Glossaire 3
Table des figures 4
1. Contexte et objectifs 5
1.1 Contexte du projet (cybersécurité, RGPD, monitoring) 5
1.2 Enjeux (surveillance, anonymisation, temps réel) 5
1.3 Objectif général 6
1.4 Objectifs techniques spécifiques 7
2. Architecture fonctionnelle et technique 8
2.1 Description globale du système 8
2.2 Schéma d'architecture (site Apache, traitement PyTorch) 8
2.3 Rôles des composants : 10
2.3.1. Proxmox (hébergement du site) 10
2.3.2. Apache (génération des logs) 12
2.3.3. Machine de Deep Learning (analyse via modèle autoencodeur) 12
2.4 Flux de données (collecte, parsing, vectorisation, détection) 13
3. Choix technologiques et justification 15
3.1 Analyse des solutions existantes (bibliographie simplifiée intégrée) 15
3.2 Raisons du choix : auto encodeur, PyTorch, logs Apache 16
3.3 Traitement non supervisé et anonymisation 17
3.4 Respect du RGPD 18
4. Implémentation et validation 19
4.1 Description de la collecte (access.log, parsing Python) 19
4.2 Préparation des données 20
4.3 Architecture du modèle autoencodeur 21
4.4 Méthode de détection (erreurs de reconstruction) 22
4.5 Résultats (exemples de logs, anomalies détectées) 23
5. Contraintes et sécurité 25
5.1 Contraintes réglementaires (non-intrusion, RGPD) 25
5.2 Contraintes techniques (temps réel, scalabilité) 25
5.3 Gestion des accès / protection des données 26
6. Perspectives d'évolution 27
6.1 Intégration continue (Kafka, dashboard) 27
6.2 Extension à d'autres types de logs (authentification, réseau) 28
6.3 Amélioration du modèle (LSTM, clustering, ring ring) 29
6.4 Réactions administrateur en fonction des anomalies détectées 30
7. Conclusion 32
8. Annexes 33

Glossaire

AD	Active directory
API	Application Programming Interface (Interface de Programmation d'Applications)
CNIL	Commission Nationale de l'Informatique et des Libertés
CPU	Central Processing Unit
DL	DeepLearning
GET/POST	Méthodes HTTP (lecture / envoi de données)
GUI	Graphical User Interface (Interface Graphique pour Utilisateurs)
IP	Internet Protocol
OS	Operating System (Système d'exploitation)
PyTorch	Framework Python de calcul tensoriel et de Deep Learning
RGPD	Règlement Général sur la Protection des Données
SCP	Secure Copy Protocol
SIEM	Security Information and Event Management
SSH	Secure Shell
URL	Uniform Resource Locator
VM	Virtual Machine
LDAP	Lightweight Directory Access Protocol

Table des figures

Figure 1 : Diagramme d'état de l'analyse du comportement utilisateur

Figure 2 : Schéma logique de l'infrastructure système

Figure 3 : Tableau récapitulatif des caractéristiques du serveur hôte

Figure 4 : Capture d'écran de l'interface utilisateur du Proxmox

Figure 5 : Tableau récapitulatif des caractéristiques du serveur DL

Figure 6 : Schéma du flux de données de l'utilisateur à la visualisation graphique de l'analyse comportementale

Figure 7 : Exemple d'une ligne de log Apache avant/après parsing

Figure 8 : Architecture du modèle autoencodeur utilisé

Figure 9 : Distribution des erreurs de reconstruction – seuil de détection en rouge

Figure 10 : Évolution du score d'anomalie au fil du temps - pics = comportements suspects

Figure 11 : Architecture d'une chaîne de transmission de logs, implémentant Kafka

Figure 12 : Schéma de fonctionnement de la solution Ring Attention PyTorch

1. Contexte et objectifs

1.1 Contexte du projet (cybersécurité, RGPD, monitoring)

Dans un contexte de multiplication des cyberattaques et d'augmentation constante des usages numériques, les systèmes d'information sont exposés à des risques croissants liés à l'usurpation d'identité, aux accès non autorisés et à la compromission des comptes utilisateurs. Face à ces menaces, les organisations doivent renforcer leurs capacités de détection comportementale, afin d'identifier rapidement les actions suspectes ou déviantes au sein de leur réseau.

Cependant, cette surveillance soulève des enjeux éthiques et juridiques majeurs, notamment en matière de protection des données personnelles. Le Règlement Général sur la Protection des Données (RGPD) impose des obligations strictes concernant la collecte, le traitement et la conservation des données identifiables. Toute solution de monitoring doit donc être conçue dans une logique de non-intrusion, en garantissant l'anonymat des utilisateurs et la minimisation des informations exploitées.

C'est dans ce cadre que s'inscrit le présent projet : il vise à démontrer qu'il est possible de détecter des anomalies comportementales à partir de logs techniques, sans recourir à des identifiants personnels, et en respectant les principes de confidentialité dès la conception. En s'appuyant sur des techniques de machine learning non supervisées, et sur des données issues de la navigation web (logs Apache), le système propose une alternative éthique aux solutions de sécurité intrusives, tout en répondant à des besoins concrets de supervision en environnement contrôlé.

1.2 Enjeux (surveillance, anonymisation, temps réel)

Le projet répond à un double enjeu, à la fois sécuritaire et réglementaire. Sur le plan sécuritaire, il s'agit de détecter des comportements anormaux susceptibles de révéler une tentative d'intrusion, une activité suspecte ou un usage détourné d'un compte utilisateur. Cette capacité de détection précoce est essentielle dans une stratégie de cybersécurité proactive, notamment dans des environnements où les systèmes ne disposent pas toujours d'un contrôle fort ou d'une supervision permanente.

Sur le plan réglementaire, l'enjeu est d'assurer une conformité stricte au RGPD, en développant une solution qui n'exploite aucune donnée à caractère personnel. Cela implique de concevoir un système capable d'observer des comportements sans les relier à une identité, ni conserver d'historique nominatif. L'objectif est donc de concilier sécurité et confidentialité, deux exigences souvent considérées comme antagonistes.

Ce projet propose ainsi un cadre de réponse technique et éthique à ces problématiques, en démontrant qu'il est possible d'analyser le comportement des utilisateurs de manière anonyme, transparente et non intrusive, tout en produisant des résultats exploitables pour l'identification d'anomalies.

1.3 Objectif général

L'objectif principal de ce projet est de concevoir et expérimenter un système de détection d'anomalies comportementales capable de repérer automatiquement des actions suspectes ou déviantes au sein d'un environnement web, tout en respectant les principes de non-intrusion et d'anonymisation.

Ce système repose sur l'analyse de données techniques issues des journaux Apache, transformées en vecteurs anonymes, et traitées à l'aide d'un modèle de machine learning non supervisé (ou autoencodeur, décrit en partie 4.3). L'ambition est de démontrer qu'il est possible de modéliser le comportement "normal" d'un utilisateur, puis d'identifier les écarts significatifs, sans recourir à des données personnelles ni à un historique individualisé.

À travers cette approche, le projet vise à proposer une solution modulaire, éthique et conforme au RGPD, tout en apportant une réponse concrète aux enjeux de cybersécurité comportementale dans les systèmes d'information modernes.

1.4 Objectifs techniques spécifiques

Pour répondre à l'objectif général de détection non-intrusive d'anomalies, le projet se décline en plusieurs objectifs techniques concrets :

Premièrement, il s'agit de développer un module de collecte capable d'extraire, à partir des fichiers access.log du serveur Apache, des données pertinentes pour l'analyse comportementale. Cette extraction doit se limiter à des champs non sensibles (horodatage, méthode HTTP, URL, code de réponse), dans le respect du principe de minimisation des données.

Deuxièmement, ces données doivent être vectorisées, normalisées et anonymisées, afin de permettre leur traitement par un modèle d'apprentissage automatique. Ce prétraitement doit garantir la neutralité des variables et empêcher tout retour vers une identité utilisateur.

Troisièmement, le cœur du système repose sur la mise en œuvre d'un modèle autoencodeur non supervisé, entraîné à reconstituer des comportements "normaux". Les écarts importants entre les données d'entrée et leur reconstruction seront utilisés pour identifier les anomalies.

Enfin, le système doit être capable de visualiser les résultats (par exemple via des histogrammes ou des courbes temporelles) et d'envisager une intégration future dans un pipeline temps réel, tout en maintenant l'ensemble des traitements dans un environnement local, sécurisé et conforme au RGPD.

