



ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE ET D'ANALYSE DES  
SYSTÈMES - RABAT

DEPARTEMENT WEB AND MOBILE ENGINEERING

Filière : IDSIT

## Rapport De Projet Base Données NOSQL

---

**Détection et prévention des menaces internes  
dans les SI à l'aide de l'apprentissage  
automatique et des bases NoSQL**

---

*Réalisé par :*

AIT MOUCH HOUDA  
SERRAJ ANDALOUSSI  
RANYA

MAHMOUDI ANAS  
GHATTRIF REDOUANE

Année Scolaire 2024/2025

---

## Résumé :

Dans un contexte où la cybersécurité est devenue un enjeu stratégique pour les entreprises, les menaces internes représentent un danger croissant. Contrairement aux attaques externes, les menaces internes proviennent d'utilisateurs autorisés, ce qui les rend particulièrement difficiles à détecter et à prévenir. Ce projet propose une solution intelligente basée sur l'apprentissage automatique pour surveiller et analyser les comportements des utilisateurs au sein d'un système d'information.

Nous avons conçu une architecture complète allant de la collecte de journaux d'activités jusqu'à la détection automatique des anomalies comportementales. Plusieurs algorithmes de machine learning ont été testés, notamment Random Forest, Autoencoders, et Isolation Forest. Les modèles ont été évalués à l'aide de données issues du CERT Insider Threat Dataset, avec des résultats très prometteurs (jusqu'à 92% de précision pour certains modèles).

La solution développée constitue un outil efficace et proactif permettant d'identifier et de prévenir les menaces internes au sein des systèmes d'information, grâce à une surveillance intelligente et continue des comportements suspects.

Ce projet ouvre des perspectives intéressantes vers une cybersécurité renforcée, autonome et pilotée par l'intelligence artificielle.

---

# Liste des abréviations

Abréviation	Désignation
SI	Systèmes d'Information
IA	Intelligence Artificielle
ML	Machine Learning (Apprentissage automatique)
RFC	Random Forest Classifier
SVM	Support Vector Machine
KNN	K-Nearest Neighbors
IF	Isolation Forest
OC-SVM	One-Class SVM
XGB	XGBoost (Extreme Gradient Boosting)
AE	Autoencodeur
DBSCAN	Density-Based Spatial Clustering of Applications with Noise

# Table des figures

3.1	les noeuds du graphe . . . . .	31
3.2	exemple de cinq noeuds du graphe . . . . .	32
3.3	Compter les nœuds par label . . . . .	33
3.4	Relations PERFORMS entre utilisateurs et activités . . . . .	34
3.5	Relations ACCESSES entre systèmes et ressources . . . . .	35
3.6	Relations USES entre utilisateurs et systèmes . . . . .	36
3.7	Relations INVOLVES entre activités et ressources . . . . .	37
3.8	Chemin complet des interactions . . . . .	38
3.9	Chemin complet des interactions . . . . .	39
3.10	Top ressources accédées . . . . .	40
3.11	Activités exécutées une seule fois par un utilisateur . . . . .	41
3.12	Accès suspects hors horaires . . . . .	42
3.13	Accès rares à des ressources . . . . .	43
3.14	Détection de mouvement latéral . . . . .	44
3.15	Top 5 des utilisateurs les plus actifs . . . . .	46
3.16	Calcul de l'entropie des activités . . . . .	47
4.17	Activités après heures . . . . .	50
4.18	Entropie des ressources vs Nombre de ressources uniques . . . . .	51
4.19	Distribution des caractéristiques . . . . .	52
4.20	Matrice de corrélation comportementale . . . . .	53
4.21	Entropie vs Nombre d'activités . . . . .	54
4.22	Centralité de degré par utilisateur . . . . .	55
4.23	Tableau de bord d'analyse des comportements utilisateurs . . . . .	56
4.24	Matrices de confusion et courbes ROC . . . . .	61
4.25	Caption . . . . .	63
4.26	Matrices de confusion par méthode de détection . . . . .	65
4.27	Courbes ROC des différents modèles de détection d'anomalies . . . . .	67
4.28	Comparaison des métriques de performance par modèle . . . . .	68

