiques, on procède à des itérations de tuning via Grid Search, Random Search ou optimisation bayésienne sur les hyperparamètres (profondeur d'arbre, seuils d'anomalie, taux d'apprentissage).

III.3.5. Évaluation finale

- **Jeu de test indépendant :** mesure de la robustesse en conditions réelles à l'aide d'un dataset jamais exploité durant l'apprentissage.
- **Analyse des performances :** examen des matrices de confusion, des courbes ROC et des distributions d'erreur pour valider la conformité aux objectifs opérationnels (taux de faux positifs, temps de détection).

III.3.6. Déploiement et surveillance

- **Monitoring continu :** mise en place d'alertes pour détecter toute dérive (drift) de la distribution des données ou de la qualité prédictive.
- I**ndicateurs clés :** suivi de la latence d'inférence, du taux de faux positifs et de la stabilité des scores.
- **Maintenance :** stratégie de réentraînement périodique ou sur déclencheur (dérive détectée) et recalibrage des seuils pour garantir une efficacité durable en production [**14].**

III.4. Étude comparative des approches d'analyse de journaux basées sur l'apprentissage automatique :

Dans cette section, nous présentons une étude comparative de travaux récents portant sur l'analyse des journaux (logs) WEB appliquée à la détection d'anomalies et d'intrusions à l'aide de l'apprentissage automatique.

L'objectif est d'évaluer les performances des approches d'apprentissage automatique supervisées et non supervisées selon plusieurs critères : les ensembles de données utilisés, la préparation des données, les algorithmes appliqués, les métriques d'évaluation et les méthodes de validation.

Notre analyse repose sur un ensemble de travaux scientifiques sélectionnés. Nous présentons ici sept études représentatives : deux portants sur des approches supervisées (classification d'attaques) et cinq sur des approches non supervisées (détection d'anomalies).

III.4.1. Critères d'évaluation comparative

Pour analyser ces travaux, nous avons retenu principalement les critères d'évaluation suivants :

- **Qualité et structure des données d'entraînement :** La fiabilité des modèles d'apprentissage automatique repose avant tout sur la qualité des données exploitées. Nous avons donc examiné [16] :
  - La date de création et les mises à jour des jeux de données,
  - Leur contenu et la granularité des événements enregistrés,
  - Le volume total d'échantillons disponibles,
  - Leur accessibilité (publics, privés ou payants).
- **Préparation des données :** Cette étape, souvent négligée, est cruciale pour garantir que les données sont dans un format approprié et qu'elles sont exemptes d'erreurs et d'incohérences.
- **Algorithmes d'apprentissage utilisées :** Ce critère évalue la pertinence et l'adéquation des algorithmes choisis pour résoudre le problème posé.
- **Évaluation des performances :** Ce critère permet de mesurer l'efficacité des approches étudiées de manière objective et quantifiable.

III.4.2. Tableau comparatif des travaux étudiés

Les tableaux 1 et 2 présentent respectivement les approches d'apprentissage supervisé et les approches d'apprentissage non supervisé

III.4.2.1. Approches utilisant l'apprentissage supervisé

Titre	Jeu de données utilisé	Attributs sélectionnés	Techniques de pré-traitement	Méthodes utilisées	Objectif	Algorithmes	Performance
17	Logs provenant d'un serveur web Apache, attaques générées par Elk et Burp	Tokens extraits des URL et des paramètres, 2955 caractéristiques	Appliquer la PCA, extraire la présence/absence des tokens (0 ou 1), éliminer les champs inutiles	Classification supervisée pour les attaques SQLi, XSS, et DT	Détection des attaques sur le trafic et les logs	SVM, arbre de décision, K-NN, AdaBoost	Précision : 0.95, F1-score : 0.91-0.98
18	Trafic réseau capturé et logs d'un serveur web Apache (BWAPP)	Valeurs des entrées utilisateur (charges utiles), commandes SQL, événements XSS, sous-chaînes récurrentes	Normalisation via dictionnaire standard, remplacement avec des tokens, conversion des métacharactères	Apprentissage supervisé pour la classification des attaques	Détection des attaques web et anomalies dans les réseaux	Régression logistique, SVM	Précision : 0.9790, F1-score : 0.9789

III.4.2.2. Approches utilisant l'apprentissage non supervisé

Titre	Ensemble de données utilisé	Attributs sélectionnés	Techniques de pré-traitement	Méthodes utilisées	Objectif	Algorithmes	Performance
19	Journaux d'accès d'un site web d'une université BUPT, logs d'attaque MACCDC 2012	Adresse IP, user-agent	Normalisation des informations de journal, remplacement des majuscules et des lettres avec la structure du répertoire préservée	Entropie, Clustering	Détection de comportement malveillant à travers des séquences d'accès	Entropie, Clustering	Haute détection, faible taux de faux alarmes
20	Journaux d'un projet industriel	Code d'état HTTP, URL, paramètres, valeurs des paramètres, types de requêtes HTTP	Extraire des caractéristiques de logs, détection de comportements normaux ou anormaux	Modèles de Markov cachés (HMM), Classification	Détection d'anomalies dans les logs web industriels	HMM, Classification	Précision: 35,54%, taux de faux positifs: 4,09%

Titre	Ensemble de données utilisé	Attributs sélectionnés	Techniques de pré-traitement	Méthodes utilisées	Objectif	Algorithmes	Performance
21	Logs d'un serveur web de l'Université de Xi'an (XUPT)	URI, ClientIP, Temps, Octets, User-Agent	Analyse des logs et normalisation, traitement des variables catégorielles avec TF-IDF, normalisation des valeurs numériques	Forêt d'Isolation	Détection d'anomalies dans les logs web	Forêt d'Isolation	Précision: 60,30%
22	Logs de trafic capturés d'un serveur web Apache (BWAPP)	Valeurs d'entrée réseau (payloads), SQL, requêtes DNS, sous-chaînes	Conversion metacharacter, remplacement des tokens, normalisation dans un dictionnaire standard	k-moyennes, DB-SCAN, MeanShift, Analyser les flux réseau	Détection des attaques web et des anomalies	k-moyennes, DB-SCAN, MeanShift, Analyser les flux réseau	Haute précision dans la classification des attaques
23	Logs de trafic web	Non spécifié dans le document	Standardisation des données et gestion des caractéristiques	Forêt d'Isolation	Détection d'anomalies dans le trafic web	Forêt d'Isolation	Précision: 93%, F1-Score: 92%

III.5. Synthèse

L'analyse comparative des méthodes basées sur l'apprentissage automatique, présentée dans les Tableau 1 (apprentissage supervisé) et Tableau 2 (non supervisé), met en évidence les points suivants :

1. **Jeux de données et prétraitement**
    - **Supervisé** : exploitation de logs issus de serveurs Apache, de traffic réseau capturé (Elk, Burp, BWAPP) et d'environnements industriels. Les techniques de prétraitement comprennent l'extraction de « tokens » à partir des URLs, l'application de PCA, la normalisation et la conversion de caractères spéciaux [17, 18].
    - **Non supervisé** : utilisation de jeux de logs variés (universités, serveurs industriels, trafic web). Le prétraitement inclut la standardisation des numéros et lettres, la vectorisation TF-IDF, le traitement catégoriel et la normalisation numérique [19–22].
2. **Caractéristiques et méthodes**

- **Supervisé** : génération de dizaines à plusieurs milliers de caractéristiques à partir des requêtes HTTP (paramètres, payloads, séquences temporelles), suivie de classification supervisée (SVM, arbres de décision, k-NN, AdaBoost, régression logistique) pour détecter des attaques SQLi, XSS ou des comportements malveillants [17, 18].
- **Non supervisé** : mise en œuvre d'algorithmes de clustering ou de détection d'anomalies (Isolation Forest, DBSCAN, k-Averages, MeanShift, modèles de Markov cachés, méthodes basées sur l'entropie) pour repérer des observations atypiques sans étiquettes préalables [19–22].

3. **Objectifs et performances**
   - **Supervisé** : ciblage de la détection d'attaques connues et classification des requêtes anormales. Les modèles obtiennent des précisions de l'ordre de 95 % et des F1-scores compris entre 0,91 et 0,98.
   - **Non supervisé** : détection plus générale des comportements malveillants ou anormaux. Les taux de détection rapportés varient entre 93 % et 96,3 %, avec des taux de faux positifs modérés (autour de 4 % dans certains cas).
4. **Avantages et limites comparés**
   - **Supervisé** : forte capacité à reconnaître des attaques pour lesquelles des données étiquetées existent, mais dépendance aux mises à jour des signatures et coût élevé d'annotation.
   - **Non supervisé** : autonomie vis-à-vis des labels et flexibilité face à de nouvelles menaces, mais sensibilité aux faux positifs et difficulté à distinguer les anomalies bénignes des attaques réelles.

En conclusion, l'étude comparative montre que :

- Les méthodes supervisées sont performantes pour la détection d'attaques connues, dès lors qu'un corpus étiqueté est disponible, mais nécessitent un effort d'annotation et de maintenance des modèles.
- Les approches non supervisées offrent une plus grande autonomie et sont adaptées à la découverte de nouveaux comportements malveillants, au prix d'un réglage minutieux pour limiter les faux positifs.

Pour une solution complète d'analyse des logs web, il est recommandé d'adopter une **approche hybride**, combinant :

1. La robustesse des modèles supervisés pour la détection d'attaques déjà identifiées,
2. La flexibilité des algorithmes non supervisés pour la surveillance continue et la détection d'anomalies émergentes.

Cette stratégie permet d'allier précision et adaptabilité, en maximisant la capacité à détecter à la fois les attaques connues et celles encore inconnues.

III.6. Conclusion

Ce chapitre a présenté une analyse détaillée des techniques et des outils d'apprentissage automatique destinés à la détection d'anomalies dans les journaux de serveurs web, en réponse à la complexification des données et à l'évolution permanente des menaces. L'étude comparative des approches supervisées et non supervisées a mis en lumière leurs forces et leurs faiblesses respectives : précision élevée pour les attaques connues d'un côté, autonomie et capacité de découverte de nouveaux schémas malveillants de l'autre côté.

Face à ces constats, une **approche hybride** s'impose comme la solution la plus adaptée : elle combine la robustesse des modèles supervisés pour identifier rapidement les attaques déjà cataloguées et la flexibilité des méthodes non supervisées pour détecter des comportements émergents ou inédits. Cette synergie permet d'améliorer la couverture des menaces, de réduire les faux négatifs et de renforcer la capacité de réaction en temps réel.

Le chapitre suivant détaillera la méthodologie de conception de notre système d'analyse des fichiers journaux web ainsi que son architecture fonctionnelle globale et détaillée, en s'appuyant sur cette approche hybride comme socle de conception et de mise en œuvre.

CHAPITRE 4 : CONCEPTION DU SYSTÈME D'ANALYSE AVANCÉE DES FICHIERS LOG WEB

IV.1. Introduction

Dans un contexte où les menaces informatiques évoluent sans cesse, l'analyse automatisée des fichiers log web devient une nécessité pour détecter et contrer les attaques de manière proactive. Les logs web contiennent une richesse d'informations sur les requêtes entrantes, dont certaines peuvent indiquer des comportements malveillants tels que des injections SQL, des scans de vulnérabilités ou des attaques par force brute.

Ce chapitre présente la conception d'un système d'analyse avancée de ces fichiers log. L'objectif est de développer une architecture capable d'exploiter à la fois les méthodes d'apprentissage supervisé (pour détecter des attaques connues) et non supervisé (pour révéler des anomalies inconnues) afin de maximiser la couverture en matière de détection.

Nous détaillons dans ce chapitre l'architecture générale du système, les modules de collecte et de prétraitement des logs, ainsi que les modèles d'apprentissage *automatique* intégrés pour la détection d'anomalies et la classification des attaques.

IV.2. Rappel des objectifs de notre travail

Le déploiement croissant des applications web a malheureusement été accompagné par une augmentation exponentielle des cyberattaques. Les serveurs web, en tant que points d'entrées principaux, sont des cibles privilégiées. Les fichiers de log web, souvent sous-exploités, représentent pourtant une source d'information cruciale pour identifier ces menaces. La problématique à laquelle notre travail répond est donc la suivante : Comment concevoir un système automatisé et robuste capable de détecter efficacement les attaques web, y compris les attaques émergentes, à partir de l'analyse des fichiers log ?

Notre solution repose sur l'intégration de techniques d'apprentissage automatique (Apprentissage automatique) afin de :

- Détecter de manière proactive les cyberattaques présentes dans les fichiers log web.
- Identifier les comportements anormaux pouvant révéler des attaques émergentes ou inconnues (*zero-day*).
- Classer les attaques selon leur nature pour faciliter la réponse adaptée par les équipes de sécurité.

IV.3. Méthodologie de conception

IV.3.1. Architecture modulaire

Notre méthodologie repose sur une architecture modulaire visant à combiner les avantages des approches d'apprentissage non supervisé et supervisé pour une analyse intelligente des journaux web. L'objectif est de détecter les comportements malveillants, y compris les attaques inconnues, tout en maintenant une performance de traitement optimale.

IV.3.2. Approche d'Analyse Séquentielle Hybride

Le processus d'analyse suit un pipeline séquentiel où Chaque log brut est d'abord traité par le module de prétraitement. Ce module nettoie, normalise et formate les données, les rendant prêtes à l'analyse par les modules suivants. Le log prétraité est d'abord analysé par le module non supervisé pour la détection d'anomalies potentielles. En cas de détection, l'enregistrement est acheminé vers le module supervisé pour une classification détaillée des attaques spécifiques. Les classifications sont ensuite consolidées par l'heuristique décisionnelle, permettant d'identifier avec certitude les attaques connues, de signaler les comportements ambigus et de discriminer les comportements normaux des comportements suspects ou inconnus.

L'adoption de ce pipeline repose sur deux considérations essentielles :

1. **Optimisation des performances** : Étant donné que la majorité des requêtes dans les logs sont bénignes, le module non supervisé permet un filtrage préliminaire efficace, réduisant la charge du classifieur supervisé.

2. **Détection de menaces inconnues** : L'approche hybride renforce la capacité du système à identifier des attaques *zero-day* : les anomalies comportementales sont repérées en amont, et, lorsque cela est possible, la nature de l'attaque est précisée en aval par le classifieur.

IV.4. Architecture du système d'analyse avancee proposé

Notre architecture est structurée autour de quatre modules principaux interdépendants, représentés dans la figure VI.1 :

1. **Module de prétraitement :** est la première étape du système. Il nettoie et prépare les journaux web bruts en standardisant leur format, afin d'assurer la qualité et la pertinence des données pour les analyses futures.
2. **Module de détection d'anomalies** utilise un modèle non supervisé pour identifier les comportements inhabituels.
3. **Module de classification des attaques**, qui, grâce à des modèles supervisés, reconnaît et catégorise les types d'attaques connus (ex: XSS, SQLi).
4. **Module heuristique de décision** consolide ces résultats pour la prise de décision finale.

Les sections subséquentes décriront en profondeur le fonctionnement de chaque module de l'architecture proposée.

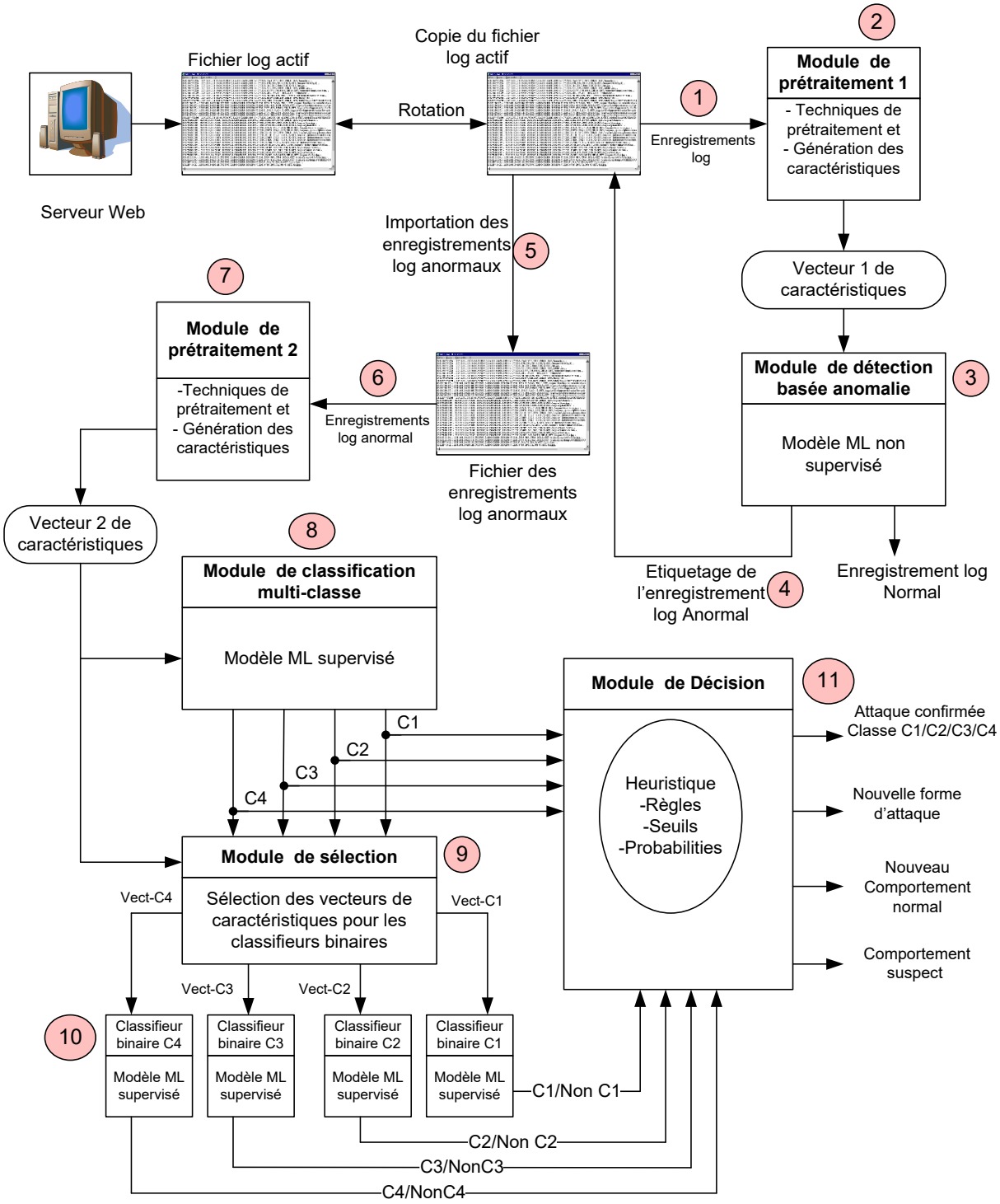


Figure IV.1 : Architecture générique du système d'analyse avancée de logs web proposé

IV.5. Description des principaux modules du système propose

Dans cette section, nous décrivons les principaux modules de notre solution. Chaque module est conçu pour assurer une chaîne de traitement robuste, de l'ingestion des logs jusqu'à la génération des caractéristiques exploitables par les modèles d'apprentissage automatique.

IV.5.1. Module de prétraitement

Le prétraitement des données est une étape déterminante dans toute démarche d'analyse fondée sur l'apprentissage automatique. En pratique, les données issues du monde réel sont souvent incomplètes, bruitées ou incohérentes. Le but du prétraitement est donc de convertir ces données brutes en un format structuré et fiable, garantissant qualité, cohérence et pertinence analytique.

Dans notre contexte, les logs web sont nettoyés et validés afin de corriger ou d'éliminer les enregistrements corrompus, redondants ou manquants. L'objectif est d'obtenir un jeu de données structuré et de qualité, facilitant la détection d'attaques. Les étapes principales sont présentées dans la figure correspondante

Les principales étapes du processus sont illustrées dans la Figure suivante :

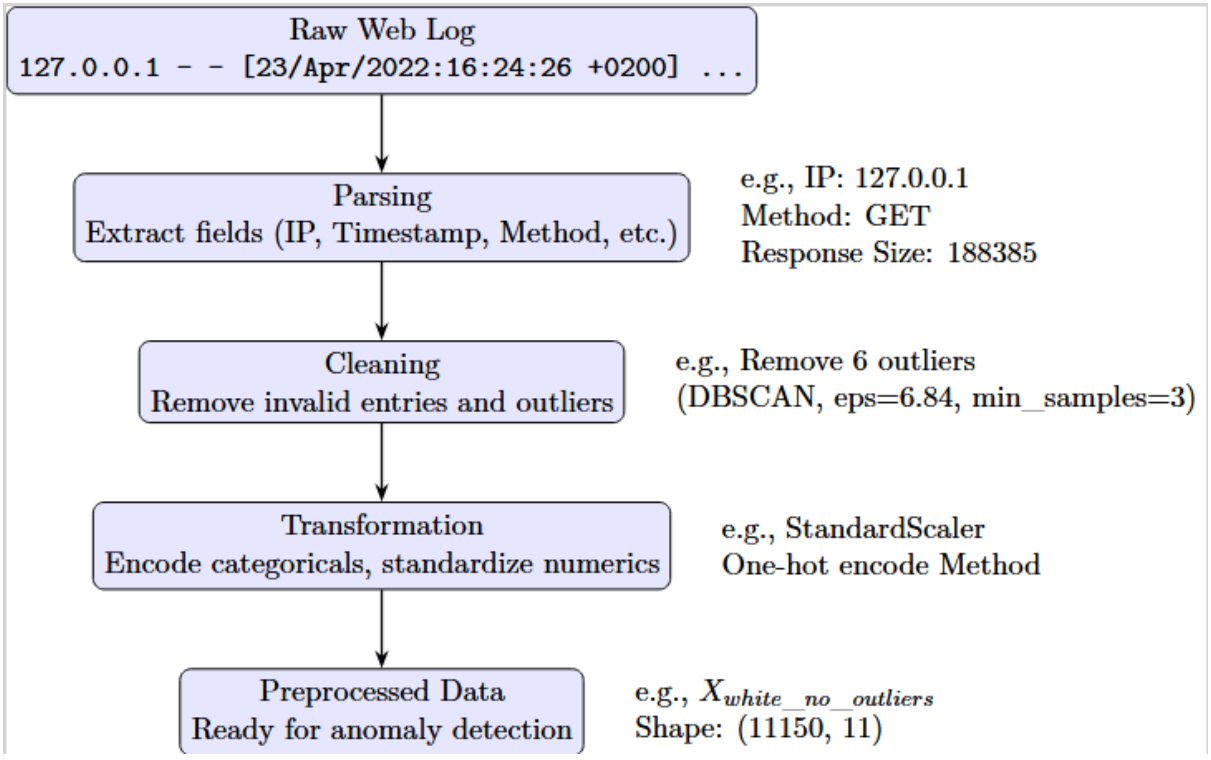


Figure IV.2 : Les phases de prétraitement des enregistrements logs

IV.5.1.1. Parsing

Cette phase consiste à extraire des lignes de logs brutes les champs pertinents pour l'analyse. Chaque ligne est segmentée selon le format du serveur web, pour obtenir un enregistrement lisible et exploitable par les algorithmes.

Par exemple, à partir d'une entrée de log issue du CERIST-DATASET, illustré dans la Figure **X** ci-dessous. on distingue typiquement :

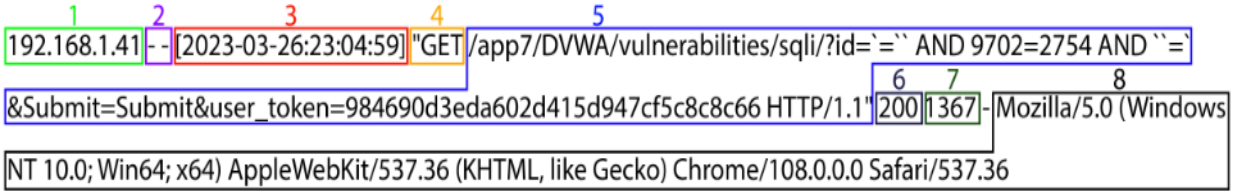


Figure IV.3 :Exemple d'un enregistrement log extrait du CERIST-Dataset 2023

1. L'adresse IP du client.
2. L'identifiant du client et le nom d'utilisateur (avec « - » ou « - - » si absents).
3. La date et l'heure de la requête (ex. : 26 mars 2023 à 04:59 »).
4. La méthode HTTP (GET, POST, etc.).
5. L'URL demandée et la version du protocole.
6. Le code de réponse (status code).
7. La taille de la réponse (bytes).
8. Le champ User-Agent, indiquant navigateur et système d'exploitation.