2. Architecture fonctionnelle et technique

2.1 Description globale du système

Le système développé dans le cadre de ce projet R&D repose sur une chaîne de traitement répartie sur deux environnements techniques distincts :

- un serveur Apache, hébergé dans une VM Proxmox, qui héberge un site web et génère les fichiers de logs (access.log) au fur et à mesure de la navigation des utilisateurs ;

- un poste de calcul Deep Learning, configuré avec Python, PyTorch et les bibliothèques associées (Pandas, Matplotlib), qui récupère les logs depuis le serveur Apache, les traite, entraîne un modèle d'autoencodeur, et identifie les comportements anormaux via les erreurs de reconstruction.

L'ensemble du système repose sur une approche non supervisée, évitant l'usage de données étiquetées ou personnelles. Le traitement se fait localement, sur des vecteurs anonymisés représentant les requêtes HTTP (URL accédée, méthode, heure, code HTTP).

2.2 Schéma d'architecture (site Apache, traitement PyTorch)

L'architecture du système repose désormais sur une infrastructure composée de quatre briques techniques interdépendantes :

- un Active Directory (AD),
- un serveur Apache avec une base de données associée,
- une plateforme d'analyse (poste Deep Learning),
- et les utilisateurs accédant au site web via une interface.

Cette architecture simule un environnement réel dans lequel les utilisateurs sont authentifiés via un AD centralisé, leurs requêtes web sont enregistrées par Apache, et les données analysées pour identifier les comportements anormaux.

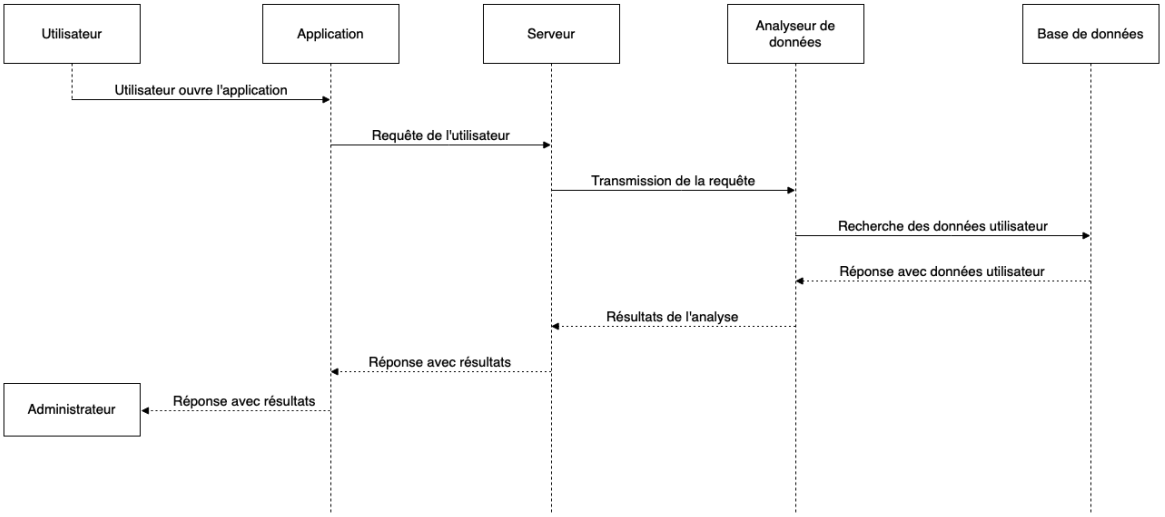

Figure 1 : Diagramme d'état de l'analyse du comportement utilisateur

D'un point de vue architecture technique, physique, la plateforme se définit grâce à ce schéma ci-dessous :

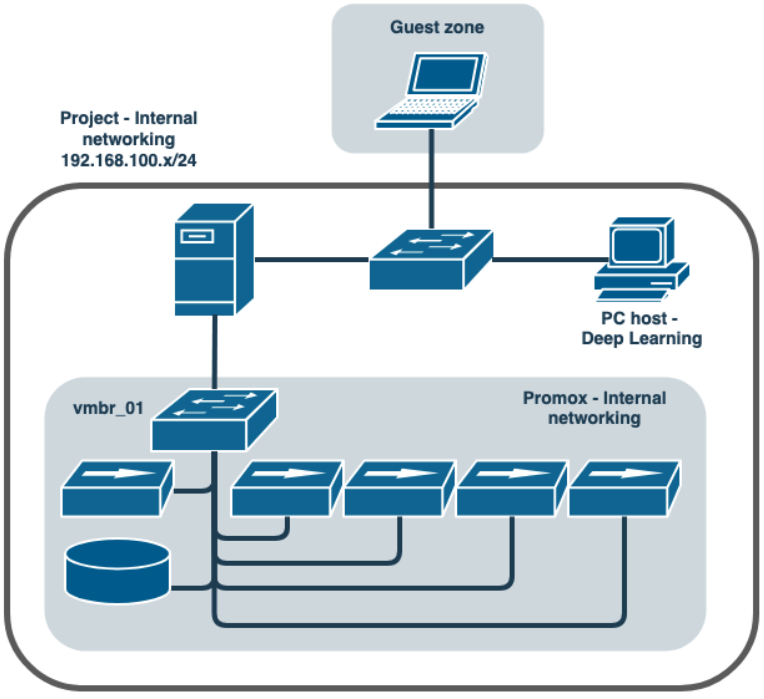

Figure 2 : Schéma logique de l'infrastructure système

Sur le plan physique, le système repose sur les éléments suivants :

- Un serveur Proxmox, qui héberge les différentes machines virtuelles nécessaires :
    - VM Apache (serveur web et base de données)
    - Contrôleur de domaine (Active Directory)
    - Base de données (MySQL)
- Un poste Deep Learning physique, isolé sur un sous-réseau sécurisé, pour l'analyse hors production.
- Une zone réseau utilisateur (guest zone), dans laquelle les postes clients accèdent à l'application web.
- Un switch d'interconnexion, permettant de relier physiquement tous les éléments au sein du même segment réseau, avec cloisonnements logiques éventuels.

2.3 Rôles des composants :

Comme dit précédemment, la plateforme comprend différentes parties techniques, pouvant être hardware comme software.

2.3.1. Proxmox (hébergement du site)

L'hyperviseur Proxmox Virtual Environment constitue le socle d'exécution de l'ensemble de l'architecture technique du projet. Il s'agit d'un hyperviseur de type 2, open-source, basé sur Debian et KVM, offrant à la fois robustesse, flexibilité et facilité d'administration.

Caractéristiques techniques du serveur hébergeant l'hyperviseur :

CPU (Coeurs)	2x12
RAM (Go)	64
Stockage en SSD (To)	4

Figure 3 : Tableau récapitulatif des caractéristiques du serveur hôte

Le rôle de Proxmox dans ce projet est central. Il héberge de manière mutualisée les services critiques du système, à commencer par le serveur Apache, qui assure l'hébergement de l'application web et la génération des journaux d'accès. Il héberge également la base de données PostgreSQL, utilisée pour stocker les éléments dynamiques de l'application, ainsi qu'un contrôleur de domaine Active Directory, chargé de la gestion des identités et de l'authentification des utilisateurs au niveau système comme au niveau applicatif. Toutes ces machines virtuelles fonctionnent de manière isolée mais

interconnectée, selon une logique réseau définie par des ponts virtuels configurés dans Proxmox.

Pour illustrer ces propos, voici une capture d'écran de l'interface utilisateur (GUI) :

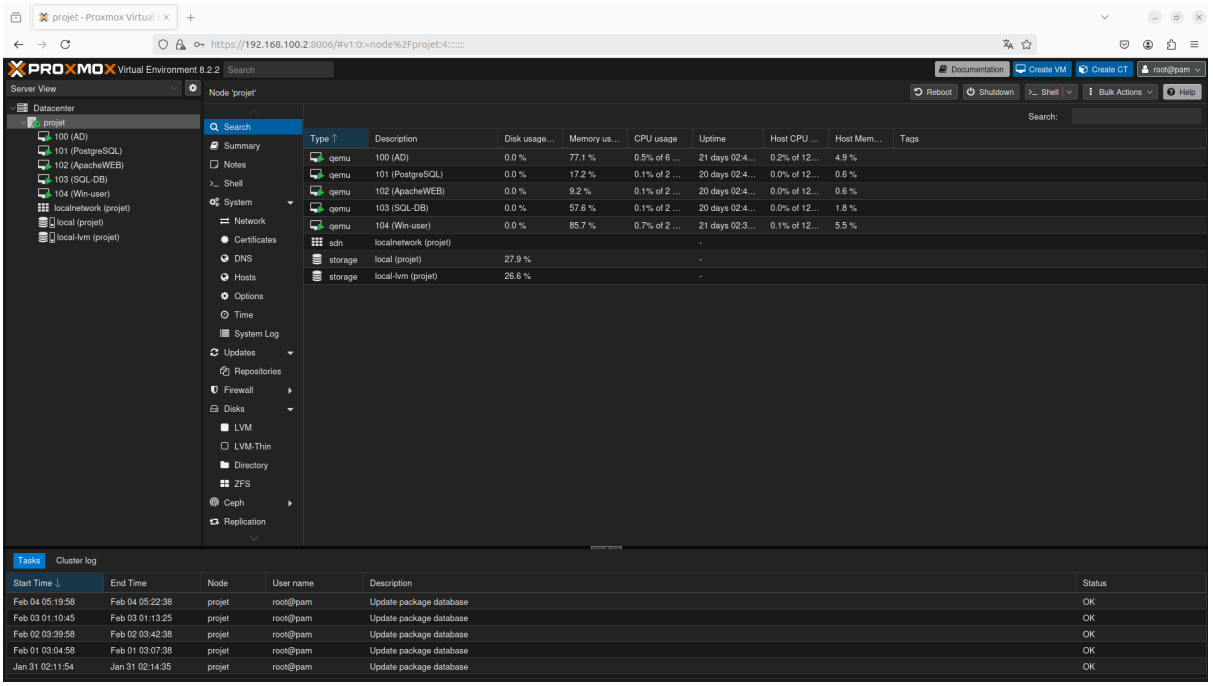

Figure 4 : Capture d'écran de l'interface utilisateur du Proxmox

L'accès à la plateforme Proxmox se fait via une interface web sécurisée en HTTPS, accessible à l'adresse IP 192.168.100.2. Cette interface permet à l'administrateur de gérer l'ensemble des VM, de surveiller leur consommation de ressources, de réaliser des snapshots, ou encore de planifier des arrêts ou redémarrages à chaud. Ce niveau d'abstraction et de contrôle rend Proxmox particulièrement adapté à des environnements de test comme celui du présent projet, où l'agilité, la réversibilité et l'isolation sont essentielles.

Le choix de Proxmox s'est imposé naturellement, notamment en raison de sa gratuité, de sa compatibilité avec des systèmes Linux et Windows, et de son intégration facile avec les outils standards de l'écosystème open-source. Il permet d'assurer la cohérence, la stabilité et la sécurité de l'infrastructure, tout en laissant à l'équipe projet une autonomie totale dans la gestion du cycle de vie des services déployés.

2.3.2. Apache (génération des logs)

Le serveur Apache constitue le point d'entrée fonctionnel du système. Déployé sur une machine virtuelle sous Proxmox, il héberge l'application web consultée par les utilisateurs depuis la guest zone. À chaque interaction, Apache enregistre automatiquement une ligne dans le fichier access.log, comprenant la méthode HTTP, l'URL demandée, le code de réponse et l'horodatage.

Ces journaux sont la source principale de données comportementales utilisées dans l'analyse. Les informations extraites sont strictement techniques et anonymisées avant traitement, conformément aux exigences du RGPD.

Le serveur est également configuré pour s'interfacer avec un Active Directory via LDAP, permettant une authentification centralisée, sans exposition des identités dans les logs analysés. Il est relié à une base PostgreSQL locale, qui alimente dynamiquement l'application web mais n'est pas directement exploitée dans la détection.

En somme, Apache joue un rôle essentiel de collecte passive, générant les traces nécessaires à l'étude du comportement utilisateur, sans intrusion ni captation sensible.

2.3.3. Machine de Deep Learning (analyse via modèle autoencodeur)

La machine Deep Learning constitue l'unité d'analyse du système. Il s'agit d'un poste physique, indépendant de l'infrastructure virtualisée, connecté au même réseau interne.

Caractéristiques technique du poste de Deep Learning :

CPU (Coeurs)	24
RAM (Go)	32
Stockage (To)	1,4

Figure 5 : Tableau récapitulatif des caractéristiques du serveur DL

Ces caractéristiques lui permettent d'exécuter efficacement des traitements lourds sur des volumes importants de logs.

Son rôle est de traiter les fichiers access.log générés par Apache, de les vectoriser, puis d'entraîner un modèle d'autoencodeur PyTorch pour modéliser le comportement utilisateur standard. Toute dérive significative par rapport à ce comportement est interprétée comme une anomalie potentielle.

Les traitements sont réalisés avec Python et les bibliothèques Pandas, scikit-learn, Matplotlib et PyTorch. Des visualisations sont automatiquement générées pour permettre l'analyse des scores d'anomalie dans le temps.

Cette machine fonctionne en mode batch, sans connexion continue au serveur Apache : les logs lui sont transférés manuellement ou via SCP. Isolée du système de production, elle garantit un traitement sécurisé, non intrusif et conforme au RGPD.

2.4 Flux de données (collecte, parsing, vectorisation, détection)

Le système repose sur une chaîne de traitement linéaire, structurée autour d'un flux de données logiques partant de l'activité utilisateur jusqu'à la détection d'anomalies. L'ensemble du flux est conçu pour respecter les principes de non-intrusivité, de traitement local et de séparation des responsabilités entre les composants.

Lorsqu'un utilisateur accède au site web hébergé sur le serveur Apache, chaque requête HTTP est automatiquement consignée dans le fichier access.log. Ce fichier contient des informations techniques telles que l'URL demandée, la méthode (GET, POST), le code de réponse du serveur et l'horodatage. Ces données sont collectées de manière passive, sans que l'utilisateur en soit conscient, et sans exploitation directe de données personnelles.

À intervalles réguliers, ce fichier est transféré vers la machine d'analyse via un canal sécurisé (typiquement en scp). Ce transfert peut être déclenché manuellement, ou automatisé à l'aide d'un script de synchronisation.

Une fois le fichier récupéré, la machine Deep Learning exécute un traitement en plusieurs étapes : parsing du fichier, extraction des champs utiles, vectorisation des données sous forme normalisée, puis passage dans un modèle d'autoencodeur PyTorch. Ce modèle, préalablement entraîné à reconnaître les comportements standards, attribue à chaque ligne de log un score d'anomalie basé sur l'erreur de reconstruction.

Ces scores sont ensuite utilisés pour générer des visualisations temporelles et des histogrammes, facilitant l'interprétation. L'ensemble du traitement s'effectue hors production, sans rétroaction directe sur les utilisateurs ou l'application web.

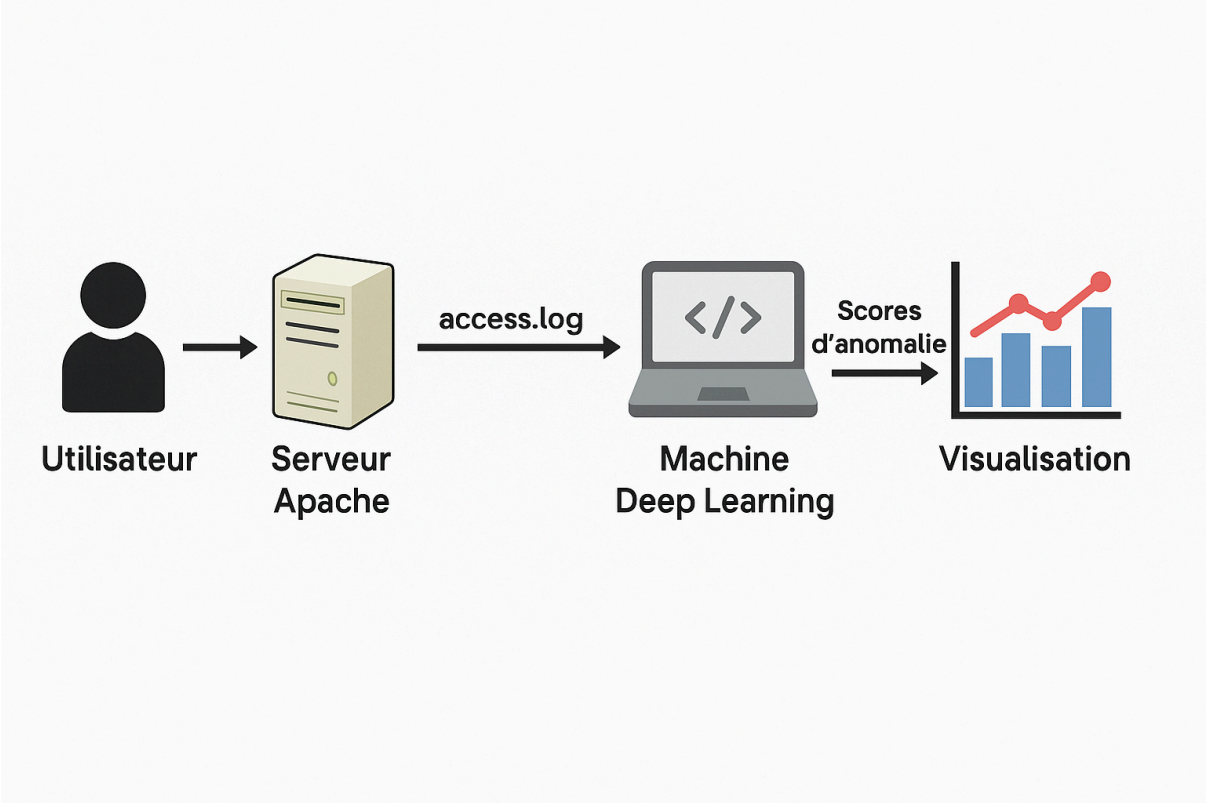

Figure 6 : Schéma du flux de données de l'utilisateur à la visualisation graphique de l'analyse comportementale