➤ 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*

IV.5.1.2. Nettoyage des données

Après parsing, le DataFrame ainsi constitué est soumis à un processus de nettoyage visant à gérer les valeurs manquantes, standardiser les formats et supprimer les doublons. Les opérations typiques sont :

- **Gestion des valeurs manquantes** :Les valeurs manquantes dans les champs essentiels (par exemple, method, url, protocol, referrer, user_agent) sont remplacées par des valeurs par défaut comme UNKNOWN, '/', ou des chaînes vides, selon le cas. Cela permet d'éviter les erreurs dues aux valeurs absentes lors de l'analyse.

- **Suppression des doublons** :Les doublons sont identifiés et supprimés sur la base de colonnes clés telles que method, url, status, bytes, referrer, et user_agent. Si des doublons sont trouvés, ils sont enregistrés dans un fichier pour une analyse ultérieure avant d'être supprimés.
- **Conversion des types de données** :Les champs status et bytes sont convertis en entiers, tandis que le champ timestamp est converti au format datetime. Les enregistrements avec des timestamps invalides sont supprimés, ce qui garantit que toutes les données sont dans un format cohérent.
- **Standardisation des horodatages** :Les horodatages sont convertis en UTC pour uniformiser les dates, en particulier lorsque les logs proviennent de serveurs dans des fuseaux horaires différents.
- **Suppression des caractères indésirables** : Les données complexes sont traitées afin de supprimer les caractères indésirables comme <, >, ", et \\x. Cela permet de simplifier et de nettoyer les données dans les champs de requêtes.
- **Suppression des lignes mal formées** :Les lignes qui échouent à la correspondance avec l'expression régulière ou qui contiennent des timestamps invalides sont identifiées et supprimées. Ces lignes sont également enregistrées dans un fichier pour une analyse plus approfondie.

- **Suppression des enregistrements invalides** :Les enregistrements avec des valeurs manquantes dans des colonnes cruciales telles que url, method, et status sont supprimés pour garantir que seules les données valides restent dans le DataFrame.
- **Décodage des URL** : Le protocole HTTP restreint l'usage de certains caractères dans les URLs, nécessitant leur encodage pour assurer une transmission correcte. Cette contrainte est souvent exploitée dans les attaques web, où des charges malveillantes sont dissimulées par un ou plusieurs niveaux d'encodage (obfuscation), rendant leur détection plus difficile. Pour contrer cette technique, une étape de décodage systématique des URLs est intégrée au prétraitement des données. Elle permet de restaurer les chaînes encodées dans une forme lisible, facilitant ainsi l'analyse automatisée et la détection des comportements anormaux.
- **Nettoyage final** : suppression des lignes dont les champs essentiels (url, method, status) restent manquants ou invalides après les étapes précédentes.

➤ *Les logs nettoyés sont ensuite enregistrés et mis à disposition pour les étapes ultérieures de transformation et d'analyse.*

IV.5.1.3. Transformation

L'objectif de cette étape est de transformer les données textuelles (URL, référent, user-agent) en vecteurs numériques exploitables par les modèles d'apprentissage automatique. Pour ce faire, nous avons appliqué des techniques de traitement automatique du langage (NLP), réparties en trois sous-étapes : la tokenisation, la vectorisation des tokens et la normalisation

- **TOKENISATION : La tokenisation consiste à segmenter les champs textuels des logs (comme les URL, user-agent ou referer) en unités lexicales appelées *tokens*, afin de faciliter leur analyse ultérieure.**
- **Encodage / vectorisation :** L'encodage vise à convertir les variables catégorielles en représentations numériques, nécessaires à l'entraînement des modèles.
- **Normalisation :** La normalisation est une étape clé permettant de mettre les caractéristiques numériques sur des échelles comparables. Elle est appliquée après l'encodage, selon la nature des modèles utilisés.

➤ *Étant donné que les techniques mises en œuvre à chacune des étapes de transformation sont spécifiques à chaque modèle, il a été jugé pertinent de documenter ces étapes directement dans la partie décrivant le modèle concerné.*

IV.5.2. Module de détection d'anomalies

Ce module s'appuie sur une approche non supervisée afin d'identifier les comportements inhabituels sans nécessiter de labels préalables. Le principe est de construire un profil du comportement « normal » d'y détecter des écarts significatifs, susceptibles d'indiquer des intrusions ou des dysfonctionnements. Cette démarche est particulièrement pertinente pour repérer des attaques inconnues (y compris de type zero-day), des erreurs logicielles ou des comportements utilisateurs inattendus.

IV.5.2.1. Principe général

La détection d'anomalies non supervisée repose sur l'hypothèse que la majorité des données disponibles correspond à un fonctionnement normal. Le modèle apprend les schémas récurrents au sein de ces données et considère comme anomalies les observations qui s'écartent notablement de ces schémas. Les types d'événements pouvant conduire à des anomalies incluent, notamment :

- Des attaques inédites ou zero-day n'ayant pas été observées auparavant.
- Des bugs ou défaillances logicielles induisant des comportements inhabituels.
- Des usages erratiques ou inappropriés de l'application par des utilisateurs légitimes.

IV.5.2.2. Module de prétraitement 1 et vecteur de caractéristiques 1

IV.5.2.2.1. Extraction des champs pertinents

Les entrées de logs bruts (de la table résultant du parsing) sont analysées pour extraire les attributs utiles à la détection :

- Adresse IP source (src_ip) : L'adresse IP du client effectuant la requête.
- Horodatage (timestamp) : La date et l'heure de la requête.
- Méthode HTTP (method) : La méthode HTTP utilisée (par exemple, GET, POST).
- URL (url) : Le chemin de l'URL demandée et la chaîne de requête.
- Protocole (protocol) : La version du protocole HTTP (par exemple, HTTP/1.1).
- Code de statut (status) : Le code de statut de la réponse HTTP (par exemple, 200, 404).
- Octets (bytes) : Le nombre d'octets transférés dans la réponse.
- Référent (referrer) : L'URL de référence, si présente.
- Agent utilisateur (user_agent) : La chaîne de l'agent utilisateur du client.

IV.5.2.2.2. Enrichissement des caractéristiques

Une fois les champs de base extraits, plusieurs caractéristiques dérivées sont calculées pour enrichir la représentation et capter des signaux pertinents pour la détection d'anomalies.

- Segmentation temporelle et comptage des requêtes :
    - Les horodatages sont regroupés à la minute la plus proche (time_bin) pour regrouper les requêtes par IP et par temps.
    - La caractéristique request_count_per_ip est calculée comme le nombre de requêtes par adresse IP dans chaque intervalle de temps, capturant la fréquence des requêtes.
- Taux de requêtes :
    - Le request_rate est calculé comme la différence de temps (en secondes) entre des requêtes consécutives d'une même IP, triées par horodatage. Les valeurs manquantes sont remplies avec l'écart de temps médian pour gérer la première requête par IP.
- Caractéristiques temporelles :

- La caractéristique request_hour extrait l'heure de la journée à partir de l'horodatage, les valeurs manquantes étant remplies par 12 (midi) comme valeur par défaut neutre**.**

- Caractéristiques du type de requête :
  - La caractéristique is_post est un indicateur binaire (1 si la méthode est POST, 0 sinon).
- Caractéristiques de taille et de complexité :
  - La caractéristique log_bytes applique une transformation logarithmique (log1p) à la colonne bytes pour réduire l'asymétrie, les valeurs manquantes étant remplies par 0.
  - La caractéristique path_length mesure la longueur de la chaîne de l'URL.
  - La caractéristique url_depth calcule le nombre de segments de chemin dans l'URL (par exemple, /path/to/resource a une profondeur de 3), avec une valeur par défaut de 1 pour les URL racines (/) ou invalides.
- Caractéristique de référent :
  - La caractéristique is_referrer_absent est un indicateur binaire (1 si le référent est vide, 0 sinon).
- Caractéristiques liées à la sécurité :
  - La caractéristique is_sensitive_url signale les URL contenant des motifs sensibles (par exemple, admin, wp-login, .env), en utilisant une regex insensible à la casse. Pour les journaux blancs, cette valeur est forcée à 0, supposant que tous sont normaux.
  - La caractéristique is_malicious_param détecte les paramètres de requête potentiellement malveillants (par exemple, motifs d'injection SQL, tentatives XSS) à l'aide d'un motif regex.

IV.5.2.2.3. Transformation des Caractéristiques

La première étape d'un pipeline d'apprentissage automatique consiste en un prétraitement rigoureux des données. Celui-ci englobe plusieurs opérations successives, allant de l'extraction initiale à la structuration des entrées du modèle. Les étapes de parsing et de nettoyage ont été abordées en détail dans les sections précédentes. La présente section se concentre sur la transformation des données, une phase essentielle pour garantir la compatibilité des données avec les algorithmes d'apprentissage automatique et optimiser les performances du système.

IV.5.2.2.4. Tokenization

Une tokenisation légère a été appliquée afin de segmenter les chaînes textuelles tout en préservant leur structure sémantique. Cette opération facilite l'identification automatique de motifs récurrents dans le trafic HTTP, qu'ils soient légitimes ou suspects (paramètres malformés, caractères spéciaux, séquences typiques d'attaques, etc.).

IV.5.2.2.5. Encodage des Variables

Les modèles d'apprentissage automatique nécessitent des représentations numériques des données d'entrée. À cette fin, plusieurs techniques d'encodage ont été mobilisées :

- **Label Encoding** : utilisé pour la variable cible (label), en attribuant 0 aux requêtes normales et 1 aux requêtes anormales.
- **Encodage binaire** : appliqué aux variables booléennes (is_post, is_referrer_absent, is_malformed, etc.), codées en 0 ou 1.
- **Encodage One-Hot (implicite)** : utilisé pour convertir certaines variables catégorielles nominales telles que method ou status en vecteurs binaires, chaque modalité étant représentée par une colonne indépendante.
- **Encodage manuel par expressions régulières** : des motifs spécifiques (liés à des charges utiles malveillantes, mots-clés suspects ou structures anormales) sont

détectés et transformés en variables binaires (1 si le motif est présent, 0 sinon).

IV.5.2.2.6. Normalisation

Les variables numériques présentent souvent des échelles très différentes, ce qui peut fausser l'apprentissage. Pour y remédier, une normalisation adaptée a été appliquée :

1- **Standardisation (StandardScaler) : Cette méthode contre les données autour de 0 et réduit leur variance à 1, ce qui garantit une contribution équitable de chaque caractéristique.**

Avantages :

- Équilibrage des échelles entre caractéristiques.
- Optimisation des modèles sensibles aux distances (KMeans, DBSCAN).
- Convergence plus rapide des modèles basés sur les gradients.
- Réduction des biais dus à l'amplitude des variables.

2- **Transformation logarithmique : Appliquée à certaines variables présentant une forte asymétrie ou des valeurs extrêmes, cette transformation permet de lisser la distribution des données.**

Avantages :

- Réduction de l'impact des valeurs extrêmes.
- Stabilisation des distributions pour une meilleure robustesse.
- Meilleure adaptation aux modèles prédictifs.
- Captation plus fine des relations non linéaires entre variables.

IV.5.2.2.7. CHOIX DES ALGORITHMES NON SUPERVISES

Plusieurs algorithmes complémentaires sont testés afin de couvrir divers types de distributions et formes d'anomalies :

1. **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)** : identifie comme anomalies les points situés dans des régions de faible densité, en regroupant

les points denses en clusters. Adapté pour détecter des groupes d'activités anormales lorsque celles-ci sont peu fréquentes.

2. **K-Means** : partitionne en k clusters. Les observations éloignées des centroïdes sont considérées comme potentiellement anormales. Nécessite un choix préalable de k et peut être sensible à la forme des clusters.
3. **Isolation Forest** : isole rapidement les observations rares via la construction d'arbres aléatoires. Les points anormaux sont isolés en général avec moins de coupures. Efficace pour des données de grande dimension.
4. **One-Class SVM** : apprend une frontière dans l'espace des caractéristiques englobant majoritairement les données normales, en détectant comme anomalies les points en dehors de cette frontière. Sensible au choix du noyau et des hyperparamètres.
5. **Autoencodeur (réseau de neurones)** : reconstruit les entrées à partir d'une couche représentations compressée. Les erreurs de reconstruction élevées signalent des observations qui ne correspondent pas au profil normal appris. Permet de capturer des relations non linéaires, mais requiert un volume de données suffisant et un réglage fin des hyperparamètres.

Ces algorithmes sont choisis pour leur complémentarité : certains sont basés sur la densité, d'autres sur l'isolement, d'autres sur la reconstruction. Cette diversité permet d'adapter la détection à différents types d'anomalies (isolées, groupées, subtiles, structurelles, etc.).

IV.5.2.2.8. REDUCTION DE DIMENSION

Pour atténuer la complexité liée à la forte dimensionnalité et extraire les composantes les plus informatives, nous avons appliqué une Analyse en Composantes Principales (PCA) [Réf.]. Cette méthode projette les données dans un nouvel espace orthogonal, en ordonnant les composants selon la variance expliquée. Elle permet ainsi de conserver l'essentiel de l'information tout en réduisant le nombre de variables, ce qui facilite notamment :

- **La visualisation** : en projetant les observations principales dans un espace bidimensionnel, on peut examiner la structure sous-jacente des données, détecter d'éventuels regroupements ou anomalies.

- **La performance des modèles** : en diminuant la dimension d'entrée, on réduit le risque de surapprentissage, le coût de calcul et la redondance de l'information, tout en conservant les caractéristiques discriminantes.
- **La robustesse du pipeline** : en éliminant les variables peu contributives ou fortement corrélées, on simplifie la représentation des données sans compromettre la qualité de l'analyse.

Le choix du nombre de composantes retenues s'appuie sur le pourcentage de variance expliquée cumulée (par exemple, conserver les composantes jusqu'à atteindre un seuil de 90 % de la variance totale). Les détails de l'implémentation (centrage préalable, calcul des vecteurs propres, critères de sélection des composantes) sont décrits dans le chapitre suivant.

IV.5.2.2.9. Evaluation des modèles

Dans cette phase, des anomalies sont injectées dans le jeu de test pour évaluer le comportement du modèle. Les performances sont mesurées à l'aide des indicateurs suivants:

1- **Précision (Precision) : mesure la proportion de prédictions positives correctes parmi toutes les prédictions positives faites par le modèle.**
2- **Rappel (Recall) : mesure la capacité du modèle à identifier correctement toutes les instances positives.**
3- **F1-Score : est la moyenne harmonique entre la précision et le rappel, utilisée pour évaluer les performances du modèle lorsqu'il y a un déséquilibre entre les classes.**
4- **Exactitude (Accuracy) : mesure la proportion de prédictions correctes parmi toutes les prédictions effectuées par le modèle.**

➤ **La méthode qui obtient les meilleurs résultats globaux, en cherchant un équilibre entre le rappel et la précision pour minimiser les faux positifs et faux négatifs, est sélectionnée pour l'intégration dans le système de détection des anomalies.**

IV.5.3. MODULE DE CLASSIFICATION DES ATTAQUES

IV.5.3.1. Principe général

Ce module a pour objectif d'identifier précisément le type d'attaque web à partir des caractéristiques extraites lors du prétraitement et de la phase d'ingénierie des caractéristiques. Nous adoptons une approche hybride combinant un classifieur multi-classes et plusieurs classifieurs binaires spécialisés, afin de tirer parti de la complémentarité des modèles et d'améliorer à la fois la robustesse et la précision de la détection.

- **Classifieur multi-classes** : permet de distinguer simultanément plusieurs catégories d'attaques en une seule étape.
- **Classifieurs binaires spécialisés** : pour chaque type d'attaque majeur présent dans le jeu de données (SQLi, XSS, LFI, Command Injection), un modèle binaire est entraîné pour détecter spécifiquement cette attaque versus le reste (Non-attaque).

Cette dualité vise à combiner :

- La vue globale offerte par le classifieur multi-classes, utile pour un premier classement rapide entre plusieurs types d'attaques.
- La capacité fine des classifieurs binaires à se spécialiser sur les signatures particulières de chaque attaque, ce qui peut améliorer la détection de variantes subtiles ou peu représentées.

IV.5.3.2 Module de Prétraitement 2 et vecteur de caractéristiques 2

À l'instar du module de détection non supervisée, la préparation rigoureuse des données est une étape préalable indispensable à l'entraînement des modèles supervisés. Cette phase ayant déjà été décrite dans la section dédiée au prétraitement, la section suivante sera consacrée à la présentation des caractéristiques sélectionnées pour l'apprentissage supervisé.

IV.5.3.3. Génération des caractéristiques pour le modèle supervisé multiclasse

Après revue de la littérature en cyber-sécurité et analyse de la nature des attaques web, nous retenons les caractéristiques suivantes, jugées discriminantes :

- **Label (classe d'attaque)** : variable cible indiquant la catégorie (par exemple « SQLi », « XSS », « LFI », « Command Injection », ou « Normal »).
- **status_code** : code HTTP retourné, pouvant révéler des comportements anormaux (erreurs 4xx/5xx fréquentes lors d'attaques).
- **method (GET/POST/…)** : la méthode HTTP peut influencer la vectorisation de l'attaque (ex. payload dans POST).
- **Param_count** : nombre de paramètres de requête ; un nombre élevé ou inhabituel peut indiquer tentatives d'injection.
- **protocol** : version HTTP utilisée, qui peut varier dans certaines attaques automatisées ou obsolètes.
- **Suspect_param** : indicateur binaire ou score signalant la présence de mots-clés ou motifs suspects dans les paramètres (injections SQL, XSS, etc.), fondé sur un lexique ou regex spécialisés.
- **response_size** : taille de la réponse, susceptible de différer pour une requête malveillante (ex. pages d'erreur, contenus dynamiques).
- **user-agent_class** : catégorisation du User-Agent (navigateur classique, robot, scanner, outil automatisé), utile pour repérer des agents non usuels.
- **user-agent_length** : longueur de la chaîne User-Agent, qui peut être atypique dans les scripts malveillants.
- **Dictionnaire de mots suspects** : plus de 125 tokens spécifiques aux payloads d'attaque (par ex. mots-clés SQLi, vecteurs XSS), utilisés pour enrichir les variables textuelles ou calculer des scores de similarité.

IV.5.3.4. Transformation des Caractéristiques

La phase de transformation suit l'extraction des champs pertinents et vise à préparer les données pour l'entraînement des modèles. Elle comporte plusieurs sous-étapes : la tokenisation, l'encodage/vectorisation des variables, puis la normalisation des caractéristiques numériques. Les choix présentés ci-dessous sont adaptés aux spécificités des journaux web et aux exigences des modèles supervisés.

IV.5.3.6. Tokenization

Pour les champs textuels (notamment les URL, les charges utiles et autres chaînes susceptibles de contenir des motifs malveillants), une tokenisation enrichie par dictionnaire a été mise en œuvre :

- **Construction du vocabulaire** : Un dictionnaire fixe d'environ 125 tokens a été élaboré à partir d'une analyse des schémas caractéristiques des principales familles d'attaques web (XSS, SQLi, LFI, CMDi).
- **Types de tokens inclus** : mots-clés spécifiques, opérateurs, séquences spéciales, fragments de paramètres d'URL fréquemment associés à des comportements malveillants.
- **Objectif** : segmenter le texte de manière à conserver les éléments syntaxiques et sémantiques pertinents pour la détection de motifs suspects, tout en limitant la dimensionnalité et en favorisant les performances.

IV.5.3.7. Encodage

Toutes les variables non numériques doivent être converties en représentations numériques adaptées aux modèles de classification supervisée.

Méthode utilisée : Afin de rendre les données exploitables par les algorithmes de classification supervisée, l'ensemble des variables catégorielles, binaires et numériques a été converti en représentations numériques appropriées. Deux principales techniques d'encodage ont été mobilisées selon la nature des données : le *Label Encoding* et le *One-Hot Encoding*.

1- **Label Encoding : Le *Label Encoding* consiste à attribuer un identifiant numérique unique à chaque catégorie d'une variable. Cette méthode a été utilisée notamment pour la variable cible représentant les types d'attaques, en encodant par exemple "SQLi" par 1, "CMDi" par 2, "LFI" par 3 et "XSS" par 4. Cette transformation est indispensable pour entraîner des modèles qui n'acceptent que des entrées numériques.**

Avantages :

- Méthode simple, rapide et économe en mémoire.
- Représentation compacte.
- Bien adaptée aux variables ordinales.
- Compatible avec de nombreux algorithmes tels que les arbres de décision.

2- **One-Hot Encoding : Le One-Hot Encoding permet de représenter des variables catégorielles non ordinales en créant une colonne binaire distincte pour chaque modalité. Par exemple, la variable method avec les valeurs "GET" et "POST" est transformée en deux colonnes : method_GET et method_POST.**

Avantages :

- Supprime toute relation d'ordre arbitraire entre les catégories.
- Représentation explicite, bien interprétée par les modèles linéaires.
- Favorise la performance des modèles tels que les réseaux de neurones ou les régressions.
- Large compatibilité avec les bibliothèques de apprentissage automatique.

Stratégies d'Encodage Appliquées :

L'encodage a été adapté au type de chaque variable :

- **Variables catégorielles** :
  - Label et user_agent_class ont été encodées par *Label Encoding*.

- method et protocol ont été transformées par *One-Hot Encoding* afin d'éviter l'introduction d'une ordinalité artificielle.
- status_code, étant une variable discrète naturellement numérique, a été conservée dans son format d'origine.

- **Variables binaires** :
  - suspect_param a été encodée manuellement : 1 indique la présence d'un ou plusieurs paramètres jugés suspects selon un lexique prédéfini ; 0 indique leur absence.

- **Variables numériques continues** :
  - param_count, response_size et user_agent_length ont été normalisées (via *StandardScaler* ou *Min-Max Scaling*) afin d'uniformiser leur échelle et d'éviter qu'elles n'influencent excessivement l'apprentissage en raison de leur amplitude.

IV.5.3.8. Normalisation des Données

Pour garantir que toutes les caractéristiques numériques soient comprises dans le même intervalle, nous avons adopté la normalisation '' Min-Max Scaling''

Formule :

$$X_{\text{norm}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

Avec

- X : la valeur d'origine (non normalisée)
- Xmin: la valeur minimale de la colonne
- Xmax : la valeur maximale de la colonne

- XnormX : la valeur transformée (normalisée entre 0 et 1)

Usage : particulièrement adapté aux modèles sensibles aux distances (KNN, réseaux de neurones, SVM) et lorsque l'on souhaite conserver la forme initiale de la distribution sans la centrer autour de zéro.

Avantages :

  - Place toutes les caractéristiques dans l'intervalle [0,1], évitant qu'une variable aux valeurs élevées n'écrase les autres.
  - Facilite la convergence des algorithmes de gradient (descente de gradient plus stable).
  - Simple à calculer et à interpréter.

IV.5.3.9.Choix des Algorithmes Supervisés pour la Classification des Attaques

Dans le cadre de la classification supervisée, plusieurs algorithmes ont été sélectionnés afin d'identifier précisément les différents types d'attaques présentes dans notre dataset. Ces algorithmes sont appliqués aussi bien aux classifieurs binaires qu'au classifieur multi-classes. Les méthodes retenues incluent :

- **Random Forest (RF)** : Cet algorithme basé sur un ensemble d'arbres de décision permet de classifier les requêtes selon les caractéristiques extraites. Il est particulièrement efficace pour gérer la diversité des données et offre une bonne robustesse face au bruit.
- **Support Vector Machine (SVM)** : Utilisé pour séparer les classes d'attaques par une frontière optimale dans un espace à haute dimension, SVM est adapté pour distinguer clairement différentes catégories d'attaques complexes.
- **K-Nearest Neighbors (KNN)** : Ce classifieur repose sur la proximité des exemples dans l'espace des features pour attribuer une classe. Il est simple mais souvent performant dans les cas où les classes sont bien séparées.

- **Régression Logistique** : Modèle probabiliste qui estime la probabilité qu'une requête appartienne à une classe d'attaque particulière, utile pour la classification binaire ou multi-classes avec une interprétation facile des résultats.

➤ *Ces algorithmes ont été choisis pour leur capacité à fournir une classification précise et fiable des attaques web, en tenant compte des différentes natures et signatures présentes dans les données.*

IV.5.3.10. Evaluation des modèles

Cette phase vise à évaluer la capacité de généralisation des modèles de classification sur des données inédites, en vérifiant leur aptitude à identifier correctement les différentes formes d'attaques web. Pour ce faire, un jeu de test indépendant est utilisé, distinct de l'ensemble d'entraînement, afin de garantir une évaluation impartiale de la robustesse et de l'efficacité des modèles.

Les performances sont mesurées à l'aide des mêmes métriques que celles appliquées lors de la détection d'anomalies : précision, rappel, F1-score et exactitude globale. Ces indicateurs permettent d'évaluer la qualité de la classification, tant en termes de détection des attaques que de limitation des faux positifs.

Le choix du modèle final repose sur une analyse comparative de ces résultats, en accord avec les priorités opérationnelles du système. Par exemple, dans un contexte de cybersécurité, un rappel élevé pourra être préféré afin de maximiser la détection des menaces, quitte à tolérer une légère diminution de la précision.

IV.5.4 . MODULE DE DECISION HEURISTIQUE

Après le filtrage initial par le module de détection d'anomalies, qui identifie les comportements inhabituels, les enregistrements jugés anormaux sont transmis au système de classification supervisée. Celui-ci mobilise simultanément cinq modèles : quatre classifieurs binaires (chacun dédié à SQLi, XSS, LFI ou Command Injection) et un classifieur multi-classes,

afin d'estimer pour chaque instance anormale les probabilités d'appartenance à une ou plusieurs catégories d'attaques prédéfinies.

IV.5.4.1. Principe général

Le module heuristique, illustré dans la figure, agrège les sorties des modèles supervisés selon une logique en deux phases, conduisant à un état final de classification. Les résultats possibles sont :

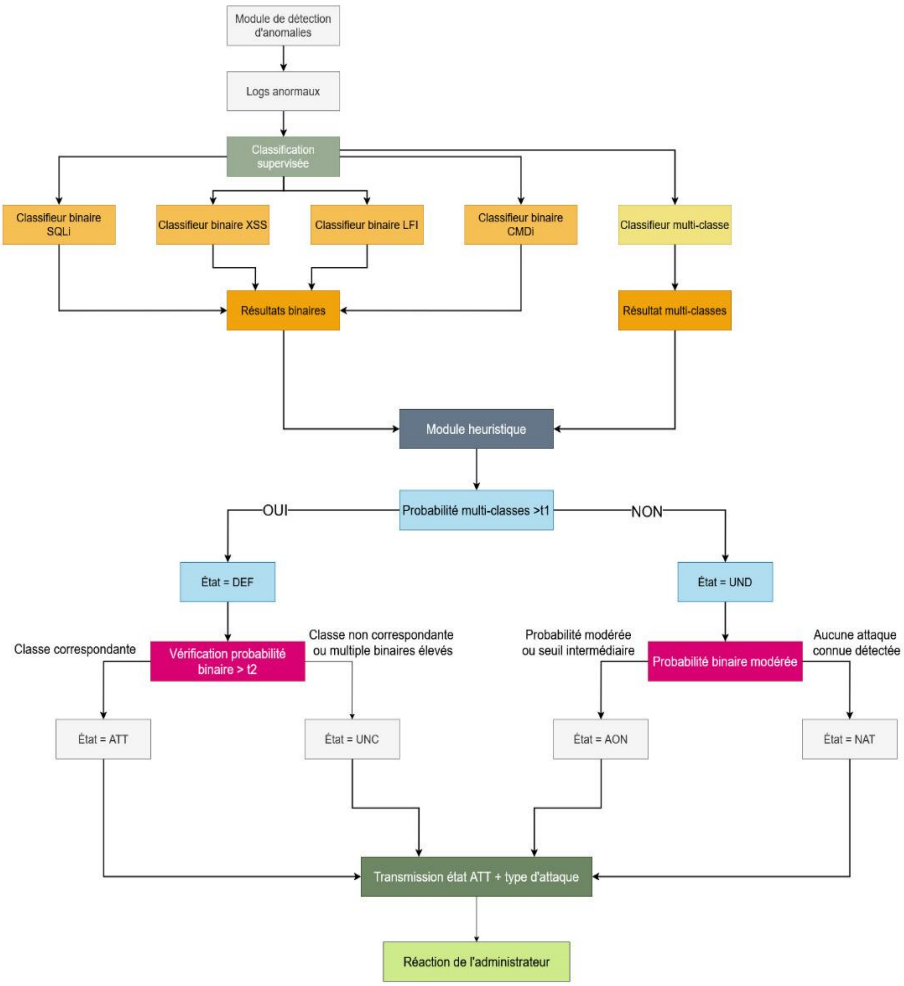


Figure IV.4 : Schéma du module de décision

- **ATT (Attaque Confirmée)** : identification certaine d'une attaque spécifique.
- **UNC (Uncertain-class)** : suspicion forte d'attaque, mais incertitude sur la catégorie exacte.
- **AON (Attack-Or-Not)** : suspicion d'attaque ou anomalie persistante, sans confirmation claire de type.
- **NAT (Non-Attack)** : absence de correspondance avec les classes connues, sans garantie de bénignité absolue.

IV.5.4.2. Principales étapes du module de décision heuristique

IV.5.4.2.1 Évaluation initiale avec le classifieur multi-classes

Le classifieur multi-classes fournit pour chaque instance anormale un vecteur de probabilités sur l'ensemble des catégories. Lorsque la probabilité maximale associée à une classe dépasse un seuil prédéfini t1, l'état temporaire est jugé **Défini** (noté DEF). Dans le cas contraire, cet état est considéré **Indéfini** (UND).

- État **DEF** : la classe la plus probable est retenue provisoirement.
- État **UND** : aucune classe n'atteint la confiance minimale, ce qui signale une incertitude initiale.

IV.5.4.2.2 Analyse des classifieurs binaires spécialisés

Chaque classifieur binaire fournit une probabilité ou un score d'appartenance à son attaque cible. Le module heuristique examine ces scores en regard du seuil t2t_2t2. Plusieurs scénarios peuvent en découler :

1. **Concordance forte avec le multi-classes (vers ATT)**

- Si l'état temporaire issu du multi-classes est DEF pour une classe d'attaque, et que le classifieur binaire correspondant affiche une probabilité supérieure à t2 pour la même classe, l'instance est classée **ATT** pour cette attaque.

2. **Conflits ou incertitudes entre classifieurs binaires et multi-classes (vers UNC)**

    - Présence de scores élevés (> t2) pour plusieurs classifieurs binaires différents, indépendamment de l'état multi-classes DEF ou UND.
    - État multi-classes UND, tandis qu'au moins un classifieur binaire dépasse t2, sans correspondance univoque entre probabilité multi-classes et classifieurs binaires.
    - Toute configuration où plusieurs classes d'attaque sont plausibles simultanément, ou où les modèles binaires et le modèle global ne convergent pas vers la même décision claire.Dans ces cas, l'état final devient **UNC**, reflétant une forte suspicion d'attaque mais une indétermination sur la catégorie précise.

3. **Suspicion modérée (vers AON)**

    - Aucune probabilité binaire n'atteint t2, mais le module non supervisé a signalé l'instance comme anormale, ou les scores binaires sont supérieurs à un seuil intermédiaire (par exemple entre un seuil minimal et t2).
    - Le classifieur multi-classes peut être DEF sur « Normal » ou UND, sans soutien clair d'un classifieur binaire.Cette catégorie **AON** indique qu'une intervention ou une analyse supplémentaire est recommandée, car il subsiste une incertitude notable quant à la nature malveillante ou non de l'événement.

4. **Pas de détection connue (vers NAT)**

    - Aucun classifieur binaire n'atteint un score significatif, et le classifieur multi-classes ne prédit pas une attaque avec confiance (probabilité maximale pour une classe d'attaque < t1 ou prédiction « Normal » au-dessus de t1).
    - Malgré le signal initial d'anomalie, les modèles supervisés ne reconnaissent pas de signature d'attaque connue. L'état **NAT** signifie que l'instance ne correspond à aucune classe d'attaque préalablement apprise. Ce statut ne garantit pas

l'innocuité complète, mais indique l'absence de correspondance avec les menaces connues par le système supervisé.

➤ *Il est important de souligner que l'état NAT ne garantit pas l'innocuité de la requête, mais seulement qu'elle ne correspond à aucune classe connue par le système supervisé. L'état AON traduit, quant à lui, une incertitude plus profonde, nécessitant une attention particulière des administrateurs.*

IV.5.4.2.3. Justification et réglage des seuils

- **Seuil t1** (multi-classes) : fixé pour équilibrer confiance et couverture. Un seuil élevé réduit les faux positifs multi-classes mais peut accroître les cas DEF=UND, donc plus d'UNC ou AON.
- **Seuil t2** (binaires) et **Seuil T3** (binaire modérée) : calibré pour chaque classifieur spécialisé, en s'appuyant sur les performances observées (précision/rappel) lors de la phase de validation.

➤ *Ajustements périodiques des seuils peuvent s'avérer nécessaires selon l'évolution des modèles et du contexte opérationnel.*

IV.5.4.2.4. Transmission du résultat et actions associées

Le module génère, pour chaque instance analysée, un **état final** parmi les suivants : **ATT** (attaque confirmée), **UNC** (incertain), **AON** (anomalie non confirmée), ou **NAT** (activité normale). Lorsqu'une menace est détectée, il peut également attribuer une **classe d'attaque probable** ou un **ensemble de classes plausibles**. L'ensemble des résultats est transmis à l'administrateur, qui peut alors engager l'une des actions suivantes :

1. **Déclenchement de contre-mesures préventives**, en fonction de l'état produit :
    - **ATT** : Activation de mesures spécifiques en fonction de la catégorie d'attaque identifiée. Par exemple, en cas d'injection SQL (SQLi), une restriction d'accès à la base de données peut être appliquée ; pour une attaque XSS, une inspection du contenu ou des scripts peut être lancée.

- **UNC** : Priorisation de l'incident pour une enquête manuelle ou une analyse approfondie. Des mécanismes complémentaires, tels que la corrélation avec d'autres sources de signaux, peuvent être mobilisés afin de réduire l'incertitude.
- **AON** : Génération d'alertes modulées accompagnées d'une demande de vérification ou d'un renforcement de la surveillance. L'administrateur peut opter pour une réponse conservatrice — par exemple, un blocage temporaire dans l'attente d'une confirmation — ou choisir une observation prolongée selon le contexte opérationnel.
- **NAT** : Enregistrement de l'événement dans les journaux à des fins de traçabilité. Bien que l'activité soit considérée comme non malveillante selon les modèles actuels, une accumulation significative d'instances NAT dans des contextes d'anomalie récurrente peut signaler l'apparition de menaces émergentes non encore reconnues.

2. **Archivage des résultats** : Chaque décision est journalisée, accompagnée de l'état final, des scores de classification associés et des caractéristiques pertinentes, afin de permettre une analyse post-incident.
3. **Constitution d'un corpus de rétroaction (feedback)** destiné à l'amélioration continue du système : les données collectées peuvent servir à réentraîner ou ajuster les modèles, notamment par l'intégration de nouvelles instances validées (ex. : attaques confirmées ou faux positifs significatifs).
4. **Réévaluation périodique des seuils de décision** (t_1, t_2) : ces paramètres peuvent être ajustés sur la base des statistiques de performance observées en production, en vue d'optimiser la précision et la réactivité du système.

IV.6. Synthèse et CONCLUSION

Dans ce chapitre, nous avons présenté l'architecture modulaire de notre solution d'analyse avancée des fichiers journaux Web. Cette architecture couvre toutes les étapes essentielles du processus : le prétraitement, l'analyse des logs — incluant d'abord la détection d'anomalies puis la classification des attaques —, et enfin la présentation des résultats aux administrateurs de sécurité.

Nous avons commencé par la **détection non supervisée des anomalies**, visant à identifier les comportements inhabituels potentiellement liés à des attaques inconnues ou à des défaillances systèmes. Ensuite, nous avons exposé notre **modèle supervisé de classification des attaques connues** (telles que SQLi, XSS, LFI), en combinant plusieurs classifieurs binaires et un classifieur multi-classe.

La grande force de notre architecture réside dans sa **modularité**. En effet, les modèles peuvent être remplacés, optimisés ou étendus sans perturber l'architecture globale. Par exemple, il est possible d'ajouter de nouveaux classifieurs binaires pour d'autres types d'attaques (comme DT, RFI, ou CSRF), ou de mettre à jour le classifieur multi-classe pour inclure de nouvelles catégories.

De plus, l'utilisation d'un **module heuristique** permet de combiner les résultats des différents modèles supervisés afin de fournir une décision plus fiable. Cette heuristique attribue à chaque instance l'un des quatre états : attaque confirmée (ATT), classe incertaine (UNC), doute sur la nature même d'une attaque (AON), ou absence d'attaque connue (NAT). Cela offre aux analystes de sécurité une vue enrichie et nuancée, leur permettant d'agir en fonction du degré de certitude et du type de menace.

Enfin, cette architecture permet une **évolutivité et une adaptabilité** qui garantissent la pertinence du système face à l'évolution constante des menaces sur le Web.

Dans le chapitre suivant, nous aborderons les détails de l'implémentation de cette solution ainsi que les résultats obtenus lors des tests réalisés sur différents scénarios d'attaque

CHAPITRE 5 : IMPLEMENTATION ET EVALUATION DES PERFORMANCES

V.1. Introduction

Dans ce chapitre, nous détaillerons l'implémentation des modèles ML utilisés dans ce projet et nous menons plusieurs expérimentations et ce dans l'objectifs de choisir les modèles ML (supervisé et non supervisé) les plus performants pour l'architecture de notre système d'analyse proposé. Après avoir présenté les performances de chaque modèle selon les métriques adoptées pour leur évaluation, nous discuterons les résultats obtenus et nous abordons une analyse comparative détaillée des performances.

Nous commencerons par décrire l'environnement de travail, les outils et les logiciels utilisés tout au long de ce projet. Ensuite, nous présenterons les étapes d'implémentation de chaque module. Après nous décrirons l'environnement d'expérimentation et procéderons à l'évaluation des modèles

V.2. Environnement de développement

V.2.1 Environnement Matériel

Nous avons utilisé deux ordinateurs portables pour le codage mais tous les tests ont

Été effectués sur le serveur Google Colab (Google Colaboratory) qui offre plus de performances matérielles.

Google colab: il est développé par Google Research, c'est 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 nenécessite 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).

Table V.1 Characteristics of the Allocated Colab Environment

Characteristics	Specification
Operating System	Ubuntu 20.04 LTS (Focal Fossa)
RAM	~12.7 GB
CPU	Intel(R) Xeon(R) CPU, 2 cores
GPU	NVIDIA Tesla K80 / T4 (free tier)

V.2.2. Langage de programmation

Nous avons utilisé Python qui est 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 les autres développeurs en raison de sa facilité d'utilisation et de son efficacité.Dans notre cas, il est utilisé pour la création et l'entraînement de modèles de

détection des anomalies et pour la classification des attaques, en traitant et analysant les données de logs web, et en appliquant des techniques d'apprentissage automatique pour identifier les comportements anormaux et classer les attaques.

V.2.3 Bibliothèques

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

Suit :

Pandas : c'est est une bibliothèque Python open-source utilisée pour la manipulation et l'analyse de données. Elle comprend des structures de données et des fonctions permettant d'effectuer des opérations efficaces sur les données. Elle est particulièrement adaptée pour travailler avec des données tabulaires telles que les feuilles de calcul ou les tables SQL. Elle est utilisée en science des données car elle fonctionne bien avec d'autres bibliothèques importantes. Elle est construite sur la bibliothèque NumPy, ce qui facilite la manipulation et l'analyse des données.

Numpy : c'est un package de traitement de tableaux à usage général. Il fournit un objet de tableau multidimensionnel haute performance, ainsi que des outils pour travailler avec ces tableaux. C'est le package fondamental pour le calcul scientifique avec Python.

sklearn (scikit-learn):C'est une bibliothèque open-source d'apprentissage automatique qui prend en charge l'apprentissage supervisé et non supervisé. Elle fournit également divers outils pour l'ajustement des modèles, la prétraitement des données, la sélection des modèles, l'évaluation des modèles et de nombreuses autres utilitaires.

Matplotlib : c'est une bibliothèque de tracé open-source puissante et polyvalente pour Python, conçue pour aider les utilisateurs à visualiser des données sous divers formats. Développée par John D. Hunter en 2003, elle permet aux utilisateurs de représenter graphiquement les données, facilitant ainsi l'analyse et la compréhension.

Urllib.parse: ce module définit une interface standard pour diviser les chaînes d'Uniform Resource Locator (URL) en composants (schéma d'adressage, emplacement réseau, chemin, etc.), pour combiner les composants à nouveau en une chaîne URL, et pour convertir une "URL relative" en une URL absolue donnée une "URL de base".

REGEX: (re) : Une expression régulière ou RegEx est une séquence spéciale de caractères qui utilise un modèle de recherche pour trouver une chaîne de caractères ou un ensemble de chaînes. Elle peut détecter la présence ou l'absence d'un texte en le comparant avec un modèle particulier et peut également diviser un modèle en un ou plusieurs sous-modèles**.** Python dispose d'un module intégré nommé "re" qui est utilisé pour les expressions régulières en Python.

Kneed : il s'agit d'un package Python open-source conçu pour identifier le point de "genou" ou "coude" dans une courbe ajustée aux données. Ce point représente l'emplacement de la courbure maximale, souvent utilisé pour déterminer le nombre optimal de clusters dans des algorithmes de clustering tels que K-Means []

tensorflow (Keras): l'API de haut niveau de la plateforme TensorFlow. Elle fournit une interface accessible et très productive pour résoudre des problèmes d'apprentissage automatique (ML), avec un accent particulier sur l'apprentissage profond moderne. Keras couvre chaque étape du flux de travail de l'apprentissage automatique, de la préparation des données à l'ajustement des hyperparamètres jusqu'au déploiement. Elle a été développée avec un accent particulier sur la possibilité d'expérimenter rapidement.[]

V.3. Les principales étapes d'implémentation

Dans cette partie, nous allons expliquer comment nous avons implémenté les principaux modules de notre solution. Nous allons détailler les étapes de traitement des données, les choix méthodologiques, et les résultats obtenus.

V.3.1. Module de prétraitement

Le prétraitement constitue une phase essentielle dans notre pipeline d'analyse, visant à transformer les journaux bruts en données structurées exploitables par les modèles d'apprentissage automatique. Il englobe le parsing des fichiers, le nettoyage des données, l'extraction des caractéristiques pertinentes et leur normalisation. Ces opérations permettent de garantir la qualité des données en entrée et de maximiser l'efficacité des modules de détection et de classification.

V.3.1.1. Parking et nettoyage des journaux

Les fichiers journaux ont été analysés à l'aide d'un script Python conçu sur mesure. Ce script, basé sur des expressions régulières robustes, permet d'extraire les champs pertinents tout en traitant les cas d'erreur et d'incohérence. Il assure la normalisation des données, l'uniformisation des formats, et la gestion des lignes incomplètes. Le résultat est un ensemble de fichiers CSV structurés, contenant les attributs suivants :

Journaux de trafic blanc :['src_ip', 'timestamp', 'method', 'url', 'protocol', 'status', 'bytes', 'referrer', 'user_agent']

Journaux de test : ['src_ip', 'timestamp', 'method', 'url', 'protocol', 'status', 'bytes', 'referrer', 'user_agent', 'label', 'parse_failed', 'raw_line', 'original_label']

V.3.1.2. Extraction et traitement des caractéristiques pour la détection d'anomalies

L'objectif de cette étape est de produire des représentations numériques capables de capturer les comportements anormaux sans supervision. Un sous-ensemble de caractéristiques a été

sélectionné sur la base de leur pouvoir discriminant vis-à-vis des requêtes suspectes. Les attributs retenus sont :

['request_count_per_ip', 'request_rate', 'request_hour', 'is_post', 'is_malformed',

'log_bytes', 'path_length', 'url_depth', 'is_sensitive_url', 'is_malicious_param',

'is_referrer_absent']

Ces caractéristiques reflètent la fréquence d'émission des requêtes, la complexité des chemins d'URL, l'absence de référents, ou encore la présence de motifs suspects dans les paramètres (ex. javascript : union select). Leur extraction repose sur des fonctions personnalisées telles que :

- **compute_url_depth** : calcule le nombre de segments dans une URL ;
- **extract_query_params** : isole les paramètres pour inspection ;
- **Détection par expressions régulières** : identifie les URL ou paramètres sensibles.

Les variables numériques ont été standardisées à l'aide de StandardScaler, tandis que les variables binaires ont été codées en 0 ou 1. Des stratégies complémentaires ont été appliquées pour optimiser la qualité des données :

- Regroupement temporel des requêtes par minute (time_bin) ;
- Imputation des valeurs manquantes (request_hour = 12, log_bytes = 0) ;
- Normalisation sélective des colonnes continues uniquement.

Les jeux de caractéristiques ainsi obtenus ont été sauvegardés sous forme de fichiers CSV, pour les deux contextes d'expérimentation : trafic légitime et trafic mixte (anomalies injectées).

V.3.1.3. Sélection et traitement des caractéristiques pour la classification des attaques

Pour le module de classification supervisée, les caractéristiques ont été choisies afin de maximiser la séparation entre les différentes classes d'attaques. Les variables sélectionnées sont :

- ['label', 'user-agent_class', 'method', 'protocol', 'status_code',
- 'param_count', 'response_size', 'user-agent_length', 'suspect_param']

Elles permettent de capturer la nature de la requête, la complexité de ses paramètres, ou encore le comportement du serveur en réponse (ex. status_code). Un pipeline de traitement a été mis en place, comprenant :

- **Encodage catégoriel** :
    - Label, user-agent_class: encodage par **Label Encoding**;
    - method, protocol : **One-Hot Encoding** pour éviter les biais d'ordre ;
- **Normalisation** :
    - param_count, response_size, user-agent_length : standardisation des valeurs ;
- **Transformation binaire** :
    - suspect_param : converti en indicateur booléen (1 si motif suspect détecté) ;
- **Conservation directe** :

  status_code : utilisé tel quel, car déjà discrétisé et informatif.

V.3.1.4. Configurations expérimentées

Afin d'optimiser les performances des modèles de classification, plusieurs configurations ont été testées :

- **Adaptation dynamique des motifs malveillants**, selon qu'il s'agisse de classifieurs binaires (attaque spécifique) ou multiclasses (toutes attaques) ;
- **Comparaison des stratégies d'encodage** (Label vs One-Hot Encoding) sur les variables catégorielles ;
- **Application sélective de la normalisation**, restreinte aux seules variables numériques pour éviter de déformer les variables binaires et catégorielles.

Cette préparation rigoureuse garantit une base cohérente et robuste pour l'entraînement et l'évaluation des modèles d'intelligence artificielle.

Un extrait de cette matrice est présenté dans la table V.2.

Table V.2 : Les caractéristiques après prétraitement

Status	Size	suspect_params	param_count	UserAgent_class	UserAgent_length	get	post	http/1.1	Label_mapped	...	cat	head	tail	ls	whoami	uname	id	env	read	eval
200	0.000499	1	0.500000	0	0.971698	1.0	0.0	1.0	1	...	0	0	0	1	0	0	0	0	0	0
200	0.297455	0	0.166667	0	1.000000	1.0	0.0	1.0	3	...	0	0	0	0	0	0	0	0	0	0
200	0.517718	1	0.166667	0	1.000000	1.0	0.0	1.0	4	...	0	0	0	0	0	0	0	0	0	0
200	0.000832	1	0.500000	2	0.273585	1.0	0.0	1.0	1	...	1	0	0	0	0	0	0	0	0	0
200	0.297455	1	0.166667	0	0.943396	1.0	0.0	1.0	3	...	0	0	0	0	0	0	0	0	0	0

V.4. Environnement d'expérimentation

V.4.1. CERIST DATASET (2023)

Les jeux de données constituent la pierre angulaire de tout projet d'apprentissage automatique. Ils représentent la matière première essentielle à l'entraînement, à l'évaluation et à la validation des modèles. La qualité, la diversité et l'annotation précise des données influencent directement la performance et la robustesse des algorithmes développés.