3. Choix technologiques et justification

3.1 Analyse des solutions existantes (bibliographie simplifiée intégrée)

La détection d'anomalies comportementales fait l'objet de nombreuses recherches dans les domaines de la cybersécurité, de l'analyse de logs et de la sécurité des systèmes d'information. Plusieurs approches sont documentées dans la littérature, allant de méthodes statistiques classiques à des solutions complexes basées sur l'intelligence artificielle.

Parmi les solutions les plus couramment mises en œuvre, on retrouve les systèmes UEBA (User and Entity Behavior Analytics), qui permettent de détecter les déviations de comportement par rapport à un profil d'activité attendu. Ces solutions, bien que efficaces dans des environnements industriels, sont souvent coûteuses, opaques dans leur fonctionnement et parfois intrusives, car elles manipulent des données personnelles sensibles. Elles nécessitent souvent une infrastructure lourde et une supervision permanente, ce qui les rend peu compatibles avec les contraintes pédagogiques et réglementaires du projet.

Dans le domaine académique, des travaux proposent des approches probabilistes, notamment la modélisation bayésienne hiérarchique des événements d'authentification Windows. Cette méthode consiste à regrouper les utilisateurs selon leurs profils d'activité, puis à détecter ceux qui s'écartent significativement du groupe. Bien que performante, cette approche reste centrée sur les systèmes d'authentification, et nécessite des données historiques importantes, parfois sensibles.

D'un autre côté, des analyses des comportements clients sur des sites web marchands ont été faites, à l'aide de deux modèles hybrides : l'un basé sur les motifs séquentiels (SEPM), l'autre sur des séries temporelles multivariées (GCBag). Ces approches montrent que l'exploitation des journaux de navigation permet d'identifier des comportements d'achat ou de consultation anormaux, mais elles sont fortement dépendantes du contenu métier du site et moins adaptées à une généralisation.

Enfin, d'un point de vue réglementaire, l'utilisation de données comportementales soulève des questions de conformité au Règlement Général sur la Protection des Données (RGPD). Toute solution impliquant l'analyse de logs utilisateurs doit intégrer des mécanismes d'anonymisation, de minimisation des données, et garantir l'absence de corrélation directe entre les données techniques et l'identité de l'utilisateur.

Au regard de ces éléments, le choix du projet s'est porté sur une solution légère, locale, non supervisée et non intrusive, exploitant un modèle d'auto encodage développé en PyTorch, capable d'apprendre les comportements standard sans avoir besoin de données étiquetées. Cette approche permet une détection souple des anomalies tout en garantissant une conformité maximale aux exigences de confidentialité.

3.2 Raisons du choix : auto encodeur, PyTorch, logs Apache

Le choix d'un auto encodeur comme modèle central de détection repose avant tout sur les contraintes fonctionnelles, techniques et réglementaires du projet. L'objectif étant de modéliser le comportement utilisateur standard à partir de logs Apache bruts, sans disposer de données étiquetées ni de vérité terrain sur ce qui constitue une "anomalie", une approche non supervisée s'est naturellement imposée.

L'auto encodeur est un type de réseau de neurones particulièrement bien adapté à ce contexte. Il permet d'apprendre une représentation compressée des données normales, puis de mesurer l'écart entre une entrée et sa reconstruction. Un comportement inhabituel produit une erreur de reconstruction plus importante, ce qui permet de le détecter sans avoir à définir explicitement des règles. Cette méthode offre ainsi une solution souple, sans dépendance à un modèle métier, et généralisable à divers environnements.

L'implémentation du modèle repose sur la bibliothèque PyTorch, choisie pour sa simplicité, sa flexibilité et son intégration naturelle avec l'environnement Python. PyTorch permet un contrôle fin de l'architecture, de l'entraînement et de l'inférence, tout en assurant une compatibilité avec les visualisations via Matplotlib. Par ailleurs, l'écosystème Python offre un ensemble cohérent d'outils complémentaires, tels que :

- Pandas, pour le traitement et le nettoyage des données extraites des fichiers logs
- scikit-learn, pour la normalisation des vecteurs d'entrée
- Matplotlib, pour la production de graphiques d'analyse (histogrammes d'erreurs, évolution temporelle des scores d'anomalie)

Ce choix technique garantit un traitement local, autonome et reproductible, ne nécessitant ni infrastructure serveur, ni connexion cloud, ni stockage centralisé. Il respecte également les contraintes RGPD, puisque seuls des vecteurs dérivés de données techniques anonymisées sont traités, sans jamais relier une requête à une identité.

3.3 Traitement non supervisé et anonymisation

L'un des fondements du projet est le respect strict du principe de non-intrusion, tant sur le plan technique que réglementaire. Cela implique que le système ne doit ni identifier, ni tracer, ni profiler directement les utilisateurs du site web. En cohérence avec cette exigence, le traitement des données repose exclusivement sur des informations techniques anonymisables, issues des journaux Apache.

Le traitement est dit non supervisé, car il ne s'appuie sur aucun étiquetage préalable des données. Aucune distinction explicite entre comportements normaux et anormaux n'est nécessaire : le modèle autoencodeur apprend lui-même à reconstruire les comportements les plus fréquents, et considère comme suspect tout écart significatif par rapport à ce qu'il a appris. Ce fonctionnement permet de détecter des anomalies comportementales de manière autonome, sans intervention humaine ni base d'exemples.

L'anonymisation est assurée dès l'étape de parsing des logs. Les champs utilisés dans l'analyse sont volontairement limités à :

- l'URL accédée (sous forme de code ou label),
- la méthode HTTP (GET, POST),
- le code de réponse HTTP (200, 403, 404, etc.),
- l'heure d'accès, arrondie à l'heure pleine pour éviter toute corrélation précise.

Les champs sensibles tels que l'adresse IP, le user-agent ou l'identifiant de session sont exclus du traitement. Les données sont vectorisées et normalisées, ce qui rend impossible tout retour inverse vers l'utilisateur d'origine. En complément, un système d'anonymisation avancée de type confidentialité différentielle (tel que RAPPOR) pourrait être envisagé dans une version future, afin de renforcer la robustesse de l'approche en cas de mutualisation des données.

Grâce à cette architecture, le projet respecte pleinement les principes du RGPD, notamment :

- la minimisation des données (seules les données strictement nécessaires sont traitées),
- la limitation des finalités (détection d'anomalies uniquement),
- l'absence de profilage personnel identifiable,
- et la protection des données par conception.

3.4 Respect du RGPD

Le projet a été conçu pour respecter strictement les principes du Règlement Général sur la Protection des Données (RGPD), en particulier ceux de minimisation des données, de non-identifiabilité et de traitement loyal (article 5, §1 du RGPD).[1]

Aucune donnée personnelle identifiable n'est exploitée. Les champs utilisés pour l'analyse (heure, méthode HTTP, code de réponse, URL simplifiée) ne permettent pas, en l'état, d'identifier un utilisateur, directement ou indirectement. Les adresses IP, user-agents ou identifiants de session sont volontairement exclus, conformément aux recommandations de la CNIL sur les traitements de logs web [CNIL, 2021].[2]

Le traitement est local, ponctuel et non corrélé : les logs sont transférés temporairement sur une machine isolée pour entraînement et détection, puis supprimés. Aucune base de données centralisée n'est constituée, ni aucun historique utilisateur.