Dans le cadre de notre travail, deux types de jeux de données étaient requis :

- Un jeu de données étiqueté, nécessaire à l'entraînement et à la validation des modèles supervisés de classification des attaques ;
- Un jeu de logs web valides, destiné à l'apprentissage non supervisé pour la détection d'anomalies.

Or, dans la littérature et les travaux connexes, un obstacle récurrent réside dans la disponibilité de jeux de données qui soient à la fois publics, récents, diversifiés et correctement annotés. Face à ces contraintes, nous avons opté pour l'utilisation du jeu de données du CEntre de Recherche sur l'Information Scientifique et Technique CERIST-dataset2023, qui présente plusieurs atouts notables :

- Il est récemment constitué et régulièrement mis à jour, ce qui garantit une meilleure représentativité des menaces actuelles.
- Il provient de plusieurs serveurs web (Apache, Nginx, Tomcat, Python HTTP Server), assurant une diversité du trafic en termes de structure et de contenu.
- Il contient une large gamme d'attaques réelles et leurs variantes, parmi lesquelles :
    - SQL Injection : Blind, Union-based, Time-based, Error-based, Second-order SQLi
    - Cross-Site Scripting (XSS) : Reflected, Stored, DOM-based
    - Inclusions de fichiers : LFI/RFI
    - Injection de commandes système

V.4.2. PRÉPARATION ET STRUCTURATION DU JEU DE DONNÉES

Afin de répondre aux objectifs de notre problématique, nous avons préparé deux versions distinctes du jeu de données à partir du **CERIST dataset 2023**, dans le but d'entraîner respectivement les modules de détection d'anomalies et de classification supervisée.

V.4.2.1 JEU DE DONNÉES NON ÉTIQUETÉ (TRAFIC LÉGITIME)

Le premier sous-ensemble, appelé jeu de données non étiqueté ou *White Traffic*, contient 54 036 entrées représentant exclusivement des requêtes légitimes. Il a été utilisé pour entraîner le module de détection d'anomalies, basé sur un apprentissage non supervisé. Ce jeu permet de caractériser les comportements normaux des utilisateurs, servant ainsi de référence pour détecter toute déviation potentiellement malveillante.

V.4.2.2. JEU DE DONNÉES ÉTIQUETÉ (TRAFIC MIXTE)

Le second sous-ensemble est un jeu de données étiqueté, constitué à la fois de requêtes malveillantes (attaques) et de trafic légitime. Il a été structuré selon les étapes suivantes :

- Séparation initiale des attaques par type :
    - XSS (Cross-Site Scripting)
    - LFI (Local File Inclusion)
    - Command Injection
    - SQL Injection

- **Partitionnement des attaques** :
    - **70 %** des requêtes malveillantes ont été utilisées pour entraîner le modèle de classification supervisée.
    - **30 %** restantes ont été fusionnées avec un échantillon de trafic légitime pour constituer l'ensemble de test, servant à évaluer la capacité du système à distinguer entre trafic normal et attaques.

V.4.2.3. ANALYSE STATISTIQUE INITIALE

Avant toute division, une analyse statistique préliminaire a été menée sur l'ensemble des données. Elle visait à:

- Évaluer la taille totale du corpus.
- Examiner la répartition des types d'attaques.
- Quantifier la proportion de trafic légitime.

Ces statistiques, illustrées dans la figure V.1, ont permis d'orienter le découpage des données de manière équilibrée et représentative.

	Type de trafic	Nombre de lignes
0	SQL Injection	31130
1	LFI	30961
2	Command Injection	6072
3	XSS	8524
4	White Traffic	54036
5	Total	130723

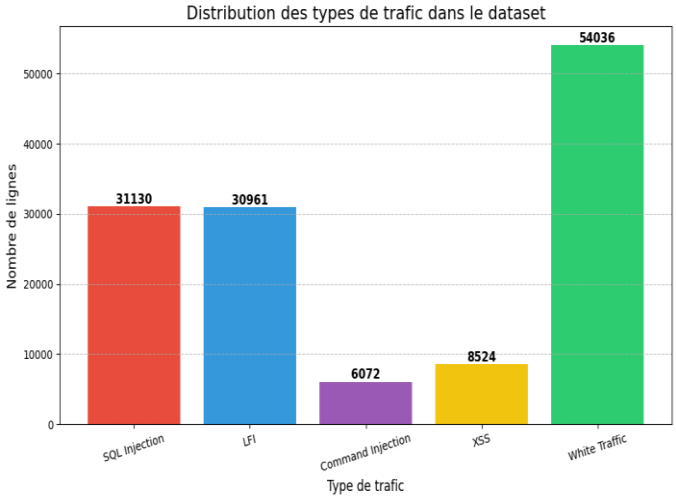


Figure V.2 : Statistiques de CERIST dataset2023 brut

V.4.2.4. NETTOYAGE ET TRAITEMENT DES DONNÉES

Une fois les sous-ensembles constitués, un nettoyage rigoureux a été effectué, citons notamment :

- Élimination des doublons
- Correction des erreurs de format
- Décodage des chaînes d'URL, afin de faciliter l'analyse des paramètres textuels dans les logs.

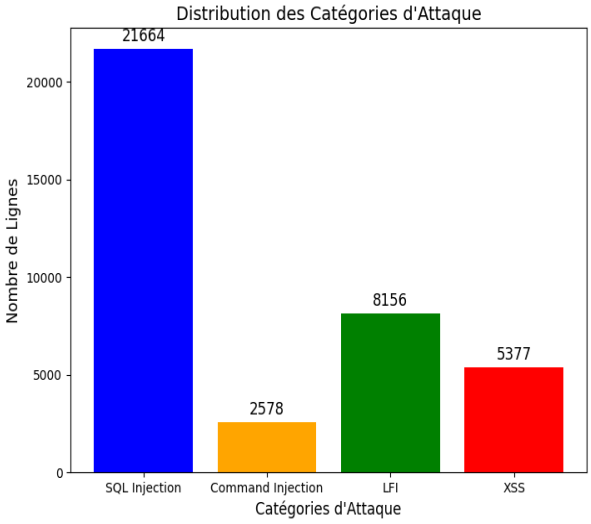
Ces étapes visent à garantir la qualité et la cohérence des données exploitées par les modules d'apprentissage.

➤ *Les détails opérationnels de ces étapes sont développés dans les sections suivantes.*

V.4.2.5. STATISTIQUES POST-TRAITEMENT

Après nettoyage, une seconde analyse statistique, présentée dans la figure V.3. a permis d'établir :

- Le nombre de lignes par catégorie (requêtes malveillantes vs. légitimes)
- La répartition des attaques entre les ensembles d'apprentissage et de test
- L'équilibre des classes, indispensable pour assurer la robustesse de l'évaluation des modèles



Catégorie	Nombre de lignes
SQL Injection	21664
Command Injection	2578
LFI	8156
XSS	5377
Total	37775

Figure V.2 : Distribution des classes d'attaques Web après nétoyage de CERIST dataset2023

V.4.3. Entraînement des modèles non supervisés

V.4.3.1. Constitution des ensembles d'entraînement et de test

On considère une partition des données « normales » :

- **Entraînement** : L'ensemble de données **« white traffic »,** composé uniquement de données normales, est utilisé pour entraîner le modèle à reconnaître le comportement normal du système,
- **Test** : Pour les tests, 30 % des attaques extraites du jeu de données « **TRAFIC MIXTE**», contenant à la fois des données normales et des attaques, sont utilisées afin d'évaluer la capacité du modèle à détecter les anomalies en présence de trafic normal et malveillant.

V.4.4. Entraînement des Modèles supervisés

V.4.4.1. Préparation du Dataset

Pour exploiter l'approche hybride, nous construisons plusieurs jeux de données supervisés distincts :

1. **Classifieurs binaires** (quatre jeux séparés) :
    - SQLi vs. non-SQLi,
    - XSS vs. non-XSS,
    - LFI vs. non-LFI,
    - Command Injection vs. Non-Command Injection.

2. **Classifieur multi-classes** : regroupe simultanément les classes d'attaque (SQLi, XSS, LFI, Command Injection).

Chaque version est divisée selon la répartition suivante :

- **Données d'entraînement (70 %)** : utilisées pour entraîner les modèles.
- **Données de test (30 %)** : utilisées pour évaluer la performance des modèles sur des données non vues.

V.4.5. Module de Detection d’anomalie

Le module de détection d’anomalie repose sur l’utilisation d’algorithmes non supervisés, conçus pour identifier les comportements anormaux sans recourir à des données étiquetées. Cinq modèles ont été testés : **K-Means**, **DBSCAN**, **Isolation Forest**, **One-Class SVM** et **Autoencodeurs**. Cette section présente les expérimentations menées ainsi que les résultats obtenus. Le modèle ayant démontré les meilleures performances a été retenu pour intégration dans la solution finale.

V.4.5.1. Apprentissage et test des modèles non supervisés

V.4.5.1.1 KMEANS

Après avoir testé plusieurs paramètres pour la méthode K-means et comparé les résultats obtenus, les meilleurs résultats ont été obtenus avec les paramètres suivants :

- n_clusters = 2 : Le nombre optimal de clusters a été déterminé à l'aide de la méthode du coude, qui identifie la valeur de k correspondant à la plus grande diminution de la somme des carrés intra-clusters (WCSS). Le nombre optimal de clusters a été fixé à 2, ce qui a permis d'obtenir les meilleures performances en termes de séparation des données.

- Seuil de taille de cluster = 5% : Les clusters contenant moins de 5 % des données d'entraînement ont été identifiés comme des anomalies.

- Seuil de distance = 2.5309 : Le seuil de distance a été défini au 90e percentile des distances aux centres des clusters dans les données d'entraînement.

Les anomalies ont été détectées dans les données d'entraînement en appliquant K-means avec le k optimal, puis en étiquetant les points situés dans les clusters contenant moins de 5 % des données comme des outliers. Les données d'entraînement nettoyées ont ensuite été utilisées pour entraîner un modèle K-means affiné. Pour les tests, les points ont été classés

comme normaux ou anormaux en fonction de leur distance aux centres des clusters, en utilisant le seuil du 90e percentile. Les métriques d'évaluation, y compris la précision, le rappel et la mesure F, sont présentées ci-dessous.la figure :

```
Confusion Matrix:
 [[ 8212  2783]
 [ 1718 21122]]

Classification Report:
              precision    recall  f1-score   support

      Normal       0.83      0.75      0.78     10995
     Anomaly       0.88      0.92      0.90     22840

    accuracy                           0.87     33835
   macro avg       0.86      0.84      0.84     33835
weighted avg       0.87      0.87      0.87     33835
```

Figure V.3 : Performances Kmeans

V.4.5.1.2. DBSCAN

À la suite de multiples expérimentations sur les paramètres pour l'algorithme DBSCAN et comparaison des résultats, les meilleurs résultats ont été obtenus avec les paramètres suivants :

- **eps = 1.1600** : Le meilleur paramètre eps a été estimé à l'aide de la méthode du k-distance (méthode du coude).
- **min_samples = 2** : Ce paramètre définit le nombre minimal de points nécessaires pour qu'un cluster soit considéré comme valide.
- Après l'entraînement du modèle DBSCAN, les points étiquetés comme -1 par l'algorithme ont été considérés comme des anomalies. L'évaluation a été effectuée à l'aide de la matrice de confusion et du rapport de classification.

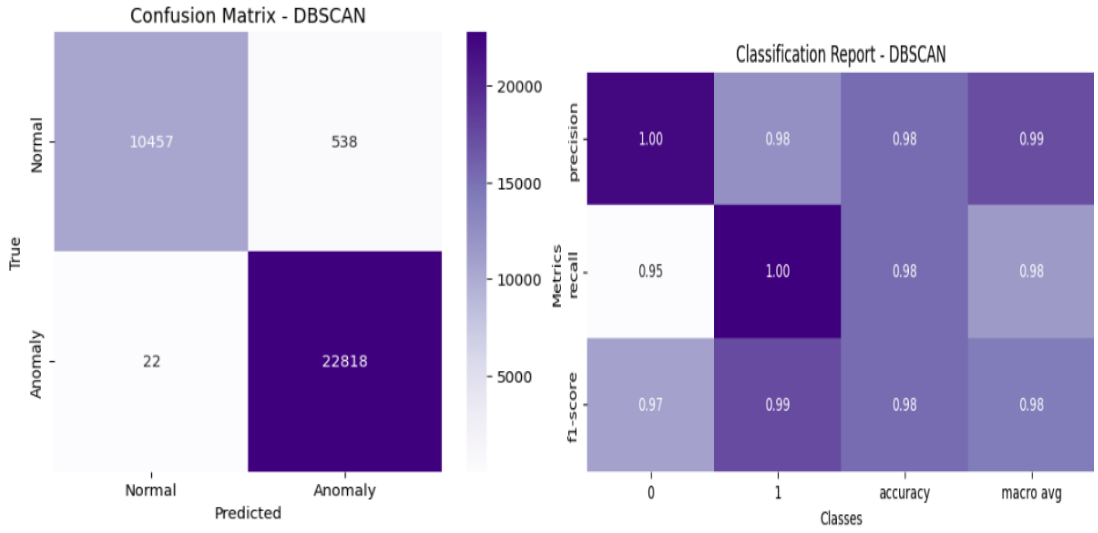


Figure V.4 : Performances DBSCAN

V.4.5.1.3 ISOLATION FOREST

Après expérimentation de diverses valeurs de paramètres pour l'algorithme Isolation Forest, et avoir comparé les résultats de chacun, les meilleurs résultats ont été obtenus avec les paramètres suivants :

- contamination = 0.0001
- n_estimators = 200
- seuil = -0.1337

Les résultats obtenus à l'aide de ces paramètres sont illustrés dans la matrice de confusion ci-dessous

```
Confusion Matrix:
[[10752   243]
 [   10 22830]]

Classification Report:
              precision    recall  f1-score   support

      Normal       1.00      0.98      0.99     10995
     Anomaly       0.99      1.00      0.99     22840

    accuracy                           0.99     33835
   macro avg       0.99      0.99      0.99     33835
weighted avg       0.99      0.99      0.99     33835
```

Figure V.5 : Performances ISOLATION FOREST

V.4.5.1.4.ONE-CLASS SVM

Une fois plusieurs ajustements paramétriques effectués, pour l'algorithme One-Class SVM, les meilleurs résultats ont été obtenus avec les paramètres suivants :

- nu = 0.00001
- gamma = 0.001
- kernel = rbf

Les résultats obtenus à l'aide de ces paramètres sont illustrés ci-dessous.

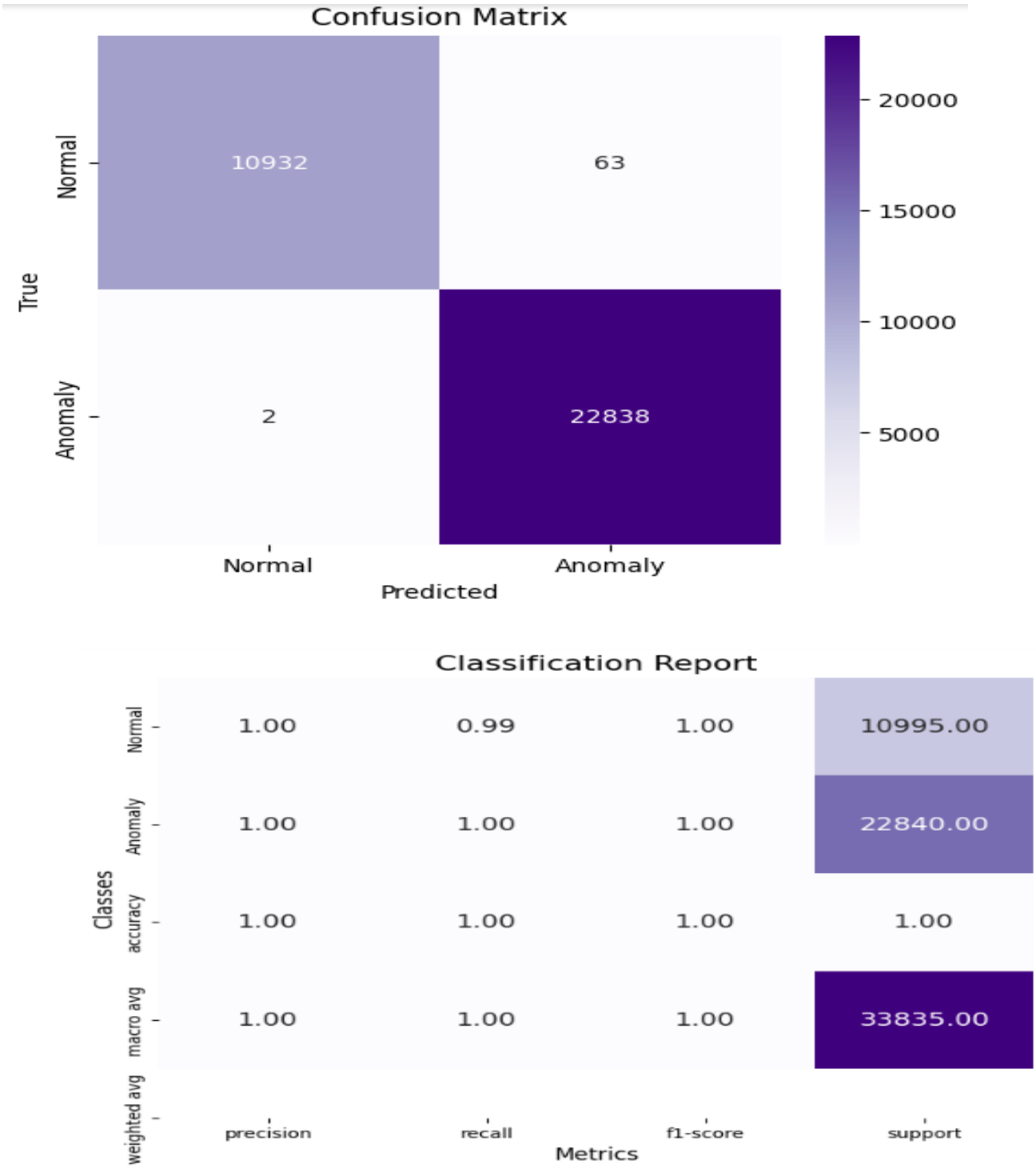


Figure V.5: Performances ONE CLASS SVM

V.4.5.1.5 AUTOENCODERS

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 [] :

(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 reconstruction résultante avec le seuil :

Si erreur < seuil : Valide

Sinon erreur > seuil : Anomalie

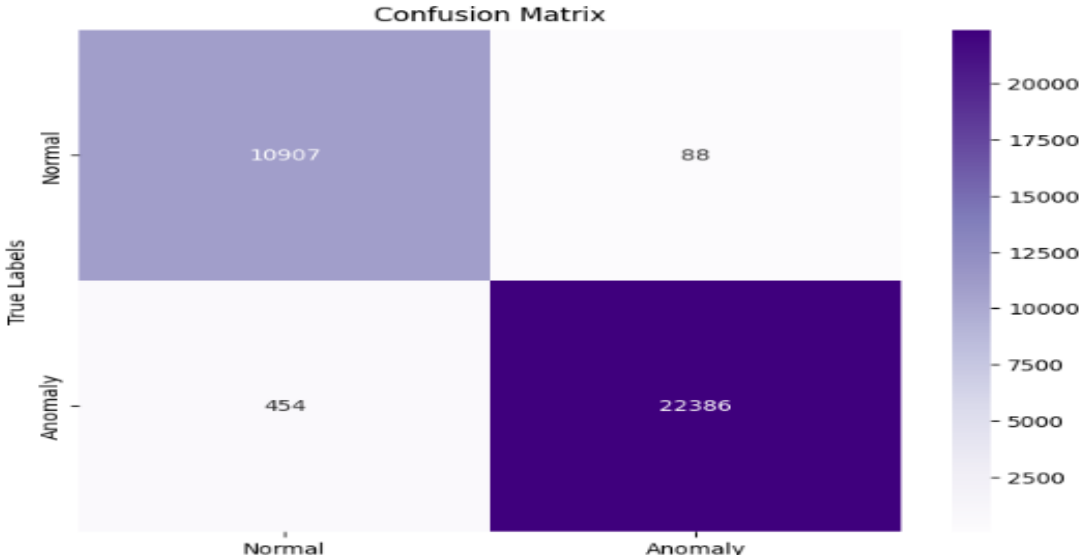
Nous avons testé plusieurs paramètres de cet algorithme tels que : le nombre de couches, le nombre de neurones par couche, la fonction d'activation...etc. Le tableau

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

Table V.3 : Les meilleurs paramètres pour autoencoder

Threshold	0.2018
Epochs	50
Optimizer	Adam
Loss Function	MSE (Mean Squared Error)

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)



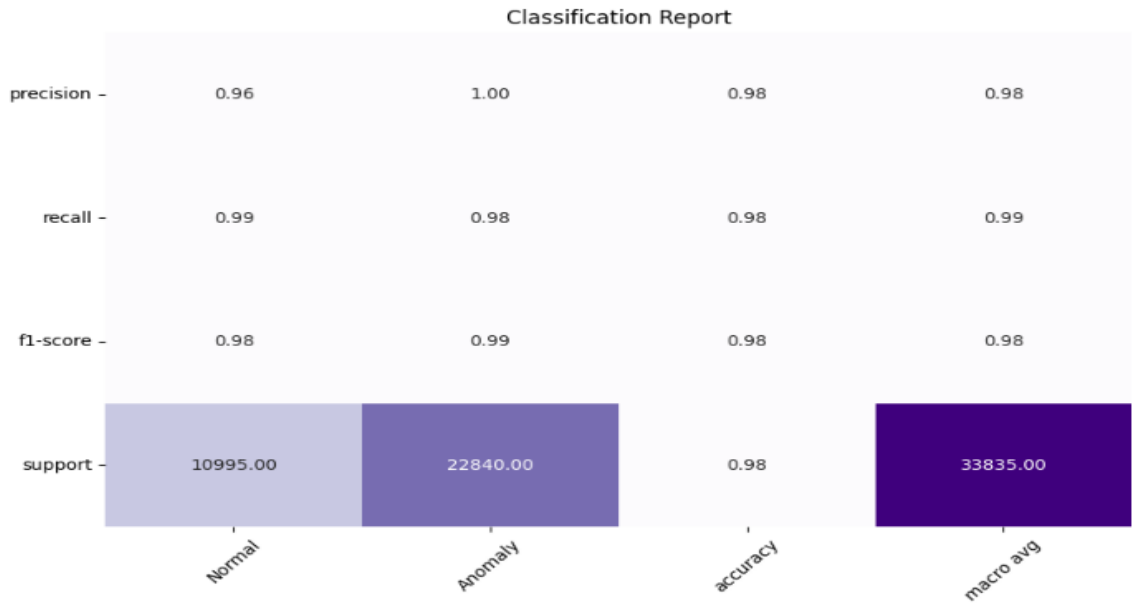


Figure V.6 : Performances AUTOENCODER

V.4.5.2. Etude comparative des modèles non supervisés expérimentés

Les différents modèles de détection d'anomalies sélectionnés ont été évalués à l'aide d'un même ensemble de tests, afin de garantir une comparaison équitable de leurs performances. Les tableaux suivants présentent une synthèse des résultats obtenus pour chacun des modèles.

Tableau V.4 : Comparaison des performances

Algorithm	General Precision	General Recall	General F1-Score	Accuracy
KMeans	0.86	0.84	0.81	0.87
DBSCAN	0.99	0.98	0.98	0.98
Isolation Forest	0.99	0.99	0.99	0.99
OCSVM	1.00	1.00	1.00	1.00
Autoencoders	0.98	0.99	0.98	0.98

Algorithm	Error Rate	Detection Rate	Precision
KMeans	2783/33835 (8.24%)	21122/22840 (92.46%)	88%
DBSCAN	538/33835 (1.59%)	22818/22840 (99.90%)	98%
Isolation Forest	243/33835 (0.72%)	22830/22840 (99.95%)	99%
OCSVM	63/33835 (0.19%)	22838/22840 (100%)	100%
Autoencoders	88/33835 (0.26%)	22386/22840 (98.03%)	98%

Discussion

Les résultats expérimentaux comparant les cinq modèles de détection d'anomalies montrent des performances globalement élevées pour les approches basées sur l'apprentissage non supervisé, avec des écarts significatifs selon les algorithmes. Le modèle **K-Means** affiche les résultats les plus faibles, avec une précision de 86 %, un F1-score de 0,81, et un taux d'erreur de 8,24 %, ce qui le rend moins adapté pour des contextes sensibles à la détection fine. À l'inverse, les modèles **Isolation Forest**, **One-Class SVM (OCSVM)** et **Autoencodeurs** se distinguent par des performances remarquables. **OCSVM** obtient un score parfait (1.00) pour la précision, le rappel, le F1-score et l'exactitude, tout en affichant le taux d'erreur le plus faible (0,19 %) et un taux de détection de 100 %. **Isolation Forest** et les **Autoencodeurs** offrent également une excellente précision (99 % et 98 %) et un taux de détection supérieur à 98 %. Ces résultats justifient le choix d'intégrer les modèles les plus performants — notamment OCSVM — dans la solution finale, afin d'assurer une détection d'anomalies fiable et robuste.

V.4.6. Module de Classification

La classification joue un rôle central dans l'identification de la nature des anomalies détectées dans les journaux web. À partir des entrées marquées comme suspectes lors de la détection d'anomalies, ce module attribue une catégorie d'attaque spécifique, telle que les injections SQL, les accès non autorisés ou les tentatives d'exploitation. L'objectif est de caractériser précisément les comportements malveillants pour renforcer les mesures de sécurité et faciliter les investigations des incidents.

V.4.6.1 Apprentissage et test des algorithmes supervisés

Dans cette section, nous présentons les tests réalisés sur les modèles ML supervisés, en distinguant d'une part le classifieur multiclasse, et d'autre part les classifieurs binaires. Les algorithmes évalués incluent SVM, K-Nearest Neighbors (KNN), régression logistique et Random Forest. Une étude comparative des résultats finaux est ensuite proposée afin d'identifier les approches les plus performantes.

V.4.6.1.1. SVM

SVM est un algorithme qui cherche à séparer les classes par une frontière avec la plus grande marge possible.

- kernel = 'rbf' : noyau radial utilisé pour gérer les données non linéaires.
- C = 10 : contrôle la régularisation (plus grand = moins d'erreurs autorisées).
- gamma = 'scale' : influence d'un point d'entraînement.
- probability = True : permet d'obtenir des probabilités de prédiction.
- random_state = 42 : pour garantir des résultats reproductibles.

a. Classifieur Multiclasse

Accuracy sur le dataset de test : 0.9744

Rapport de classification :

Tableau V.5 : Rapport de classification SVM Multiclasse

Rapport de classification :

Classe	Précision	Rappel	F1-score	Support
-1	0	0	0	34
1	0.998824	0.98889	0.993832	9451
2	0.998084	0.852003	0.919277	1223
3	0.913281	0.999779	0.954574	4519
4	0.996009	0.945286	0.969985	2376
Macro avg	0.78124	0.757192	0.767534	17603
Weighted avg	0.974503	0.974379	0.973436	17603

Matrice de confusion

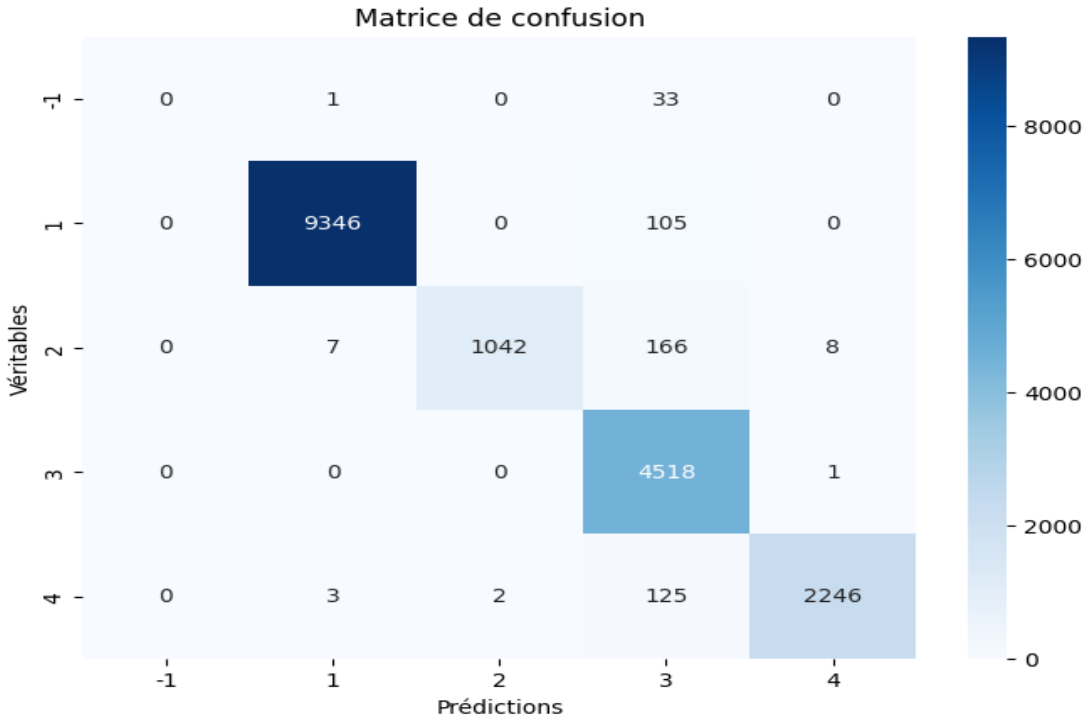


Figure V.7 : Matrice de confusion SVM MULTICLASSE

b. Classifieurs Binaires

Classifieur binaire SQLi

Accuracy pour SQLi : 0.9841

Rapport de classification :

Tableau V.6 : Rapport de classification SVM Binaire SQLi

	precision	recall	f1-score	support
0	0.969	0.998	0.983	8152
1	0.998	0.972	0.985	9451
accuracy			0.984	17603
macro avg	0.983	0.985	0.984	17603
weighted avg	0.984	0.984	0.984	17603

Matrice de confusion :

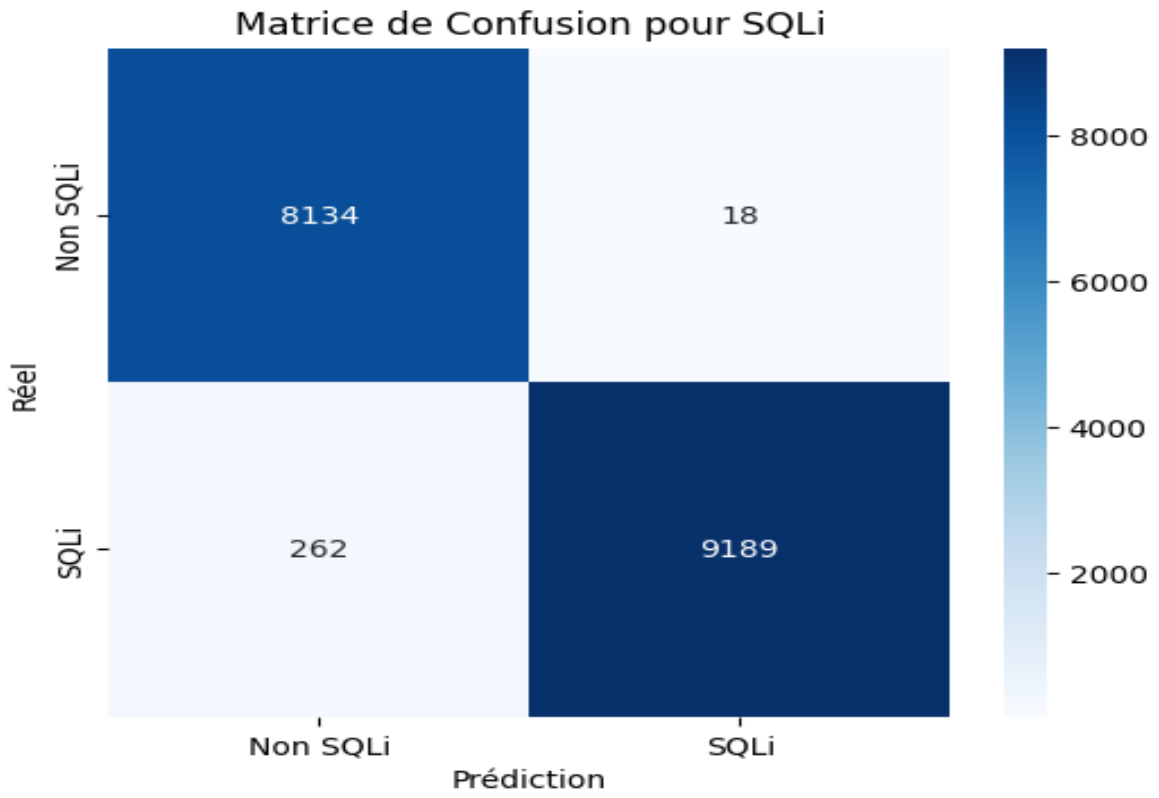


Figure V.8 : Matrice de confusion SVM BINAIRE SQLi

Classifieur binaire CMDi

Accuracy pour CMDi : 0.9896

Rapport de classification :

Tableau V.7 : Rapport de classification SVM Binaire CMDi

Rapport de classification pour CMDi :

	precision	recall	f1-score	support
0	0.989	1.00	0.994	16380
1	1.00	0.85	0.919	1223
accuracy			0.99	17603
macro avg	0.994	0.925	0.957	17603
weighted avg	0.99	0.99	0.989	17603

Matrice de confusion :

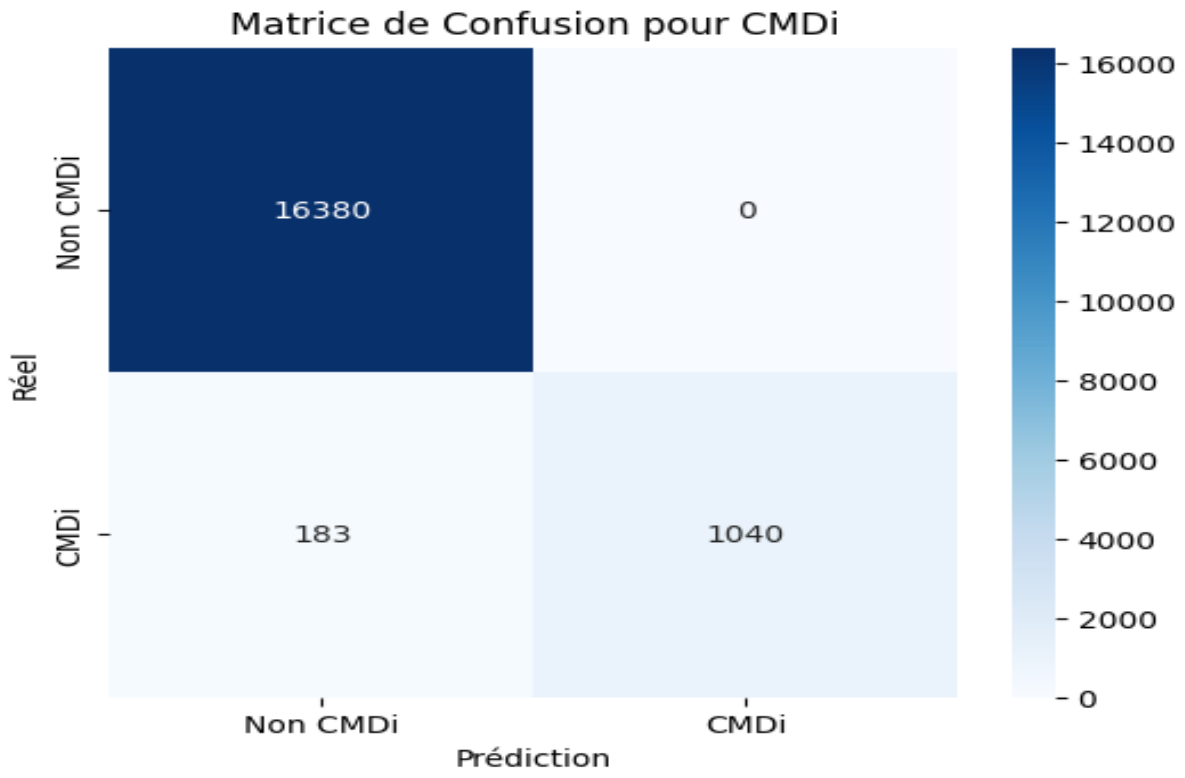


Figure V.9 : Matrice de confusion SVM BINAIRE CMDI

Classifieur binaire LFI

Accuracy pour LFI : 0.8758

Rapport de classification :

Tableau V.8 : Rapport de classification SVM Binaire LFI

Rapport de classification pour LFI :

	precision	recall	f1-score	support
0	0.861	0.994	0.922	13084
1	0.966	0.535	0.689	4519
accuracy			0.876	17603
macro avg	0.913	0.764	0.806	17603
weighted avg	0.888	0.876	0.862	17603

Matrice de confusion :

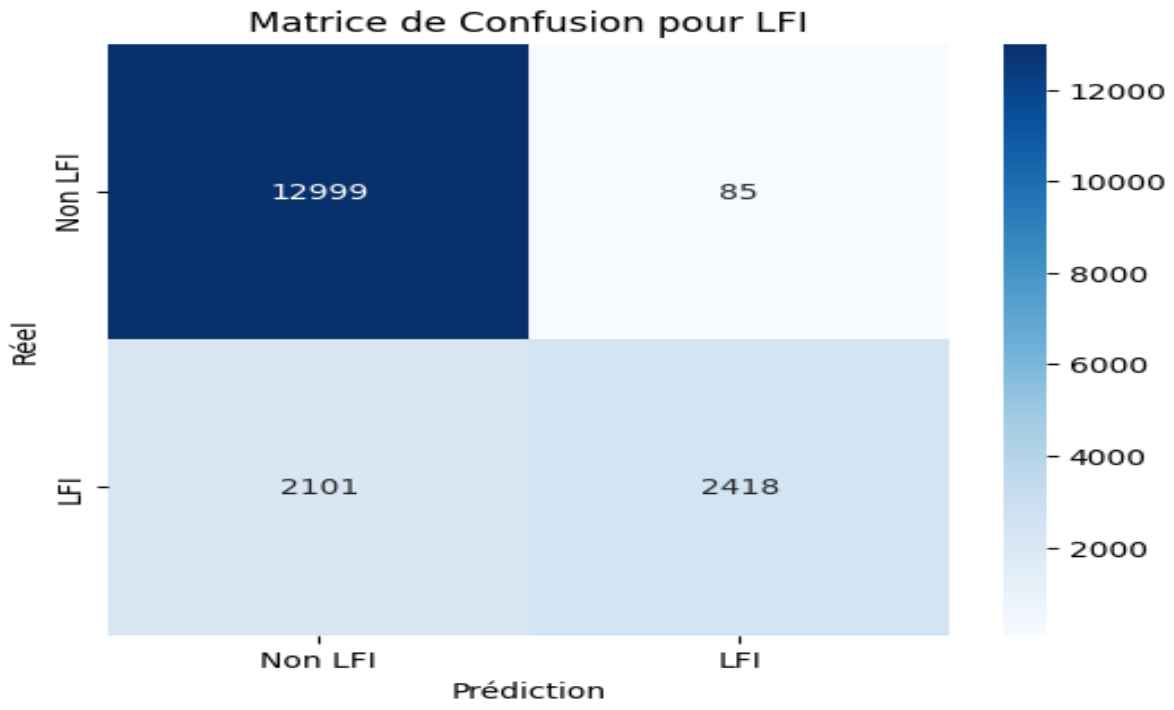


Figure V.10 : Matrice de confusion SVM BINAIRE LFI

Classifieur binaire XSS

Accuracy pour XSS : 0.9843

Rapport de classification :

Tableau V.9 : Rapport de classification SVM Binaire XSS

Rapport de classification pour XSS :

	precision	recall	f1-score	support
0	0.982	1.00	0.991	15227
1	0.999	0.884	0.938	2376
accuracy			0.984	17603
macro avg	0.991	0.942	0.965	17603
weighted avg	0.985	0.984	0.984	17603

Matrice de confusion :

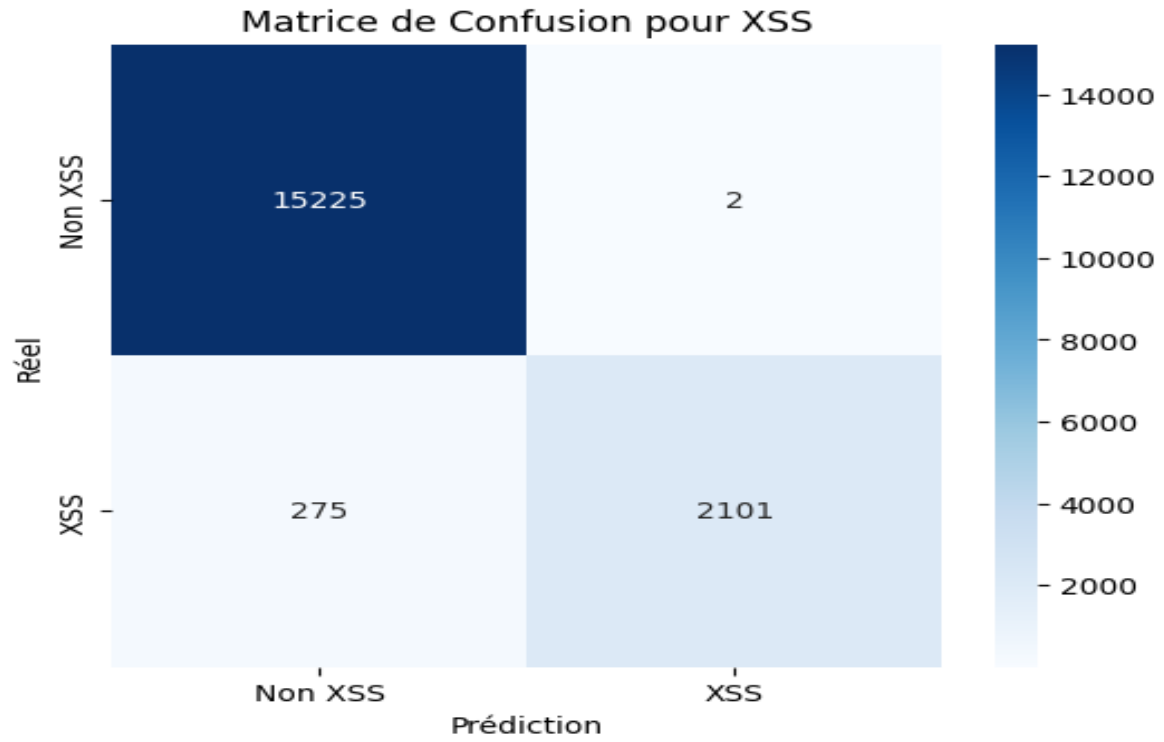


Figure V.11: Matrice de confusion SVM BINAIRE XSS

V.4.6.1.2. KNN

KNN classe un échantillon selon les classes des k voisins les plus proches.

- n_neighbors = 5 : nombre de voisins pris en compte.
- algorithm = 'auto' : sélection automatique de l'algorithme de recherche.
- weights = 'uniform' : tous les voisins ont le même poids.

a. Classifieur Multiclasse

Accuracy sur le dataset de test : 0.9916

Rapport de classification :

Tableau V.10 : Rapport de classification KNN Multiclasse

Rapport de classification :

Classe	Précision	Rappel	F1-score	Support
-1	0	0	0	34
1	0.993064	0.999894	0.996468	9451
2	0.994881	0.953393	0.973695	1223
3	0.985153	0.998451	0.991757	4519
4	0.997002	0.979798	0.988325	2376
Macro avg	0.79402	0.786307	0.790049	17603
Weighted avg	0.989773	0.991649	0.990653	17603

Matrice de confusion :

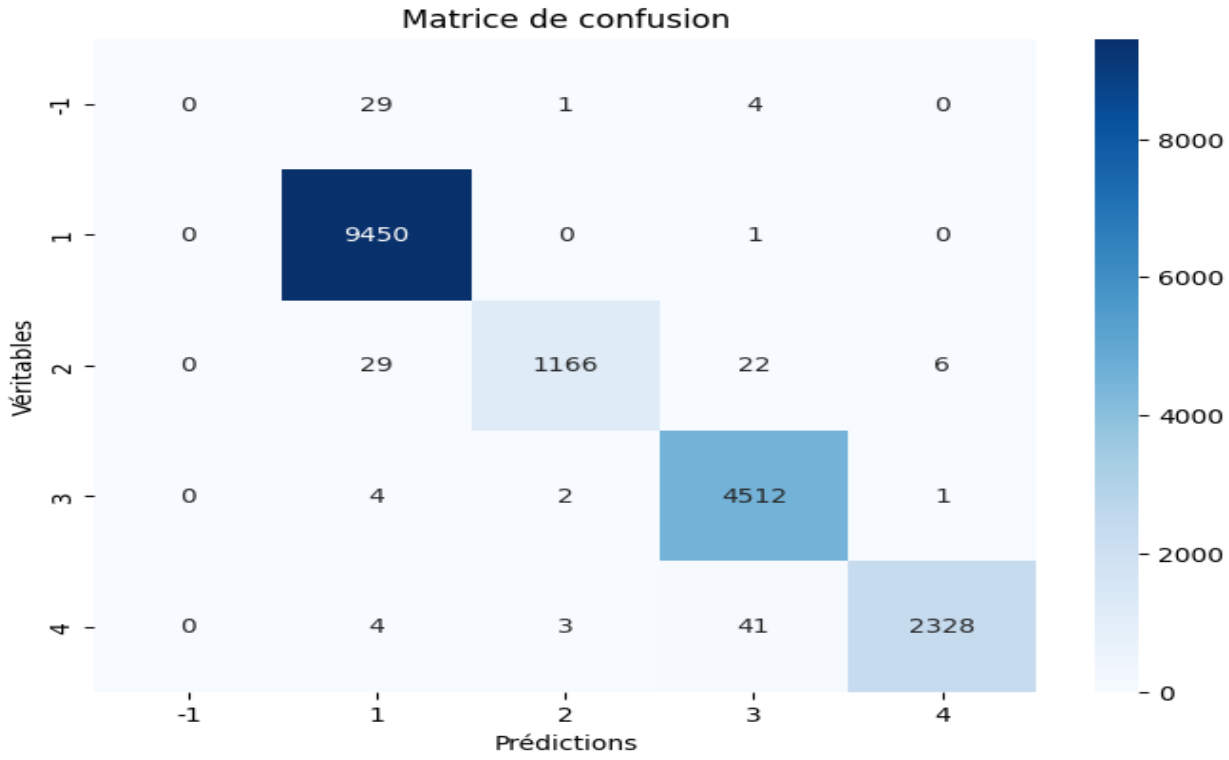


Figure V.12 : Matrice de confusion KNN MULTICLASSE

b. Classifieurs Binaires

Classifieur Binaire SQLi

Accuracy pour SQLi : 0.9918

Rapport de classification :

Tableau V.11 : Rapport de classification KNN Binaire SQLi

Classe	Precision	Recall	F1-score	Support
0	1.0	0.98	0.99	8152
1	0.99	1.0	0.99	9451
accuracy			0.99	17603
macro avg	0.99	0.99	0.99	17603
weighted avg	0.99	0.99	0.99	17603

Matrice de confusion :