Cette approche respecte les principes de privacy by design et démontre qu'il est possible d'appliquer des techniques de machine learning à l'analyse comportementale sans porter atteinte à la vie privée.

[1] Règlement (UE) 2016/679 du Parlement européen et du Conseil (RGPD), article 5
[2] CNIL, « Les bonnes pratiques pour l'exploitation des logs », 2021

4. Implémentation et validation

4.1 Description de la collecte (access.log, parsing Python)

La collecte des données comportementales repose exclusivement sur le fichier access.log généré automatiquement par le serveur Apache. Ce journal contient, pour chaque requête utilisateur, des informations techniques telles que la méthode HTTP utilisée (GET, POST...), l'URL demandée, le code de réponse du serveur (200, 404, etc.), ainsi qu'un horodatage complet.

Voici maintenant un exemple des lignes présentes dans le fichier access.log

```
192.168.100.15 - - [15/05/2025:14:21:06 +0200] "GET /index.html HTTP/1.1" 200 1024 "https://example.com/home" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"
```

Figure 7 : Exemple d'une ligne de log Apache avant/après parsing

Le fichier est récupéré depuis la machine virtuelle Apache via un transfert sécurisé (scp) vers la machine Deep Learning. Une fois transféré, il est traité à l'aide d'un script Python développé sur mesure, basé sur le module "re" (expressions régulières) pour extraire les lignes pertinentes.

Le parsing du log consiste à identifier, pour chaque ligne :

- la méthode HTTP,
- l'URL appelée,
- le code de statut HTTP,
- l'heure de la requête (convertie en heure pleine).

Ces données sont ensuite structurées dans un DataFrame Pandas, normalisées et converties en vecteurs numériques. À ce stade, aucune donnée personnelle n'est retenue : les adresses IP, les user-agents et les identifiants de session sont ignorés.

Cette phase constitue l'entrée du pipeline de traitement, et a été conçue pour être automatisable, légère et conforme aux exigences de non-intrusivité du projet.

4.2 Préparation des données

Après l'extraction des champs pertinents via parsing, les données issues du fichier access.log sont transformées en vecteurs exploitables pour l'apprentissage automatique. Cette étape consiste à convertir les éléments textuels ou catégoriels en valeurs numériques, à normaliser les échelles, et à s'assurer de la cohérence du format d'entrée du modèle.

Concrètement, les champs suivants sont traités :

- La méthode HTTP est convertie en valeur entière (ex. : GET = 0, POST = 1).
- Le code de statut HTTP (ex. 200, 403, 404) est conservé tel quel, en tant que valeur numérique.
- L'URL demandée est encodée via un LabelEncoder, ce qui permet de transformer chaque chaîne de caractères unique en un identifiant entier.
- L'heure de la requête, extraite depuis l'horodatage, est simplifiée à l'heure pleine (valeurs de 0 à 23) pour capturer les habitudes temporelles.

Une fois les colonnes converties, elles sont normalisées avec MinMaxScaler, afin d'uniformiser leur échelle sur l'intervalle [0, 1]. Ce prétraitement est essentiel pour garantir que le modèle ne privilégie pas un champ numérique au détriment des autres lors de l'apprentissage.

Les données ainsi préparées sont stockées dans un tenseur PyTorch, format attendu par le modèle autoencodeur. Ce tenseur constitue le jeu d'entraînement et de test utilisé pour identifier les schémas comportementaux typiques.

Cette phase de préparation garantit la neutralité du traitement, tout en assurant une représentation uniforme et anonyme du comportement utilisateur, indispensable au bon fonctionnement du modèle.

4.3 Architecture du modèle autoencodeur

Le cœur du système de détection repose sur un autoencodeur, un type de réseau de neurones non supervisé conçu pour apprendre une représentation compacte des données d'entrée. L'objectif est de reconstruire des comportements considérés comme "normaux", et d'identifier comme anomalies ceux qui génèrent une erreur de reconstruction élevée.

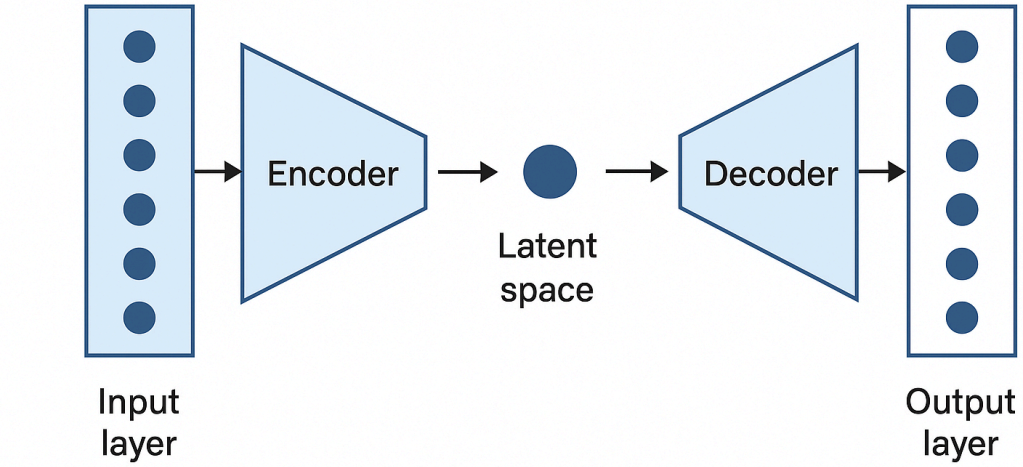

Figure 8 : Architecture du modèle autoencodeur utilisé

Le modèle développé dans ce projet a été implémenté en PyTorch, et se compose de deux parties principales :

- L'encodeur, qui réduit la dimension d'entrée :
    - Couche 1 : 4 neurones d'entrée (correspondant aux 4 attributs comportementaux),
    - Couche 2 : 3 neurones (ReLU),
    - Couche 3 : 2 neurones (représentation latente).
- Le décodeur, qui reconstruit les données à partir de la représentation comprimée :
    - Couche 4 : 3 neurones (ReLU),
    - Couche 5 : 4 neurones de sortie (mêmes dimensions que l'entrée initiale).

L'activation ReLU est utilisée dans les couches intermédiaires pour introduire de la non-linéarité, tandis que la dernière couche est linéaire. Le modèle est entraîné à minimiser l'erreur quadratique moyenne (MSE) entre l'entrée et la sortie.

Ce réseau simple mais efficace permet d'apprendre les patterns dominants dans les comportements utilisateurs, sans supervision ni catégorisation. Lors de la phase d'inférence, chaque nouvelle requête est reconstruite par le modèle ; si l'erreur dépasse un seuil défini dynamiquement (moyenne + 2 écarts-types), le comportement est signalé comme anormal.

Cette architecture légère a été choisie pour son efficacité en contexte local, sa lisibilité pédagogique, et sa capacité à modéliser des comportements discrets sur des données structurées comme les logs web.

4.4 Méthode de détection (erreurs de reconstruction)

La détection des comportements anormaux repose sur une approche quantitative, basée sur l'erreur de reconstruction produite par le modèle autoencodeur. L'idée est simple : un comportement "habituel" est bien reconstruit par le modèle, tandis qu'un comportement "atypique" produit une reconstruction de mauvaise qualité, traduite par une erreur élevée.

Une fois le modèle entraîné sur les données considérées comme représentatives du comportement normal, chaque nouvelle requête issue du fichier access.log est transformée en vecteur, passée à travers l'autoencodeur, puis comparée à sa propre reconstruction.

L'erreur est mesurée à l'aide de la moyenne quadratique des écarts (MSE) entre les valeurs d'entrée et de sortie. Plus cette erreur est élevée, plus le comportement est considéré comme anormal.

Pour différencier les comportements normaux des anomalies, un seuil dynamique est défini à partir de la distribution des erreurs sur les données d'entraînement :

- seuil = moyenne des erreurs + 2 écarts-types.

Ce seuil permet de prendre en compte la variabilité naturelle des données tout en filtrant les cas extrêmes. Toutes les requêtes dont l'erreur dépasse ce seuil sont étiquetées comme "anormales".

Ce mécanisme de détection est entièrement automatisé, reproductible et transparent. Il permet de visualiser précisément quand et pourquoi un comportement a été classé comme suspect, en s'appuyant uniquement sur des données techniques anonymisées.

4.5 Résultats (exemples de logs, anomalies détectées)

L'évaluation du système a été menée à partir de deux types de jeux de données : un fichier access.log collecté en condition réelle sur le serveur Apache, et un fichier enrichi synthétiquement contenant des comportements volontairement anormaux (requêtes POST sur des URLs interdites, pics d'activité à des heures inhabituelles, codes HTTP inhabituels).

L'analyse a permis de constater que le modèle autoencodeur, entraîné sur des comportements réguliers, détecte correctement les écarts injectés. Les erreurs de reconstruction liées à ces requêtes déviantes dépassent nettement le seuil dynamique calculé sur le jeu d'apprentissage.

Pour valider visuellement les résultats, deux représentations graphiques ont été produites à l'aide de Matplotlib :

- Un histogramme des erreurs de reconstruction, montrant la distribution globale des erreurs. Les requêtes anormales apparaissent à l'extrémité droite du graphique, bien au-delà du seuil fixé.

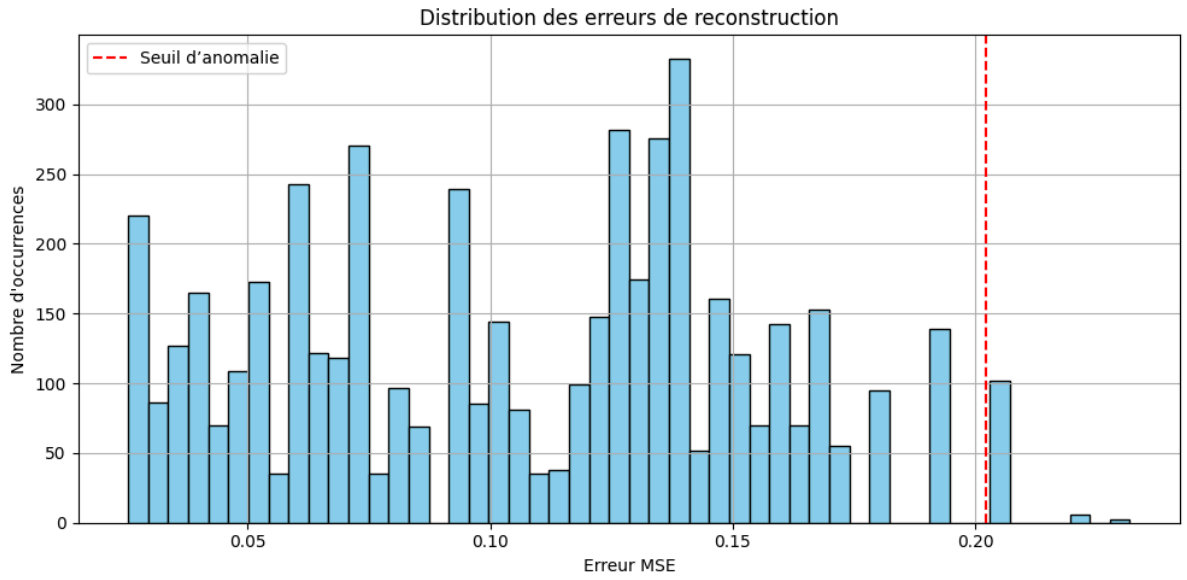

Figure 9 : Distribution des erreurs de reconstruction – seuil de détection en rouge

- Une courbe temporelle des scores d'anomalie, mettant en évidence les pics de comportement suspect dans la chronologie des requêtes. Chaque pic correspond à une action considérée comme inhabituelle par le modèle.

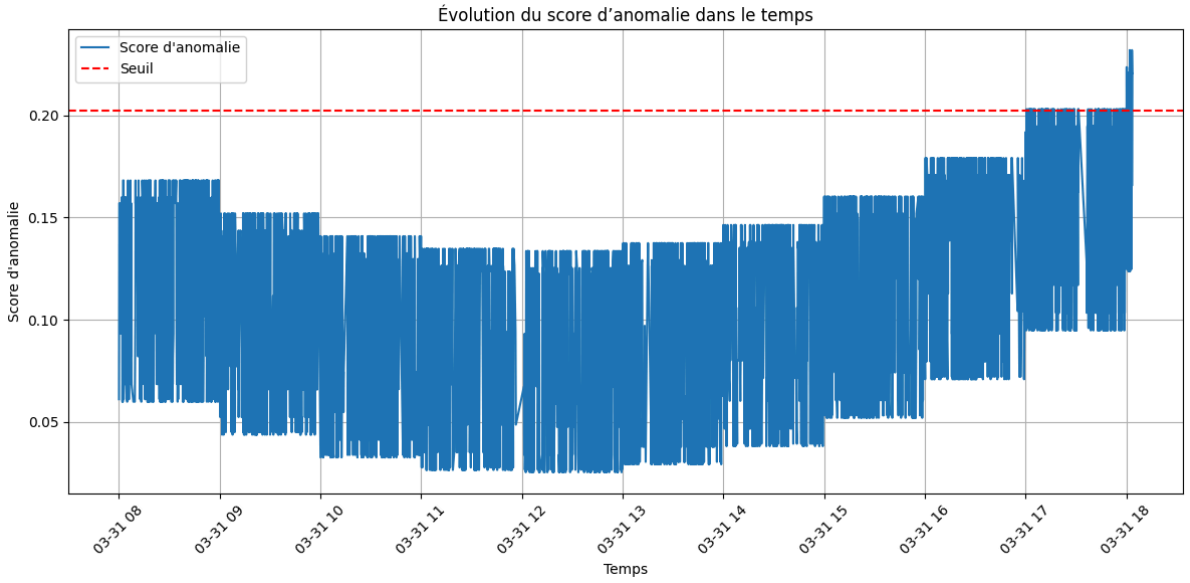

Figure 10 : Évolution du score d'anomalie au fil du temps - pics = comportements suspects

Ces visualisations confirment la capacité du système à distinguer les comportements standards des anomalies, de manière automatique, sans supervision, et sans compromettre la confidentialité des utilisateurs. Elles renforcent la lisibilité des résultats et permettent une interprétation claire des sorties du modèle.

5. Contraintes et sécurité

5.1 Contraintes réglementaires (non-intrusion, RGPD)

L'une des contraintes majeures du projet réside dans le respect strict du cadre juridique imposé par le Règlement Général sur la Protection des Données (RGPD). Étant donné que le système repose sur l'analyse de journaux issus de la navigation web, il était impératif de s'assurer qu'aucune information ne permette d'identifier directement ou indirectement un utilisateur.

Pour cela, le traitement a été conçu selon une logique de non-intrusion, en appliquant les principes du privacy by design (article 25 du RGPD)[3]. Aucun élément personnel n'est collecté : les adresses IP, les identifiants de session ou les chaînes User-Agent sont systématiquement exclus du traitement. De plus, les requêtes sont analysées uniquement à travers des attributs techniques génériques (heure, méthode, code HTTP, URL encodée), qui ne permettent aucun recoupement individuel.

La nature non supervisée et locale du traitement renforce encore la conformité : les données sont analysées ponctuellement, sur une machine isolée, sans stockage externe ni historisation, ce qui limite fortement les risques liés à la sécurité ou à l'usage détourné des informations.

5.2 Contraintes techniques (temps réel, scalabilité)

Sur le plan technique, le projet est confronté à deux contraintes principales : la réactivité du système face aux événements en temps réel, et sa capacité à traiter un volume d'activité croissant sans dégradation des performances.

Bien que l'architecture actuelle fonctionne en mode batch, avec des transferts manuels de fichiers de logs (access.log), l'objectif initial visait un traitement en temps réel ou quasi temps réel. Cela impose des limites liées à la fréquence de collecte, à la vitesse de parsing et à la rapidité de réponse du modèle de détection.

[3] Règlement (UE) 2016/679, article 25

Par ailleurs, la scalabilité est un enjeu dès lors que le nombre d'utilisateurs simultanés augmente. Le traitement vectoriel et l'apprentissage par autoencodeur doivent être capables de gérer plusieurs milliers de requêtes par heure, ce qui nécessite des optimisations à plusieurs niveaux :

- prétraitement parallèle des lignes de logs,
- traitement par mini-batch dans PyTorch,
- éventuelle exploitation de GPU pour accélérer les inférences.

Dans sa forme actuelle, le système reste adapté à une échelle de démonstration, mais son extension à un contexte réel impliquerait une refonte partielle du pipeline, notamment en ce qui concerne la gestion des flux, la persistance des alertes et l'intégration avec un système de supervision centralisé.

5.3 Gestion des accès / protection des données

Afin de garantir la sécurité de l'ensemble du système, une attention particulière a été portée à la gestion des accès et à la protection des données à chaque étape du traitement. L'architecture a été conçue de manière à limiter l'exposition des données sensibles, même si celles-ci sont anonymisées.