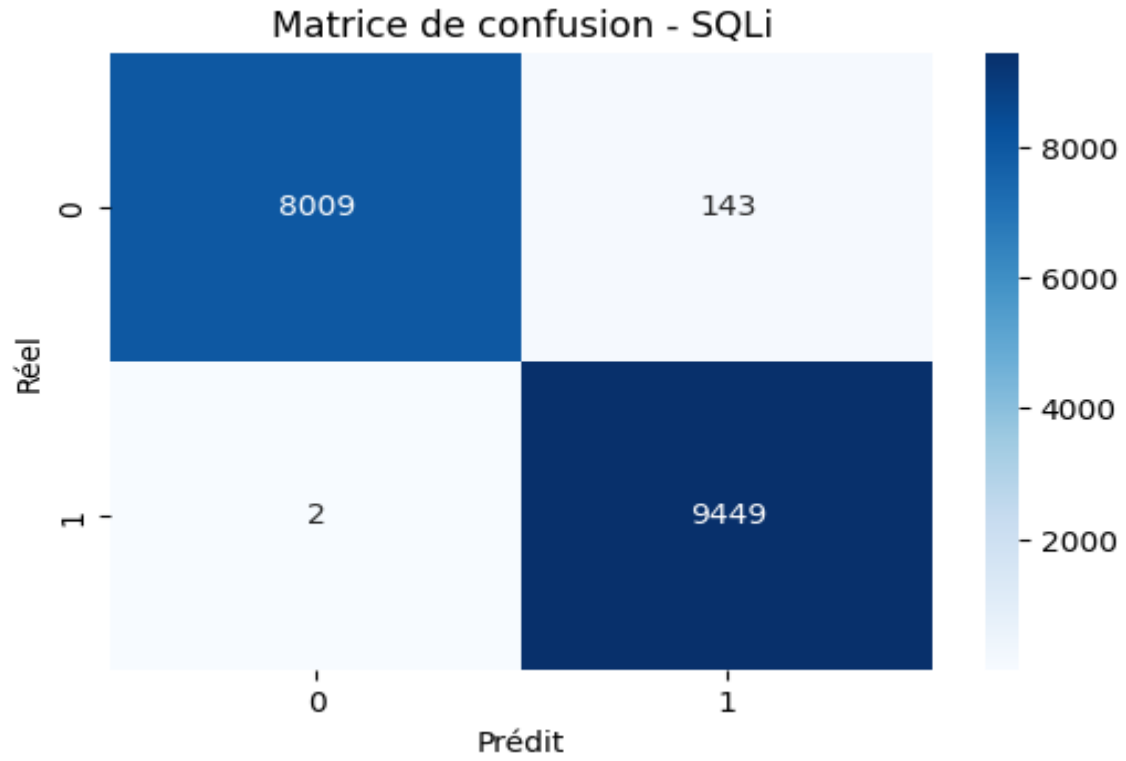


Figure V.13: Matrice de confusion KNN BINAIRE SQLi

Classifieur Binaire CMDi :

Accuracy pour CMDi : 0.9937

Rapport de classification :

Tableau V.12 : Rapport de classification KNN Binaire CMDi

Classe	Precision	Recall	F1-score	Support
0	0.99	1.0	1	16380
1	0.99	0.91	0.95	1223
accuracy			0.99	17603
macro avg	0.99	0.96	0.97	17603
weighted avg	0.99	0.99	0.99	17603

Matrice de confusion :

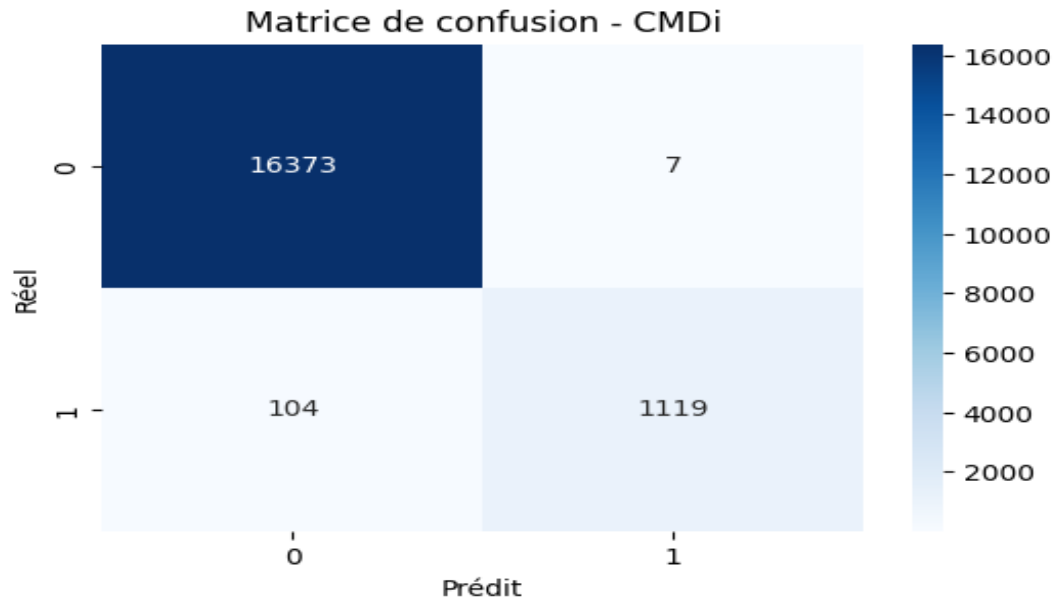


Figure V.14: Matrice de confusion KNN BINAIRE CMDI

Classifieur Binaire LFI :

Accuracy pour LFI : 0.9514

Rapport de classification :

Tableau V.13 : Rapport de classification KNN Binaire LFI

Classe	Precision	Recall	F1-score	Support
0	0.97	0.96	0.97	13084
1	0.9	0.91	0.91	4519
accuracy			0.95	17603
macro avg	0.93	0.94	0.94	17603
weighted avg	0.95	0.95	0.95	17603

Matrice de confusion :

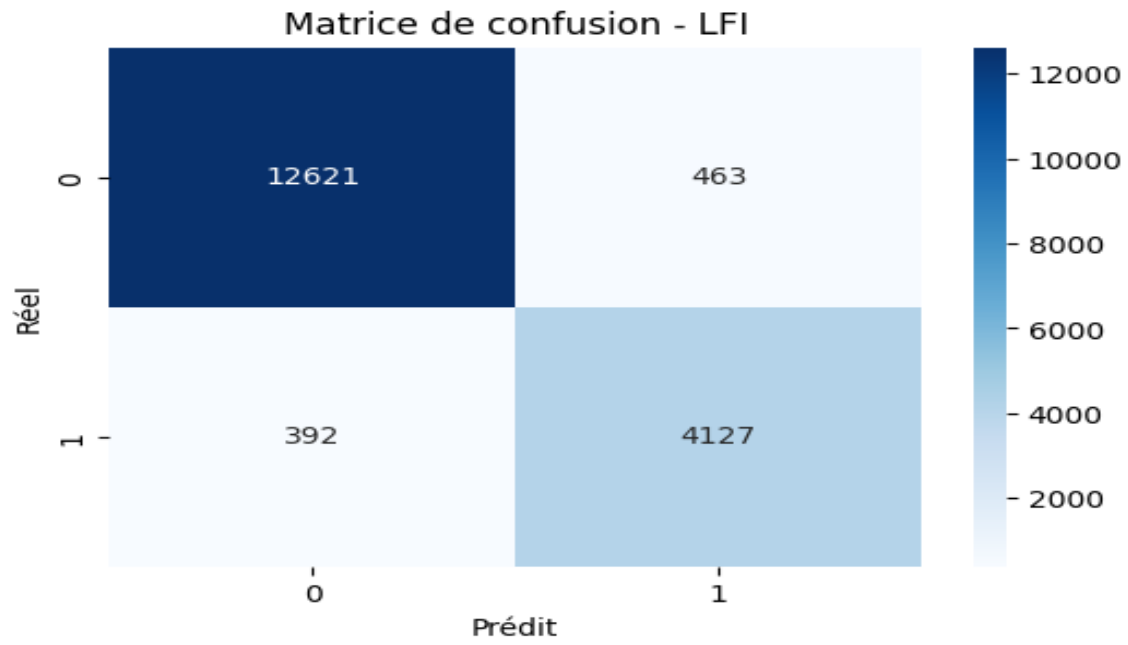


Figure V.15: Matrice de confusion KNN BINAIRE LFI

Classifieur Binaire XSS :

Accuracy pour XSS : 0.9910

Rapport de classification :

Tableau V.14 : Rapport de classification KNN Binaire XSS

Classe	Precision	Recall	F1-score	Support
0	0.99	1.0	0.99	15227
1	1.0	0.93	0.97	2376
accuracy			0.991	17603
macro avg	0.99	0.97	0.98	17603
weighted avg	0.99	0.99	0.99	17603

Matrice de confusion :

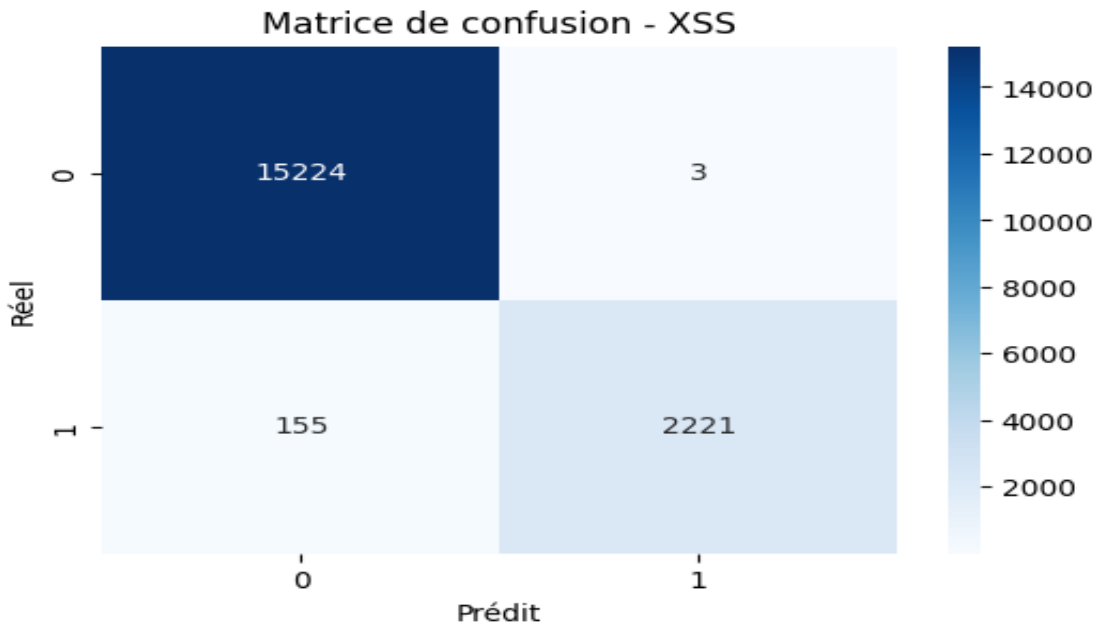


Figure V.16: Matrice de confusion KNN BINAIRE XSS

V.4.6.1.3. Regression Logistique

La régression logistique est un modèle linéaire adapté à la classification binaire ou multiclasse simple.

- max_iter = 2000 : nombre maximal d'itérations.
- C = 1.0 : paramètre de régularisation (L2).
- solver = 'liblinear' : méthode d'optimisation utilisée.
- multi_class = 'ovr' : stratégie One-vs-Rest pour la classification multiclasse.
- penalty = 'l2' : pénalisation L2 pour éviter le surapprentissage.
- class_weight = 'balanced' : équilibre les poids des classes.
- random_state = 42 : assure des résultats constants.

a. Classifieur Multiclasse

Accuracy sur le dataset de test avec la régression logistique : 0.9780

Rapport de classification :

Tableau V.15 : Rapport de classification RL Multiclasse

Classe	Precision	Recall	F1-score	Support
-1	0.0	0.0	0	34
1	0.99	1.0	0.99	9451
2	0.99	0.85	0.92	1223
3	0.95	1.0	0.97	4519
4	1.0	0.93	0.96	2376
accuracy			0.98	17603
macro avg	0.78	0.76	0.77	17603
weighted avg	0.98	0.98	0.98	17603

Matrice de confusion :

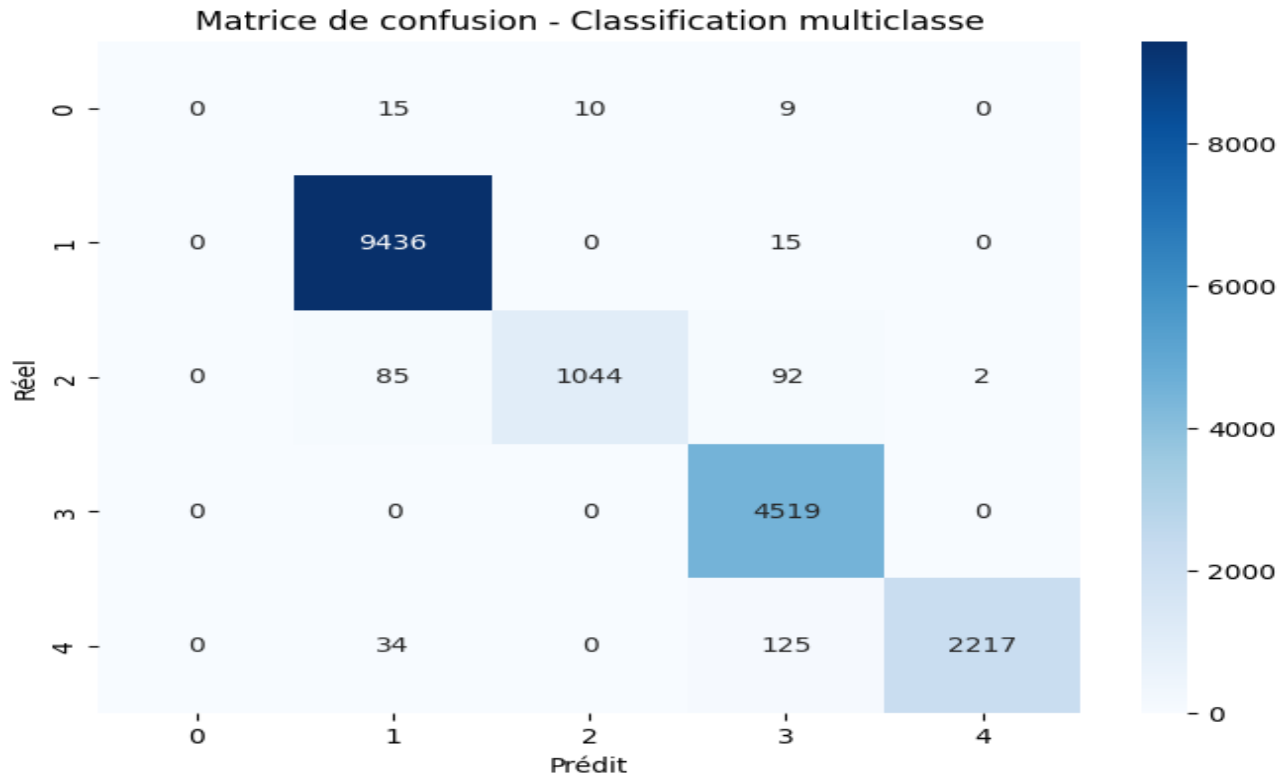


Figure V.17: Matrice de confusion RL MULTICLASSE

b. Classifieurs Binaires

Classifieur Binaire SQLi :

Accuracy pour SQLi : 0.9873

Rapport de classification :

Tableau V.16 : Rapport de classification RL Binaire SQLi

Classe	Precision	Recall	F1-score	Support
0	1.0	0.97	0.99	8152
1	0.98	1.0	0.99	9451
accuracy			0.99	17603
macro avg	0.99	0.99	0.99	17603
weighted avg	0.99	0.99	0.99	17603

Matrice de confusion :

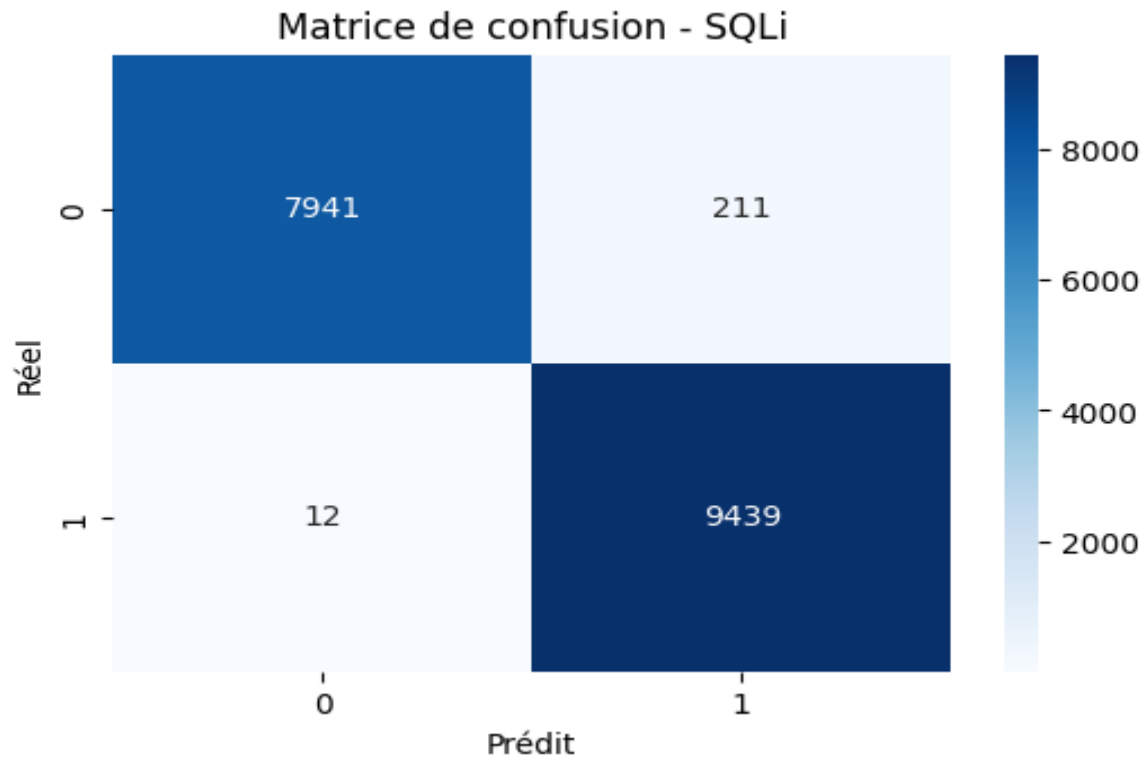


Figure V.18: Matrice de confusion RL BINAIRE SQLi

Classifieur Binaire CMDi :

Accuracy pour CMDi : 0.9532

Rapport de classification :

Tableau V.17 : Rapport de classification RL Binaire CMDi

Classe	Precision	Recall	F1-score	Support
0	0.95	1.0	0.98	16380
1	0.99	0.33	0.5	1223
accuracy			0.95	17603
macro avg	0.97	0.67	0.74	17603
weighted avg	0.95	0.95	0.94	17603

Matrice de confusion :

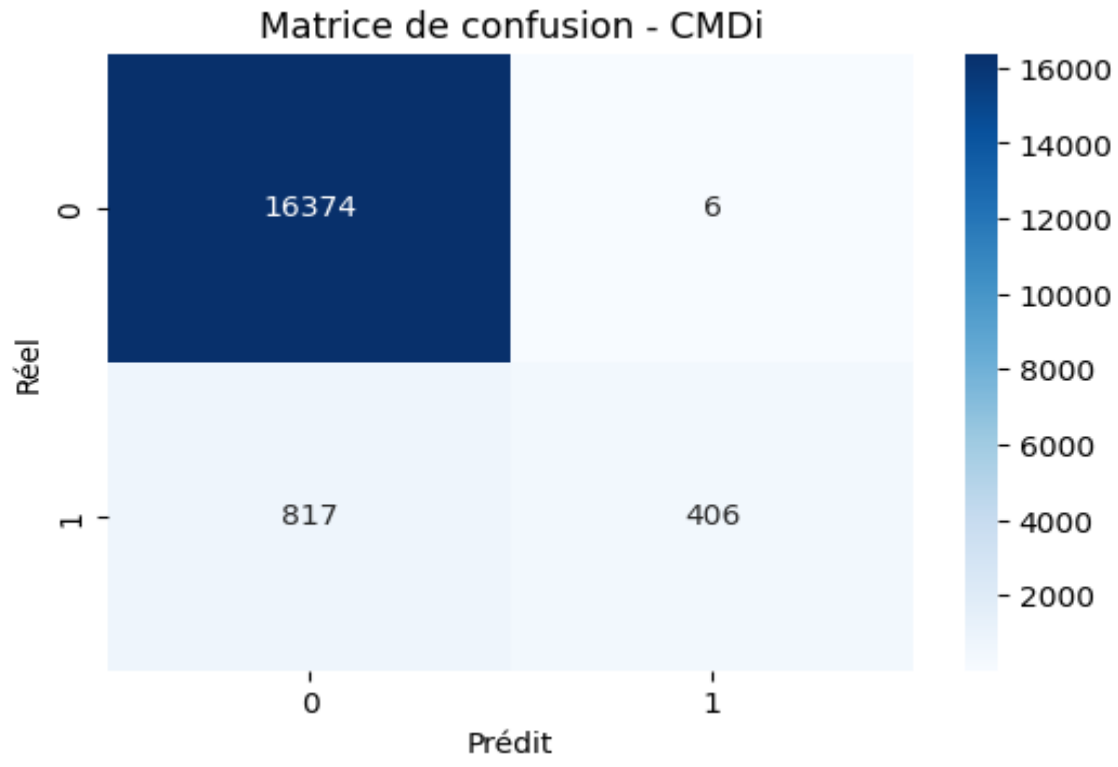


Figure V.18: Matrice de confusion RL BINAIRE CMDI

Classifieur Binaire LFI :

Accuracy pour LFI : 0.9702

Rapport de classification :

Tableau V.18 : Rapport de classification RL Binaire LFI

Classe	Precision	Recall	F1-score	Support
0	0.97	0.99	0.98	13084
1	0.98	0.9	0.94	4519
accuracy			0.97	17603
macro avg	0.97	0.95	0.96	17603
weighted avg	0.97	0.97	0.97	17603

Matrice de confusion :

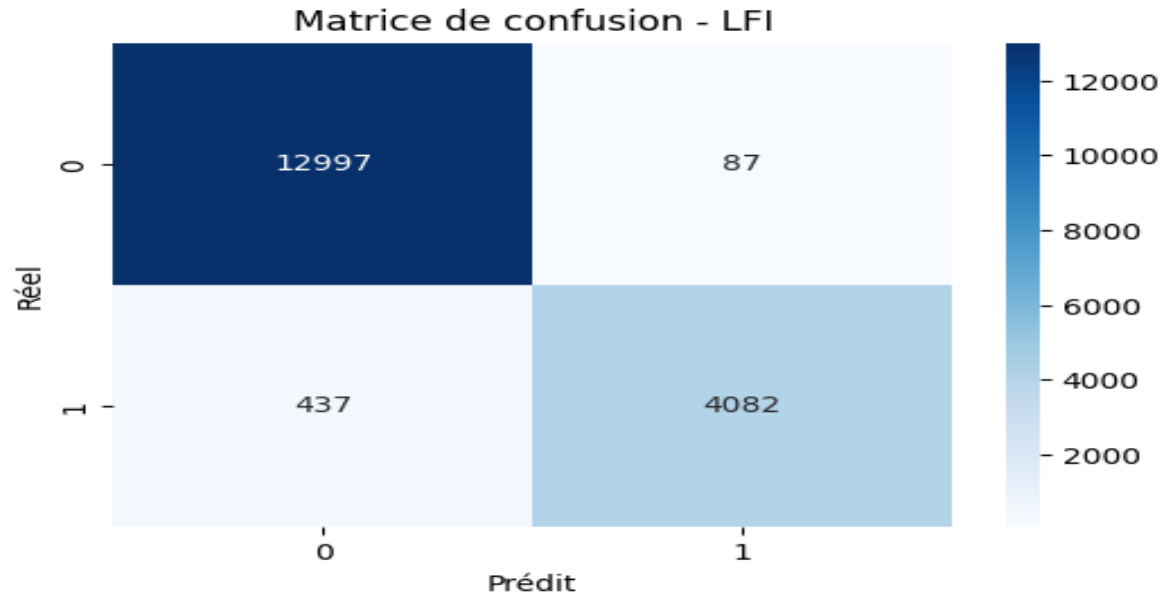


Figure V.19 : Matrice de confusion RL BINAIRE LFI

Classifieur Binaire XSS :

Accuracy pour XSS : 0.8728

Rapport de classification :

Tableau V.19 : Rapport de classification RL Binaire XSS

Classe	Precision	Recall	F1-score	Support
0	0.87	1.0	0.93	15227
1	0.99	0.06	0.11	2376
accuracy			0.87	17603
macro avg	0.93	0.53	0.52	17603
weighted avg	0.89	0.87	0.82	17603

Matrice de confusion :

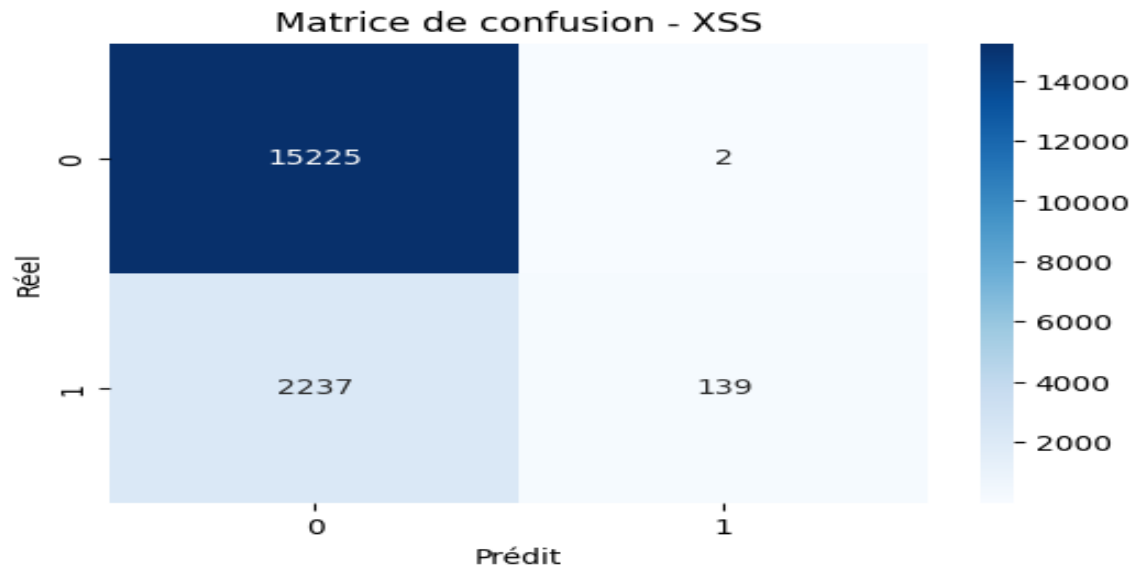


Figure V.20 : Matrice de confusion RL BINAIRE XSS

V.4.6.1.4. Random Forest

Random Forest est un ensemble d'arbres de décision utilisé pour améliorer la précision et réduire le surapprentissage.

- n_estimators = 200 : nombre d'arbres dans la forêt.
- max_depth = 10 : profondeur maximale de chaque arbre.
- random_state = 42 : rend les résultats reproductibles.

a. Classifieur Multiclasse :

Accuracy sur le dataset de test : 0.9260

Rapport de classification :

Tableau V.20 : Rapport de classification RF Multiclasse

Rapport de classification :

Classe	Précision	Rappel	F1-score	Support
-1	0	0	0	34
1	0.884583	0.999894	0.938711	9451
2	0.999045	0.855274	0.921586	1223
3	0.982143	0.803275	0.883749	4519
4	0.999081	0.915404	0.955414	2376
Macro avg	0.77297	0.714769	0.739892	17603
Weighted avg	0.931327	0.926035	0.923853	17603

Matrice de confusion :

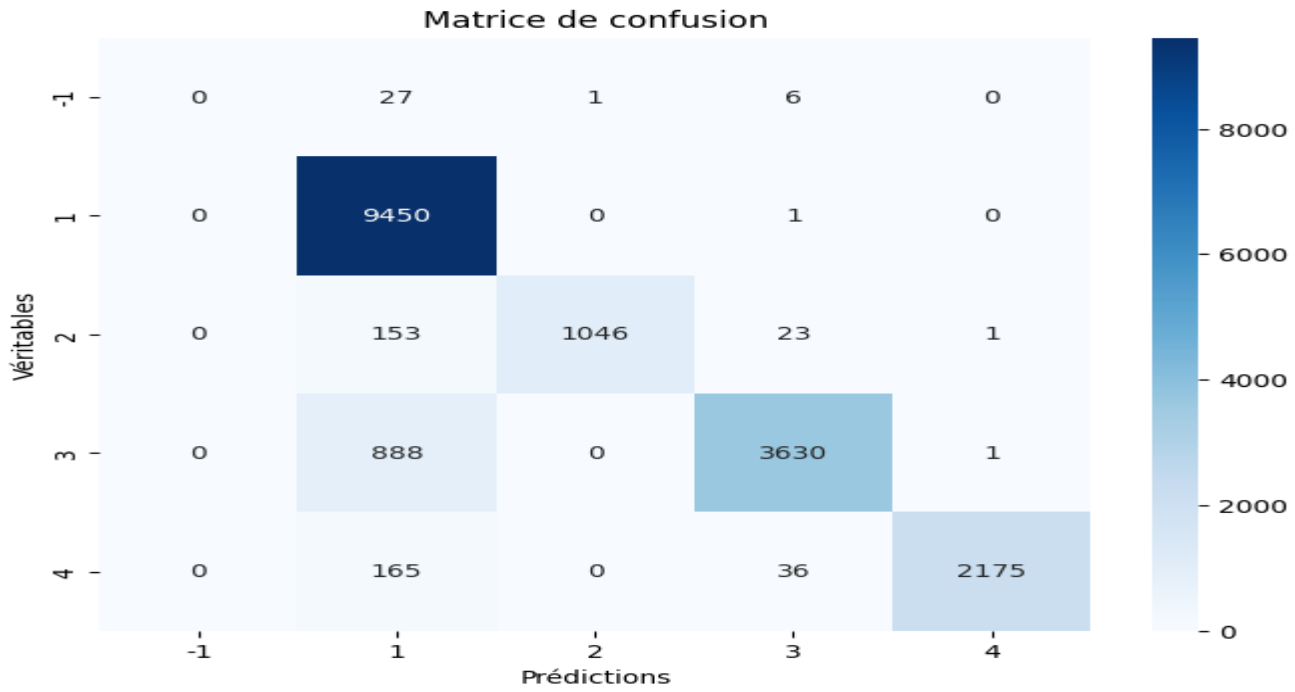


Figure V.21 : Matrice de confusion RF MULTICLASSE

b. Classifieurs Binaires

Classifieur Binaire SQLi :

Accuracy pour SQLi avec Random Forest : 0.5371

Rapport de classification : ;

Tableau V.21 : Rapport de classification RF Binaire SQLi

Classe	Precision	Recall	F1-score	Support
0	1.0	0.0	0	8152
1	0.54	1.0	0.7	9451
accuracy			0.54	17603
macro avg	0.77	0.5	0.35	17603
weighted avg	0.75	0.54	0.38	17603

Matrice de confusion :

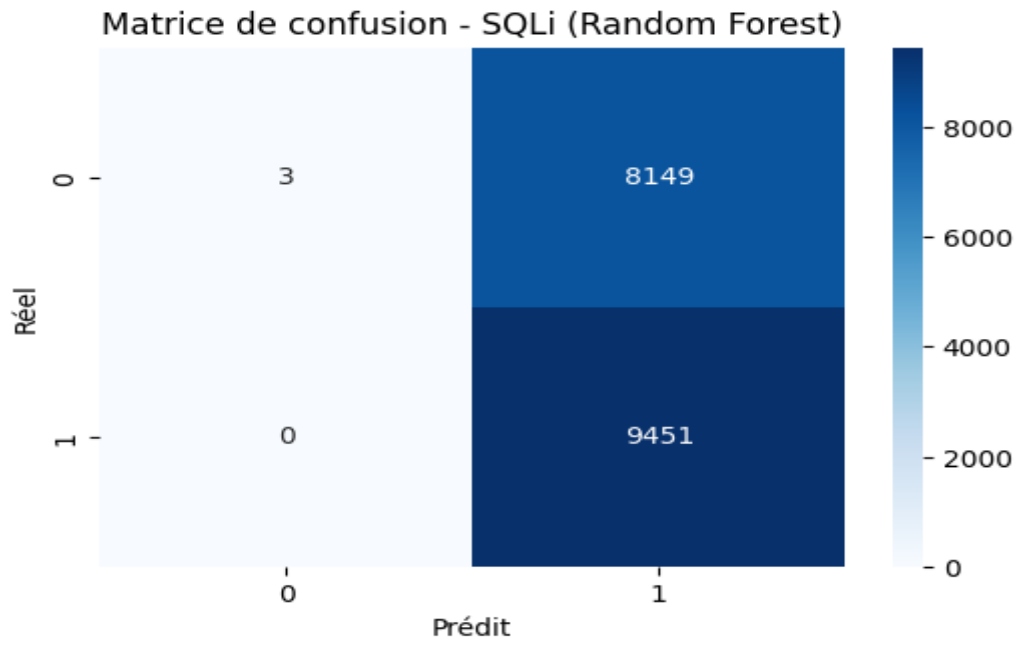


Figure V.22 : Matrice de confusion RF BINAIRE SQLi

CMDi :

Accuracy pour CMDi avec Random Forest : 0.9896

Rapport de classification :

Tableau V.22 : Rapport de classification RF Binaire CMDi

Classe	Precision	Recall	F1-score	Support
0	0.99	1.0	0.99	16380
1	1.0	0.85	0.92	1223
accuracy			0.99	17603
macro avg	0.99	0.93	0.96	17603
weighted avg	0.99	0.99	0.99	17603

Matrice de confusion :

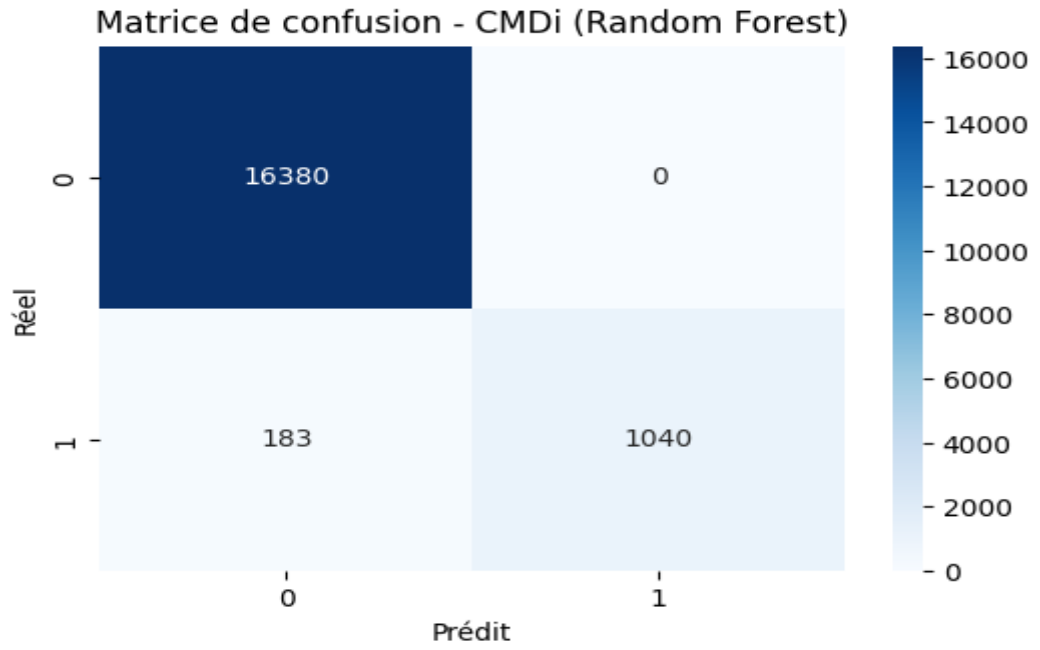


Figure V.23: Matrice de confusion RF BINAIRE CMDI

Classifieur Binaire LFI :

Accuracy pour LFI avec Random Forest : 0.7852

Rapport de classification :

Tableau V.23 : Rapport de classification RF Binaire LFI

Classe	Precision	Recall	F1-score	Support
0	0.78	1.0	0.87	13084
1	1.0	0.16	0.28	4519
accuracy			0.79	17603
macro avg	0.89	0.58	0.58	17603
weighted avg	0.83	0.79	0.72	17603

Matrice de confusion :

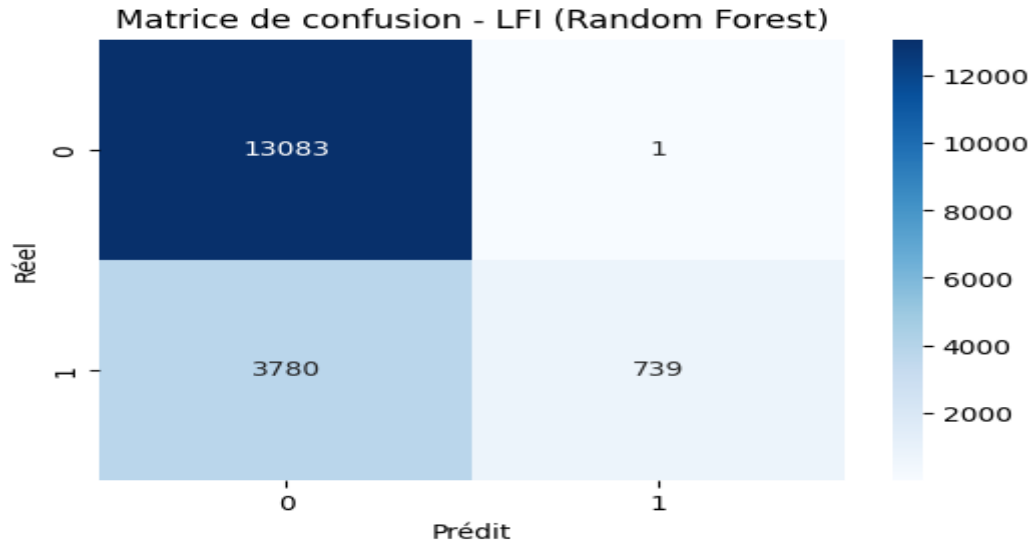


Figure V.24 : Matrice de confusion RF BINAIRE LFI

Classifieur Binaire XSS :

Accuracy pour XSS avec Random Forest : 0.9001

Rapport de classification :

Tableau V.24 : Rapport de classification RF Binaire XSS

Classe	Precision	Recall	F1-score	Support
0	0.9	1.0	0.95	15227
1	1.0	0.26	0.41	2376
accuracy			0.9	17603
macro avg	0.95	0.63	0.68	17603
weighted avg	0.91	0.9	0.87	17603

Matrice de confusion :

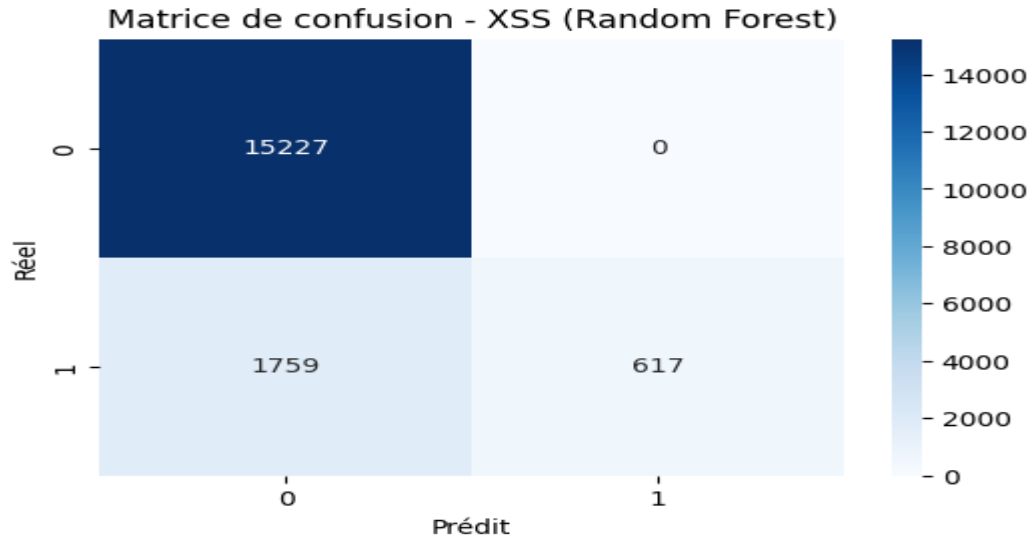


Figure V.25 : Matrice de confusion RF BINAIRE XSS

V.4.6.2. Etude comparative

1) Les Modèles Multiclasse

Tableau V.25 : La précision les modèles multiclasses expérimentés

Modèle	Accuracy (Multiclasse)
SVM	0.9744
KNN	0.9916
Régression Logistique	0.9780
Random Forest	0.9260

Discussion

Les résultats obtenus mettent en évidence des performances globalement élevées des modèles testés, tant pour la classification multiclasse que pour les classifieurs binaires spécialisés par type d'attaque. Pour la classification multiclasse, le modèle K-Nearest Neighbors (KNN) se distingue avec une précision de 99,16 %, suivi de près par la régression logistique (97,80 %) et SVM (97,44 %). En revanche, Random Forest affiche des performances moindres dans ce contexte, avec une précision de 92,60 %.

Les modèles des classifieurs binaires pour chaque attaque

Tableau V.26 : la précision les modèles de classifieurs binaires expérimentés

Attaque	SVM	KNN	Régression Logistique	Random Forest
SQLi	0.9841	0.9918	0.9873	0.5371
CMDi	0.9896	0.9896	0.9532	0.9896
LFI	0.8758	0.9514	0.9702	0.7852
XSS	0.9843	0.9910	0.9910	0.9001

Discussion

En ce qui concerne les modèles binaires, une approche par attaque a permis d'affiner les résultats. Pour les attaques SQLi et CMDi, SVM et KNN affichent des précisions proches ou supérieures à 98 %, tandis que Random Forest présente une efficacité très variable (notamment très faible sur SQLi avec 53,71 %). La régression logistique s'avère particulièrement performante pour l'attaque LFI (97,02 %) et XSS (99,10 %). Cette diversité des performances confirme la pertinence d'une spécialisation des modèles binaires selon la nature de l'attaque.

Ainsi, le choix des classifieurs retenus dans notre solution repose à la fois sur leurs performances empiriques et leur stabilité selon le type d'attaque ciblé :

- Pour le classifieur multiclasse, KNN a été privilégié pour sa précision supérieure.
- Concernant les modèles binaires, les meilleurs résultats observés pour chaque type d'attaque ont guidé une sélection différenciée, garantissant une détection fine et adaptée à chaque vecteur de menace. Les choix des modèles binaires sont présentés dans le

Tableau V.27 : Meilleur Classifieur binaire retenu pour chaque attaque

Attaque	Meilleur Modèle Binaire
SQLi	KNN
CMDi	SVM / Random Forest
LFI	Régression Logistique
XSS	KNN

V.4.7 Module de décision

Dans ce processus de détection d'attaques, nous avons appliqué des règles basées sur des seuils de probabilité pour classifier les instances en fonction de leur statut d'attaque. Trois étapes principales ont été suivies : la classification multiclasse, la vérification des attaques définies, et la gestion des instances indéfinies. Chaque étape repose sur des seuils de probabilité bien définis, permettant de prendre des décisions claires pour chaque instance analysée.

1. Classification multiclasse

$$P_{\max}(i) > T1 \Rightarrow \text{Etat}(i) = \text{"DEF"} \quad \text{sinon} \quad \text{Etat}(i) = \text{"UND"}$$

2. Vérification des attaques définies :

$$\max(P_{\text{sqli}}(i), P_{\text{cmdi}}(i), P_{\text{lfi}}(i), P_{\text{xss}}(i)) > T2 \Rightarrow \text{Etat}(i) = \text{"ATT"} \quad \text{sinon} \quad \text{Etat}(i) = \text{"UNC"}$$

3. Gestion des instances indéfinies

$$\max(P_{\text{sqli}}(i), P_{\text{cmdi}}(i), P_{\text{lfi}}(i), P_{\text{xss}}(i)) > T3 \Rightarrow \text{Etat}(i) = \text{"AON"} \quad \text{sinon} \quad \text{Etat}(i) = \text{"NAT"}$$

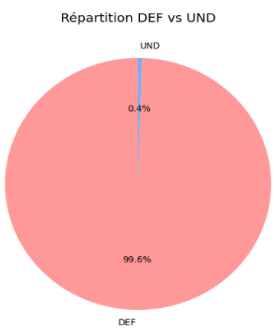
Seuils :

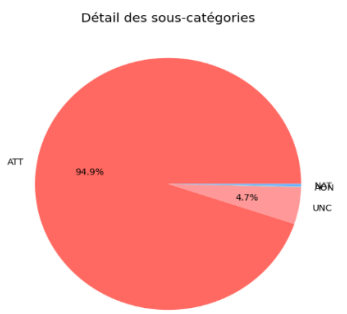
- T1 = 0.7 : Pour déterminer si l'attaque est définie ("DEF") ou indéfinie ("UND").
- T2 = 0.8 : Seuil pour classer une instance comme attaque détectée ("ATT") ou incertaine ("UNC").
- T3 = 0.3 : Seuil modéré pour classer les instances "UND" en absence d'attaque ("AON") ou trafic naturel ("NAT").

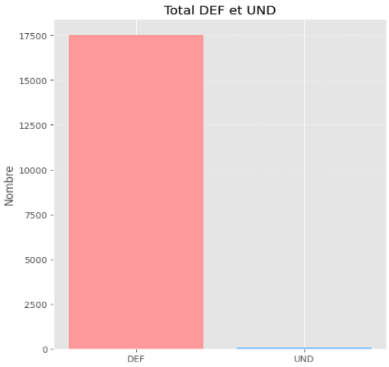
Ces règles permettent de classifier efficacement les instances, en tenant compte des probabilités associées aux différents types d'attaques et en gérant les cas d'incertitude de manière cohérente.