Seules deux machines accèdent directement aux fichiers de logs : le serveur Apache (qui les génère) et la machine Deep Learning (qui les traite). Le transfert de données se fait via un canal chiffré SSH, en utilisant la commande scp. Aucune donnée n'est transmise en clair sur le réseau, et aucun service tiers (cloud ou API externe) n'est impliqué.

Sur la machine d'analyse, les fichiers sont stockés localement, temporairement, dans un répertoire restreint, accessible uniquement à l'utilisateur système dédié au traitement. Les résultats (scores d'anomalie, graphiques) sont générés sous forme de fichiers .png sans journalisation des données sources. Ce choix garantit un traitement éphémère, isolé, et non historisé.

La machine Deep Learning n'est pas connectée à Internet durant les phases critiques d'analyse, réduisant considérablement les risques d'exfiltration de données ou d'interception

6. Perspectives d'évolution

6.1 Intégration continue (Kafka, dashboard)

Dans sa version actuelle, le système fonctionne en mode batch : les fichiers de logs sont collectés manuellement ou de façon périodique, puis traités par lots sur la machine d'analyse. Bien que suffisant pour une phase de démonstration, ce mode de fonctionnement présente des limites dès lors que l'on souhaite évoluer vers une surveillance continue ou temps réel. Une première piste d'amélioration serait donc l'intégration d'un système de traitement en flux, reposant sur des outils comme Apache Kafka.

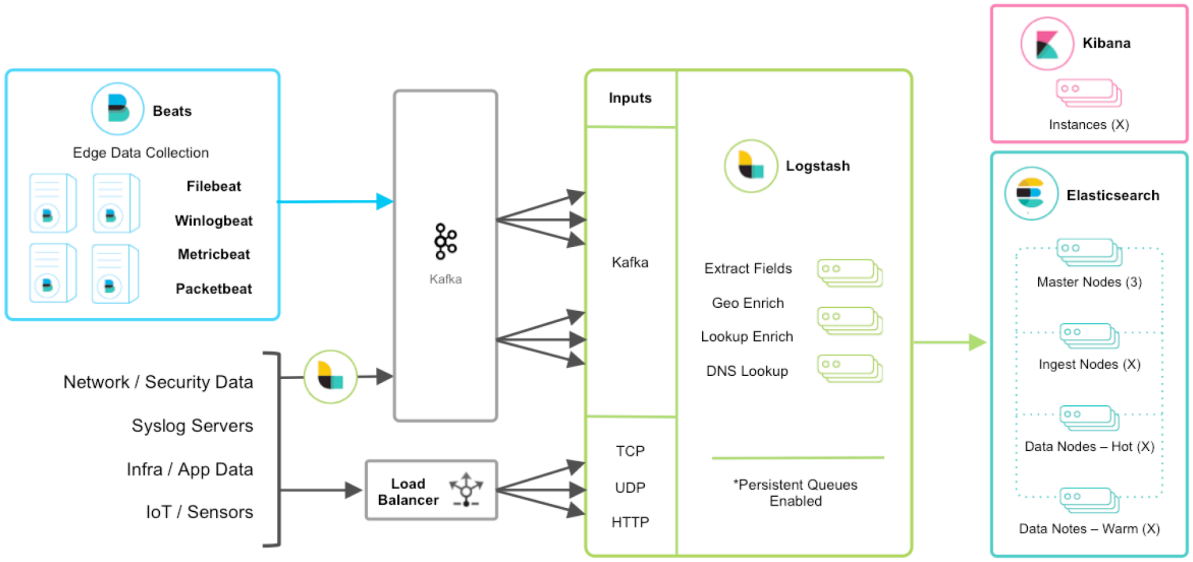

Figure 11 : Architecture d'une chaîne de transmission de logs, implémentant Kafka

Kafka permettrait de transformer chaque ligne de log en un événement, capté à la volée dès sa génération par Apache. Ces messages seraient directement envoyés vers un module d'analyse, chargé de parser et de vectoriser les données, puis de les faire transiter par le modèle de détection. Ce fonctionnement ouvrirait la voie à une détection immédiate des anomalies, sans phase de transfert intermédiaire ni traitement différé.

En parallèle, une seconde amélioration envisageable concerne la visualisation dynamique des résultats. Actuellement, les anomalies sont exportées sous forme de graphiques statiques. L'ajout d'un dashboard interactif (développé avec Streamlit, Dash ou Grafana) permettrait à un administrateur de visualiser en temps réel les scores d'anomalie, les requêtes suspectes, ou les courbes

d'activité agrégée. Ce tableau de bord pourrait aussi intégrer des filtres, un historique, et un système de notification ou d'export automatique.

En combinant ces deux évolutions – Kafka en entrée, dashboard en sortie – le système passerait d'une preuve de concept isolée à une chaîne d'intégration continue, adaptée à un environnement de supervision de la sécurité en production.

6.2 Extension à d'autres types de logs (authentification, réseau)

Le système développé pour les fichiers access.log peut être facilement adapté à d'autres types de journaux, dès lors qu'ils présentent une structure exploitable. Grâce à sa logique modulaire, la chaîne de traitement peut être réutilisée pour d'autres sources avec peu d'ajustements.

Une première extension naturelle concerne les logs d'authentification (Windows, Linux, Active Directory). En analysant les horaires, les adresses IP d'origine ou les échecs de connexion, il serait possible de détecter des comportements de connexion anormaux, tels que des tentatives de brute force ou des connexions hors plage habituelle.

Le système pourrait aussi être appliqué aux logs réseau, comme ceux d'un pare-feu ou d'un proxy. Ces journaux révèlent des schémas d'accès, de trafic sortant, ou de tentatives de scan, qui pourraient être modélisés comme des comportements standards.

Enfin, une intégration dans un outil centralisé comme Wazuh ou ELK Stack permettrait d'unifier la collecte et d'étendre la portée du modèle, tout en conservant une analyse locale, respectueuse de la confidentialité.

6.3 Amélioration du modèle (LSTM, clustering, ring ring)

L'autoencodeur utilisé dans ce projet constitue une solution simple et efficace pour détecter des anomalies sans supervision. Toutefois, plusieurs pistes d'amélioration peuvent être envisagées pour renforcer sa robustesse et sa sensibilité.

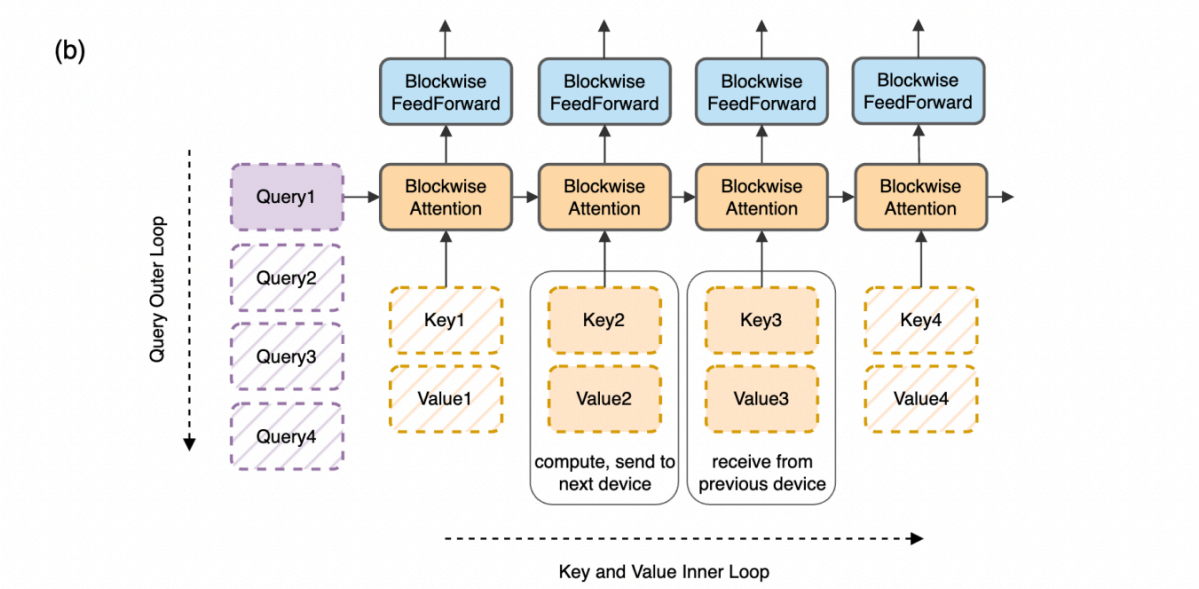

Figure 12 : Schéma de fonctionnement de la solution Ring Attention PyTorch

Une première évolution technique consisterait à utiliser un autoencodeur récurrent (RNN ou LSTM), capable de prendre en compte l'ordre temporel des requêtes. Cette approche permettrait d'analyser des séquences complètes de comportements et non plus des événements isolés, en identifiant des anomalies de contexte ou de fréquence.