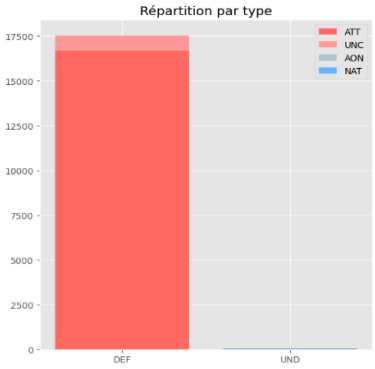
Tableau V.26 : Résultat de l'application de l'heuristique

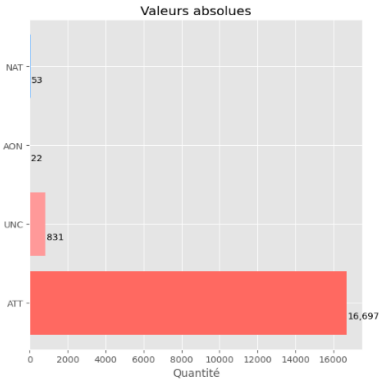
Application de l'heuristique			
DEF		UND	
17528		75	
ATT	UNC	AON	NAT
16697	831	22	53



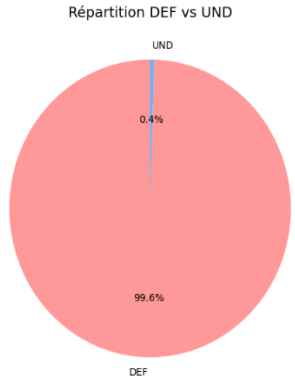


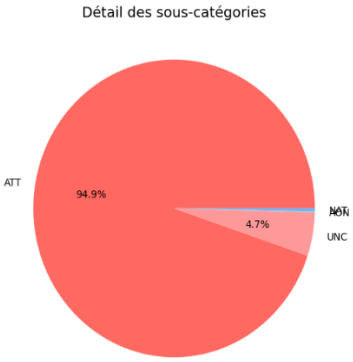


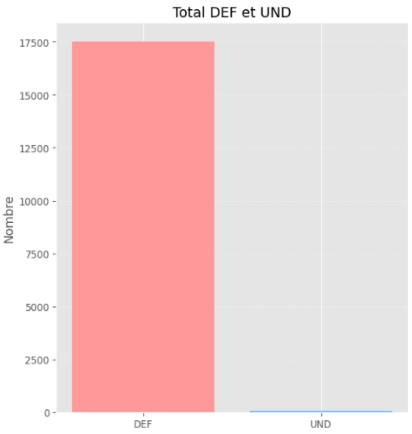


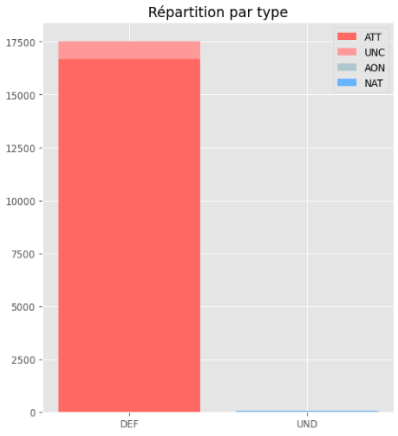


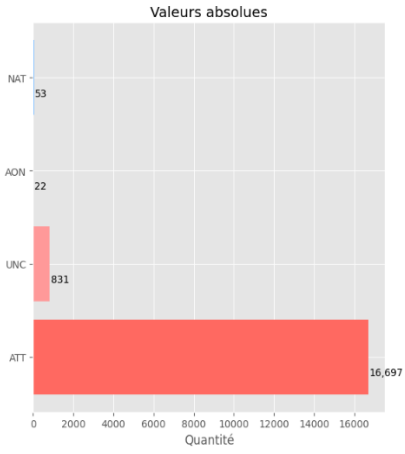
Explication :
DEF = Défini
UND = Non défini
ATT = Attaques/Attributs
UNC = Incertain
AON = Anomalies
NAT = Naturel











Explication :
DEF = Défini
UND = Non défini
ATT = Attaques/Attributs
UNC = Incertain
AON = Anomalies
NAT = Naturel

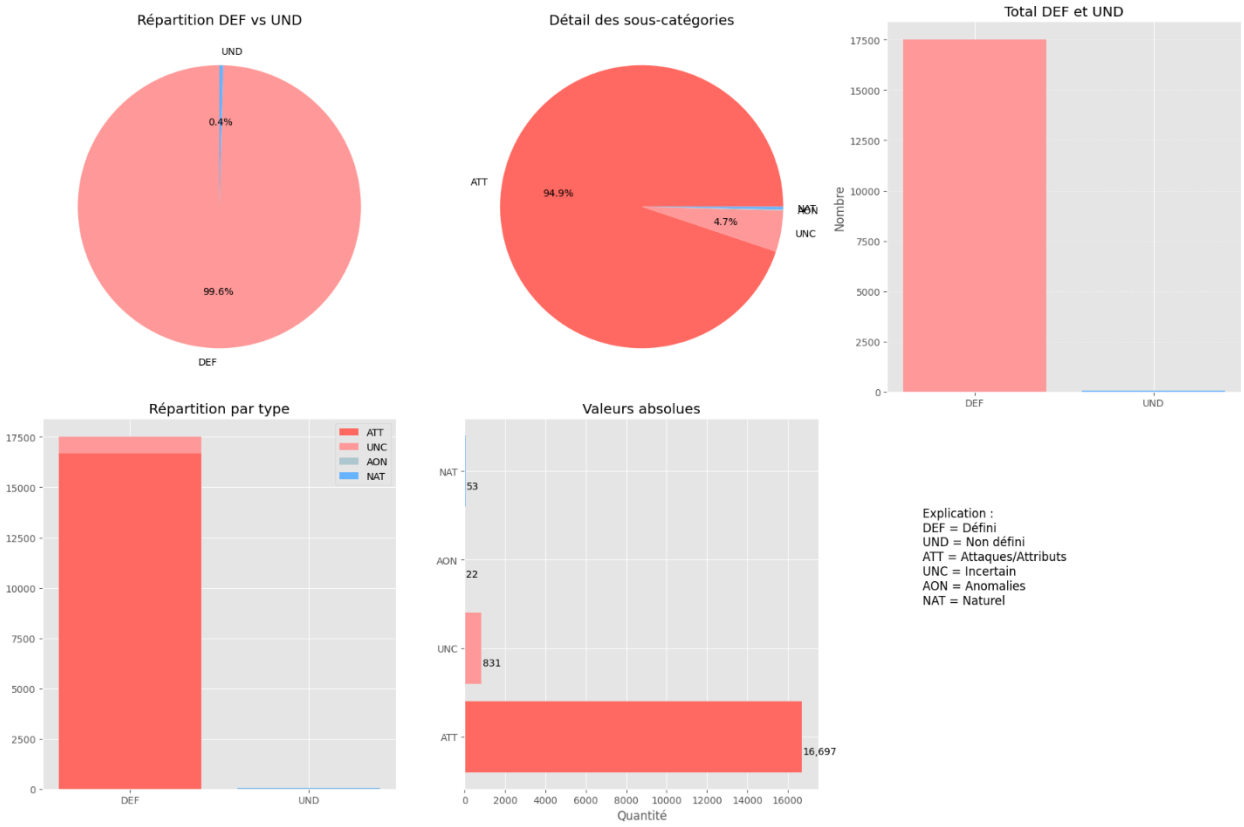


Figure V.26 : Résultats de l'application de l'heuristique

V 5. Conclusion

Ce chapitre a détaillé la mise en œuvre technique de l'ensemble des modules constituant notre système d'analyse des journaux web. Il a d'abord présenté les étapes de prétraitement des données, indispensables pour structurer et normaliser les logs en vue d'une exploitation automatique. Ensuite, les modules d'intelligence artificielle ont été développés et testés : un détecteur d'anomalies basé sur des algorithmes non supervisés, un classifieur supervisé multiclasse suivie de quatre classifieurs en parallèle pour l'identification des types d'attaques, et enfin une heuristique de décision permettant d'intégrer les différentes sorties dans une logique de détection cohérente.

Les expérimentations ont permis d'évaluer les performances individuelles des modèles selon différents jeux de données, ainsi que leur capacité à détecter des attaques, y compris inédites. Les résultats obtenus démontrent la pertinence de l'approche hybride adoptée, combinant des modèles d'apprentissage machine et des règles heuristiques afin de renforcer la précision, la robustesse et la généralisation du système.

Dans un contexte où les menaces informatiques deviennent de plus en plus sophistiquées, fréquentes et furtives, la capacité des organisations à détecter proactivement les comportements malveillants constitue un enjeu stratégique majeur. Ce mémoire s'est attaché à répondre à la problématique suivante : *comment concevoir un système automatisé, robuste et évolutif capable de détecter efficacement aussi bien les attaques connues que les menaces émergentes à partir de l'analyse des fichiers de logs web ?*

1. Principaux Aspects Abordés

Pour répondre à cette problématique, nous avons focalisés nos efforts surs :

1.1) l'étude des problèmes de sécurité des applications web et les stratégies de protection existantes avec leurs avantages et limitations ;

1.2) L'orientation vers ce paradigme mystérieux et prometteur des fichiers journaux web, les méthodes d'analyses existantes manuelles et automatiques et le besoin d'une analyse avancée intelligente ;

1.3) L'orientation vers les méthodes d'intelligence artificielle particulièrement l'apprentissage automatique et les techniques de machines supervisées et non supervisées.

2. Contribution apportée

Un système d'analyse avancée de logs web basée sur :

1) Une approche hybride a été élaborée, articulant des techniques d'apprentissage supervisé pour la classification des attaques avérées, et des méthodes non supervisées pour la détection d'anomalies comportementales. Cette architecture modulaire permet de bénéficier à la fois de la précision des modèles supervisés et de la capacité d'adaptation des algorithmes non supervisés face à des menaces inédites (notamment les attaques zero-day). Elle repose sur un

pipeline structuré intégrant des modules de prétraitement avancé, d'ingénierie des caractéristiques, d'entraînement, d'évaluation et de prise de décision.

2) Un module de décision basée sur une heuristique qui offre à la classification des attaques un aspect scientifique et ajoute un deuxième niveau d'intelligence au système en plus de la généralisation qui est le fruit de l'apprentissage.

3. Avantages de la contribution

Les principaux avantages de la solution proposée résident dans sa capacité à traiter un volume important de données hétérogènes, sa résilience face aux attaques inconnues, ainsi que dans la pertinence opérationnelle des alertes générées, qui permettent de réduire significativement les faux positifs. L'adoption de techniques d'encodage, de normalisation et de vectorisation avancées renforce la qualité des représentations utilisées pour l'apprentissage machine, assurant ainsi des performances fiables en conditions réelles.

4. Limites de la contribution

Toutefois, certaines limites subsistent, notamment liées à la dépendance à la qualité des jeux de données, à la difficulté de généralisation dans des environnements très hétérogènes, ou encore à la complexité du paramétrage des modèles non supervisés. Ces éléments nécessitent des développements futurs.

5. Perspectives

Parmi les perspectives d'amélioration, plusieurs axes peuvent être envisagés :

- Intégration de données contextuelles (métadonnées réseau, logs système, journaux applicatifs) pour enrichir la corrélation inter-source ;
- Automatisation du recalibrage des seuils décisionnels par des techniques de meta-learning ou de reinforcement learning ;
- Déploiement en production via une API sécurisée ou un module embarqué dans une architecture SIEM ;

Ce travail démontre que l'union raisonnée entre l'intelligence artificielle et l'exploitation intelligente des logs web constitue une voie prometteuse pour renforcer les dispositifs de cybersécurité modernes, en leur conférant à la fois robustesse et capacité d'anticipation.

Bibliographie

{1} Kanawati, R. (2023, janvier). Introduction à la cybersécurité. https://www-lipn.univ-paris13.fr/~kanawati/pdf/cybersec-CYU.pdf

{2} Bouadjemi, A. (2022–2023). Sécurité informatique. http://dspace.univ-relizane.dz/home/handle/123456789/371

{3} Sullivan, B., & Liu, V. (n.d.). Web Application Security – A Beginner's Guide. https://fr.scribd.com/document/465406616/Web-Application-Security-A-Beginner-s-Guide-pdf

{4} Ekpobimi, H., & Fasanmade, A. (2024, septembre). Front-end development and cybersecurity: A conceptual approach to building secure web applications. https://www.researchgate.net/

{5} OWASP. (n.d.). OWASP Top Ten. https://owasp.org/www-project-top-ten/

{6} Charif, A. (2022, décembre 20). What is a log file? CrowdStrike. https://www.crowdstrike.com/en-us/cybersecurity-101/next-gen-siem/log-file

{7} The Graylog Team. (2020, décembre 9). What is a log format? Graylog. https://www.graylog.org

{8} Schmidt, K. J., Chuvakin, A. A., & Phillips, C. (2012, novembre). Logging and log management: The authoritative guide to understanding the concepts surrounding logging and log management.

{9} Kortak, M. (2024, juillet 9). A log analysis tutorial. https://builtin.com/articles/log-analysis

{10} Bruckman, A. (2006, janvier). Chapter 58: Analysis of log file data to understand behavior and learning in an online community. Georgia Institute of Technology. https://sites.cc.gatech.edu/fac/Amy.Bruckman/papers/chapters/log-file-analysis-06.pdf

{11} Locke, S., Li, H., Chen, T.-H. (Peter), Shang, W., & Liu, W. (2021, mai). LogAssist: Assisting log analysis through log summarization. IEEE. https://www.researchgate.net/publication/351894040_LogAssist_Assisting_Log_Analysis_Through_Log_Summarization

{12} Splunk. (2023, décembre). The importance of log management. https://www.splunk.com/en_us/blog/learn/log-analysis.html

{13} Vargas, G., Martins, E., & Almeida, J. (2021, octobre). Apprentissage automatique in log analysis: A review. Journal of Network and Computer Applications, 88, 57–69. https://doi.org/10.1109/MSEC.2021.3113275

{14} Janiesch, C., Zschech, P., & Heinrich, K. (2021, avril). Apprentissage automatique et apprentissage profond. https://doi.org/10.1007/s12525-021-00475-2

{15} Vallée, L. N., & Aka, J.-B. (2024). Intelligence artificielle et entrepreneuriat. Communication, technologie et développement, 16. https://journals.openedition.org/ctd/12942

{16} Siwach, M., & Mann, S. (2022). Anomaly detection for web log data analysis: A review. Journal of Algebraic Statistics, 13(1), 129–148. https://publishoa.com

{17} Branco, R., Cogo, V., & Medeiros, I. (2024). Towards a web application attack detection system based on network traffic and log classification. LASIGE, Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa. https://doi.org/10.54499/UIDB/00408/2020 ; https://doi.org/10.54499/UIDP/00408/2020

{18} Saleem, S., Sheeraz, M., Hanif, M., & Farooq, U. (2020). Web server attack detection using apprentissage automatique. In 2020 International Conference on Cyber Warfare and Security. IEEE. https://www.researchgate.net/publication/347979779_Web_Server_Attack_Detection_using_Machine_Learning

{19} Gao, Y., Ma, Y., & Li, D. (2017). Anomaly detection of malicious users' behaviors for web applications based on web logs. In 2017 17th IEEE International Conference on Communication Technology. https://doi.org/10.1109/ICCT.2017.8359854

{20} Cao, Q., Qiao, Y., & Lyu, Z. (2017). Apprentissage automatique to detect anomalies in web log analysis. In Proceedings of the 3rd IEEE International Conference on Computer and Communications. https://doi.org/10.1109/CompComm.2017.8322600

{21} Zhang, W., & Chen, L. (2019). Web log anomaly detection based on Isolation Forest algorithm. In Proceedings of the 3rd IEEE International Conference on Computer and Communications. https://doi.org/10.1109/ISKE47853.2019.9170430

{22} Chua, W., Pajas, A. L. D., Castro, C. S., Panganiban, S. P., Pasuquin, A. J., Purganan, M. J., Malupeng, R., Pingad, D. J., Orolfo, J. P., Lua, H. H., & Velasco, L. C. (2024). Web traffic anomaly detection using Isolation Forest. Informatics, 11(1), 83. https://doi.org/10.3390/informatics11040083

Annexe 1 : Algorithmes de Apprentissage automatique

1.1. Approches supervisées

Les approches supervisées utilisent des données étiquetées (avec des classes connues) pour entraîner des modèles capables de prédire ou de classer de nouvelles observations.

- **SVM (Support Vector Machine)**
  Algorithme qui cherche à trouver l'hyperplan optimal séparant les classes avec la plus grande marge. Très efficace pour les données à faible dimension ou les problèmes linéairement séparables.

- **Random Forest**
  Ensemble d'arbres de décision formés sur des sous-ensembles aléatoires du jeu de données. Offre une bonne précision, évite l'overfitting et gère bien les données bruitées.

- **KNN (K-Nearest Neighbors)**
  Classifie un point selon la majorité des k voisins les plus proches dans l'espace des caractéristiques. Simple et efficace, mais coûteux en calcul lors de la prédiction.

- **Régression Logistique**
  Modèle linéaire utilisé pour des problèmes de classification binaire ou multi-classe. Donne une probabilité d'appartenance à chaque classe.

1.2. Approches non supervisées

Ces méthodes s'appliquent sur des données non étiquetées pour découvrir des structures cachées, des anomalies ou des regroupements.

- **K-Means**
  Algorithme de clustering qui regroupe les données en K clusters en minimisant la distance intra-cluster. Efficace mais sensible au choix de K et aux valeurs initiales.

- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**
  Regroupe les points denses et identifie les points isolés comme du bruit. Très utile pour détecter des structures non sphériques et des anomalies.

- **Isolation Forest**

  Spécialisé dans la détection d’anomalies. Fonctionne en isolant les observations via des arbres aléatoires ; les anomalies s’isolent plus vite que les points normaux.

- **One-Class SVM**

  Variante de SVM pour détecter les anomalies en apprenant uniquement sur les données "normales" et en identifiant les points qui s’en éloignent.

- **Autoencoder**

  Réseau de neurones entraîné à reconstruire ses entrées. Les erreurs de reconstruction importantes indiquent souvent des anomalies. Très puissant pour les données complexes ou non structurées.

Annexe 2 : Techniques d'évaluation de performance

2.1. Métriques de classification

L'évaluation des performances d'un modèle de classification repose sur plusieurs métriques statistiques. Ces métriques permettent de quantifier la qualité des prédictions en comparant les classes prédites aux classes réelles. On utilise généralement les quantités suivantes :

- **TP (True Positives)** : nombre de vrais positifs
- **TN (True Negatives)** : nombre de vrais négatifs
- **FP (False Positives)** : nombre de faux positifs
- **FN (False Negatives)** : nombre de faux négatifs

Les principales métriques utilisées sont :

- **Précision (Precision)** : mesure la proportion de prédictions positives correctes parmi toutes les prédictions positives effectuées par le modèle.

$$\text{Précision} = \frac{TP}{TP + FP}$$

- **Rappel (Recall ou Sensibilité)** : indique la proportion d'éléments réellement positifs qui ont été correctement identifiés par le modèle.

$$\text{Rappel} = \frac{TP}{TP + FN}$$

- **F1-score** : correspond à la moyenne harmonique entre la précision et le rappel. Elle permet d'obtenir un compromis entre ces deux mesures, surtout en cas de déséquilibre entre classes.

$$F1\text{-score} = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

- **Taux de faux positifs (FPR - False Positive Rate)** : mesure la proportion de négatifs incorrectement classés comme positifs.

$$\text{FPR} = \frac{FP}{FP + TN}$$

- **Exactitude (Accuracy)** : représente la proportion globale de bonnes prédictions (positives et négatives) par rapport à l'ensemble des prédictions.

$$\text{Exactitude} = \frac{TP + TN}{TP + TN + FP + FN}$$

Annexe 3 : Extraits de code source

Cette annexe présente quelques extraits représentatifs du code source développé lors de l'implémentation du système proposé.

1. Nettoyage des données

- ### 1.1. Nettoyage général

```
# Étape 1 : Supprimer les doublons en ignorant 'IP' et 'Date'
df_cleaned = df.drop_duplicates(subset=["Method", "URL", "Protocol", "Status", "Size", "Referer", "User-Agent", "Label"])

# Étape 2 : Gérer les valeurs manquantes, remplacer les valeurs manquantes par 'NaN' dans les colonnes autres que 'IP' et 'Date'
df_cleaned.loc[:, ["Method", "URL", "Protocol", "Status", "Size", "Referer", "User-Agent", "Label"]] = df_cleaned[["Method", "URL", "Protocol", "Status", "Size", "Referer", "User-Agent", "Label"]].fillna('NaN')

# Vérification après nettoyage
df_cleaned.isnull().sum()  # Vérifier si des valeurs manquantes subsistent
```

- ### 1.2. Nettoyage des paramètres d'URL (URL-PARAM)

```
import json
import math

def dict_to_str(val):
    if isinstance(val, dict):
        return json.dumps(val, sort_keys=True)
    if val is None or (isinstance(val, float) and math.isnan(val)):
        return "NaN"
    return str(val)

data['URL_params_str'] = data['URL_params'].apply(dict_to_str)

data = data.drop_duplicates(subset=['URL_params_str'], keep='first')

data = data.drop(columns=['URL_params_str'])
```

2. Parsing des fichiers journaux

```
# Regex
pattern = r'^(\S+) - - \[(.*?)\] "(\w+) (.*?) ([Hh][Tt][Tt][Pp]/[\d.]+)" (\d{3}) (\S+) "(.*?)" "(.*?)" \((.*?)\)$'

data = []
for line in lignes:
    match = re.match(pattern, line.strip())
    if match:
        ip = match.group(1)
        date = match.group(2)
        method = match.group(3)
        url = match.group(4)
        protocol = match.group(5)
        status = int(match.group(6))
        size = match.group(7)
        referer = match.group(8)
        user_agent = match.group(9)
        label = match.group(10)
        size = 0 if size == '-' else int(size)
        data.append([ip, date, method, url, protocol, status, size, referer, user_agent, label])

df = pd.DataFrame(data, columns=["IP", "Date", "Method", "URL", "Protocol", "Status", "Size", "Referer", "User-Agent", "Label"])

print(f"Nombre de lignes parsées : {len(df)}")
print(df.head())
```

3. Normalisation des données

- **3.1. Encodage avec LabelEncoder**

```
# Créer un mappage explicite des étiquettes
label_mapping = {
    'sql injection': 1,
    'command injection': 2,
    'lfi': 3,
    'xss': 4
}

# Appliquer ce mappage sur la colonne 'Label' pour créer la nouvelle colonne 'Label_mapped'
data['Label_mapped'] = data['Label'].map(label_mapping)

# Vérifier le résultat du mappage
print("\nAperçu des étiquettes après mappage :")
print(data[['Label', 'Label_mapped']].head())
```

- **3.2. Encodage avec OneHotEncoder**

```
from sklearn.preprocessing import OneHotEncoder

# Fixer les catégories à gérer
method_encoder = OneHotEncoder(categories=[['get', 'post']], sparse_output=False)

# Encodage One-Hot de la colonne 'Method'
method_encoded = method_encoder.fit_transform(data[['Method']])

# Création du DataFrame des colonnes encodées
method_encoded_df = pd.DataFrame(method_encoded, columns=method_encoder.categories_[0])

# Réinitialiser les index pour un alignement parfait
data = data.reset_index(drop=True)
method_encoded_df = method_encoded_df.reset_index(drop=True)

# Supprimer la colonne d'origine et concaténer
data = data.drop(columns=['Method'])
data = pd.concat([data, method_encoded_df], axis=1)
```

- **3.3. Mini-Max**

```
from sklearn.preprocessing import MinMaxScaler

# Création du scaler MinMax
scaler = MinMaxScaler()

# Normaliser la colonne 'Size' avec Min-Max
data['Size'] = scaler.fit_transform(data[['Size']])
```

4. Entraînement des modèles

```
from sklearn.neighbors import KNeighborsClassifier
import joblib

# Créer un modèle KNN avec des paramètres ajustés
knn_model = KNeighborsClassifier(n_neighbors=5, algorithm='auto', weights='uniform')  # Utilisation de k=5

# Entraîner le modèle avec les données d'entraînement
knn_model.fit(X_train, y_train)

# Enregistrer le modèle entraîné dans un fichier
joblib.dump(knn_model, 'knn_model_trained_improved.joblib')

print("Modèle KNN amélioré entraîné et enregistré sous 'knn_model_trained_improved.joblib'")
```

5. Prédiction et classification

```
from sklearn.neighbors import KNeighborsClassifier
import joblib
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from tabulate import tabulate
import matplotlib.pyplot as plt
import seaborn as sns

# Charger le modèle KNN entraîné
knn_model = joblib.load('knn_model_trained_improved.joblib')

# Faire des prédictions sur le dataset de test
y_pred_knn = knn_model.predict(X_test_data)

# Calculer l'accuracy (la précision globale du modèle)
accuracy_knn = accuracy_score(y_test_data, y_pred_knn)

# Générer un rapport de classification (précision, rappel, f1-score par classe)
report_knn = classification_report(y_test_data, y_pred_knn, output_dict=True)

# Générer la matrice de confusion (pour voir les erreurs de classification)
conf_matrix_knn = confusion_matrix(y_test_data, y_pred_knn)

# Affichage des résultats dans un format élégant
print(f"\n# Résultats de l'évaluation du modèle KNN #")
print(f"\nAccuracy sur le dataset de test : {accuracy_knn:.4f}\n")

# Affichage du rapport de classification sous forme de tableau bien organisé
print("Rapport de classification :")
headers = ["Classe", "Précision", "Rappel", "F1-score", "Support"]
table = []
```

5.Séléction dynamique des caractéristiques pour les classifieurs binaires :

```
sqli_features = ['Status', 'Size', 'suspect_params', 'param_count', 'UserAgent_class', 'UserAgent_length',
                 'get', 'post', 'http/1.1', 'select', 'union', 'insert', 'drop', 'update', 'delete', '--',
                 ';', ' or ', ' and ', 'sleep', 'benchmark', 'information_schema', 'concat', 'waitfor delay',
                 'having', 'group by', 'cast', 'convert', 'exec', 'execute', 'declare', 'set', 'fetch',
                 'sysusers', 'order by', 'limit', 'substring', 'if(', 'case', 'when', 'then', 'else', 'end',
                 'password']
```