Une autre possibilité serait d'introduire un mécanisme de seuil dynamique adaptatif, reposant sur des statistiques glissantes ou un clustering préalable. Cela permettrait d'ajuster automatiquement la sensibilité du modèle en fonction de l'évolution des comportements globaux.

Enfin, l'intégration d'un module d'explicabilité (par exemple via SHAP ou LIME) offrira une lecture plus fine des anomalies détectées, facilitant leur interprétation par un analyste ou un administrateur.

Ces évolutions renforceraient la valeur du modèle dans un environnement réel, tout en conservant les garanties de confidentialité et d'autonomie du système initial.

6.4 Réactions administrateur en fonction des anomalies détectées

L'un des prolongements fonctionnels envisagés pour le système concerne l'exploitation active des anomalies détectées par l'infrastructure d'analyse comportementale. Au-delà de la visualisation graphique des scores d'anomalie, il est pertinent de définir un ensemble de réactions automatisables ou assistées, permettant à un administrateur de réagir en temps réel ou en différé, selon le type d'écart observé.

Chaque catégorie d'anomalie peut ainsi donner lieu à des actions spécifiques, à la fois proportionnées, réversibles et conformes au principe de non-intrusion :

- Anomalies de navigation incohérente

Les séquences de navigation anormales (accès à des pages sans logique apparente, rupture dans les parcours standards, absence de pages intermédiaires attendues) peuvent révéler un script automatisé ou une tentative de contournement de l'interface.

Réactions possibles :

- ❖ Renforcement temporaire des journaux pour la session concernée (niveau de log augmenté) ;
- ❖ Redirection vers une page neutre ou intermédiaire, incluant un contrôle d'intégrité ou une confirmation d'intention (ex. captcha) ;
- ❖ Notification à l'administrateur via un canal sécurisé (mail, webhook, SIEM).

- Activité excessive ou séquentielle

Une fréquence élevée de requêtes, la répétition d'URLs identiques ou une volumétrie anormale en un court laps de temps peuvent indiquer une activité de type scan, brute-force ou scraping.

Réactions possibles :

- ❖ Application de règles de throttling (limitation du débit de requêtes) via Apache ou un proxy inverse ;
- ❖ Enregistrement d'une trace d'audit complète pour post-analyse.

- Accès à des ressources sensibles ou obsolètes

L'accès à des chemins connus pour être critiques ou obsolètes (/admin, /old-login, /backup) peut constituer un indicateur de reconnaissance ou d'attaque ciblée.

Réactions possibles :

- ❖ Journalisation dédiée avec extraction vers un canal SIEM ou SOC ;
- ❖ Renforcement des règles d'accès à ces ressources (authentification multifacteur, alerte instantanée) ;
- ❖ Blocage conditionnel basé sur des patterns de navigation contextualisés.

- Comportement hors plage horaire

Une navigation à des horaires atypiques (nuit, week-end, heures creuses) sur un site normalement actif en journée peut constituer un indicateur d'usage détourné ou de compte compromis.

Réactions possibles :

- ❖ Génération d'une alerte en dehors de la plage horaire définie comme normale ;
- ❖ Application de mesures compensatoires (par exemple, désactivation temporaire des actions critiques) ;
- ❖ Archivage sécurisé et horodaté des journaux associés pour analyse ultérieure.

Ces actions peuvent être mises en œuvre via des scripts de supervision locaux, des outils de sécurité réseau (pare-feu, IDS/IPS), ou encore par l'intégration dans un système centralisé de gestion des événements (type Wazuh, ELK Stack). À terme, cette capacité de réaction ciblée transforme le système en véritable outil de décision pour l'administrateur, combinant détection précoce, visualisation intuitive et réponse adaptée aux écarts comportementaux.

Bien sûr, puisque l'ensemble des données analysées sont anonymisées, ces actions sont plus difficiles à qualifier et à appliquer sur les équipements, pages ou logiciels spécifiques au comportement anormal détecté.

7. Conclusion

Ce projet R&D a permis de concevoir, développer et expérimenter un système de détection d'anomalies comportementales respectueux de la vie privée, basé sur l'analyse de journaux web générés par un serveur Apache. En s'appuyant sur un modèle d'autoencodeur non supervisé implémenté en PyTorch, le système a démontré sa capacité à identifier automatiquement des comportements déviants, sans nécessiter de données personnelles ni d'étiquetage préalable.

La mise en œuvre technique, réalisée dans un environnement virtualisé via Proxmox, a permis de structurer une architecture modulaire, réaliste et reproductible, incluant une machine d'analyse dédiée, un Active Directory pour la centralisation des identités, et une interface web génératrice de logs. L'ensemble du traitement a été pensé selon les principes du RGPD, avec un strict respect de la non-intrusion et de l'anonymisation des données.

Les résultats obtenus, validés à travers des tests sur des jeux de données réels et simulés, ont permis de visualiser clairement la détection des écarts comportementaux. Cette preuve de concept pose les bases d'une solution plus large, pouvant évoluer vers des traitements en flux (Kafka), des visualisations temps réel (dashboard), et des extensions à d'autres types de logs (authentification, réseau, supervision).

Au-delà de ses résultats techniques, le projet illustre la possibilité de concilier sécurité informatique et respect de la vie privée, en proposant une approche éthique, explicable et opérationnelle de l'analyse comportementale en environnement numérique.

8. Annexes

Nom d'usage	Connexion	IP	Activité	CPU	RAM	Stockage	OS
Proxmox	HTTP(s)	192.168.100.2	Full-Time	2x12	64	4000	Linux
AD	RDP	192.168.100.11	Full-Time	1	4	40	Windows Server
SNMP	SSH	192.168.100.X	Full-Time				Linux
Apache	SSH	192.168.100.13	Full-Time	1	2	30	Linux
PostgreSQL	SSH	192.168.100.12	Full-Time	1	2	50	Linux
Ubuntu Deep Learning	Physique	192.168.100.25	Full-Time	6x4	32	1400	Linux
Poste client	RDP	192.168.100.15	Sous-demande	2	4	20	Windows

Annexe 1 : Listing de l'ensemble des VM's du projet + caractéristiques techniques de chacunes d'entre elles

```
import re
from datetime import datetime

pattern = re.compile(r'(?P<ip>\\S+) \\S+ \\S+ \\[(?P<datetime>.*?)\\]
"(?P<method>\\S+) (?P<url>\\S+) \\S+" (?P<status>\\d{3}) (?P<size>\\d+|-)')

with open("access.log", "r") as file:
        for line in file:
        match = pattern.match(line)
        if match:
        data = match.groupdict()
        dt = datetime.strptime(data["datetime"].split()[0], "%d/%b/%Y:%H:%M:%S")
        print(dt, data["method"], data["url"], data["status"])
```

Annexe 2 : Extrait du code python pour le parsing des logs Apaches

Objectif = extraire depuis un fichier access.log les champs nécessaires à l'analyse sans stocker d'IP ni d'identifiants.

```
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
import pandas as pd
import torch

df["hour"] = df["datetime"].dt.hour
df["method_enc"] = df["method"].map({"GET": 0, "POST": 1}).fillna(2)
df["url_enc"] = LabelEncoder().fit_transform(df["url"])

features = df[["url_enc", "hour", "method_enc", "status"]].values
scaled = MinMaxScaler().fit_transform(features)
X_tensor = torch.tensor(scaled, dtype=torch.float32)
```

Annexe 3 : Script pour la préparation des vecteurs pour l'analyse

Objectif = transformer les données textuelles et temporelles en vecteurs numériques normalisés pour l'entraînement du modèle

```
import matplotlib.pyplot as plt

# Erreur de reconstruction calculée
errors = torch.mean((X_tensor - model(X_tensor)) ** 2, dim=1).numpy()
threshold = errors.mean() + 2 * errors.std()

plt.hist(errors, bins=50, color="skyblue")
plt.axvline(threshold, color="red", linestyle="--", label="Seuil")
plt.title("Distribution des erreurs de reconstruction")
plt.xlabel("Erreur MSE")
plt.ylabel("Occurrences")
plt.legend()
plt.show()
```

Annexe 4 : Script pour la visualisation des anomalies via matplot

Objectif = visualiser la séparation entre comportements normaux et comportements détectés comme anomalies