# Table de matières

<b>Introduction Générale</b>	<b>10</b>
<b>Chapitre I : Contexte du travail</b>	<b>12</b>
1.1 Importance du Machine Learning et des bases NoSQL . . . . .	14
1.1.1 Apports du Machine Learning . . . . .	14
1.1.2 Avantages des bases NoSQL . . . . .	14
1.2 Contributions attendues . . . . .	15
1.2.1 Détection efficace des comportements suspects . . . . .	15
1.2.2 Expérimentation comparative . . . . .	15
1.3 Conclusion . . . . .	15
<b>Chapitre II : État de l'art</b>	<b>17</b>
2.1 Revue systématique de la littérature scientifique . . . . .	18
2.1.1 Méthodologie de recherche documentaire . . . . .	18
2.1.2 Synthèse des travaux existants . . . . .	19
2.2 Menaces internes et risques pour les SI . . . . .	20
2.2.1 Définition et taxonomie des menaces internes . . . . .	20
2.2.2 Vecteurs d'attaque et comportements à risque . . . . .	20
2.3 Approches ML pour la cybersécurité . . . . .	21
2.3.1 Modèles supervisés pour la détection des menaces . . . . .	21
2.3.2 Apprentissage non supervisé et détection d'anomalies . . . . .	22
2.3.3 Approches hybrides et ensemblistes . . . . .	22
2.4 Bases NoSQL pour l'analyse des comportements . . . . .	23
2.4.1 Avantages des bases NoSQL pour la cybersécurité . . . . .	23
2.4.2 Neo4j et l'analyse de graphes pour la détection . . . . .	23
2.4.3 Cassandra pour le stockage et l'analyse de logs . . . . .	24
2.4.4 Vulnérabilités spécifiques aux bases NoSQL . . . . .	25
2.5 Défis et perspectives . . . . .	25
2.5.1 Limitations des approches actuelles . . . . .	25
2.6 Conclusion . . . . .	26
<b>Chapitre III : Méthodologie et Conception</b>	<b>27</b>
3.1 Choix technologiques . . . . .	28
3.1.1 Justification de Neo4j pour la modélisation des relations . . . . .	28
3.1.2 Justification de Cassandra pour le stockage des logs et événements . . . . .	29
3.2 Modélisation des interactions internes . . . . .	30
3.2.1 Construction du graphe utilisateur → actions → systèmes accédés . . . . .	30

3.2.2 Exploration et validation du modèle . . . . .	31
1.Analyse des Données via Requêtes Cypher dans Neo4j . . . . .	31
Détection de comportements suspects . . . . .	40
3.3 Conclusion . . . . .	47
<b>Chapitre IV : Implémentation et Expérimentations</b>	<b>48</b>
4.1 Introduction . . . . .	49
4.2 Développement du système de détection . . . . .	49
4.2.1 Collecte et structuration des données . . . . .	49
4.2.2 Exploration des Données et Comportements Utilisateurs . . . . .	50
4.3 Algorithmes de détection basés sur Neo4j et ML . . . . .	57
4.3.1 Implémentation du système de détection . . . . .	57
Conclusion : . . . . .	64
4.4 Introduction aux Métriques d'Évaluation . . . . .	65
4.5 Conclusion . . . . .	70
<b>Chapitre V : Résultats et Discussion</b>	<b>71</b>
5.1 Introduction . . . . .	72
5.2 Analyse des résultats et limitations . . . . .	72
5.2.1 Analyse des résultats . . . . .	72
5.2.2 Limitations . . . . .	73
5.3 Comparaison avec les solutions traditionnelles (SIEM, IDS) . . . . .	74
5.3.1 Systèmes IDS (Intrusion Detection Systems) . . . . .	75
5.3.2 Systèmes SIEM (Security Information and Event Management) . . . . .	75
5.3.3 Comparaison avec notre approche . . . . .	76
5.3.4 Avantages spécifiques de notre approche . . . . .	77
5.3.5 Conclusion de la comparaison . . . . .	78
<b>Chapitre VI : Conclusion et Perspectives</b>	<b>79</b>
6.1 Applications possibles en entreprise et améliorations futures . . . . .	80
6.1.1 Améliorations futures . . . . .	80
6.2 Conclusion . . . . .	80
<b>Conclusion générale</b>	<b>81</b>

# Introduction Générale

Face à l'évolution constante du paysage de la cybersécurité, les organisations font face à une menace souvent sous-estimée mais particulièrement dévastatrice : les menaces internes. Contrairement aux attaques externes qui tentent de franchir les périmètres de défense, les menaces internes émanent d'individus ayant déjà accès légitime aux systèmes d'information. Ces utilisateurs privilégiés qu'ils soient motivés par la malveillance, le gain financier, ou simplement victimes de négligence représentent un défi majeur pour les systèmes traditionnels de détection.

Les statistiques récentes sont alarmantes : selon le rapport Ponemon Institute de 2023, le coût moyen d'un incident lié à une menace interne atteint 15,4 millions de dollars, avec un temps de détection moyen de 85 jours. Plus inquiétant encore, 62% des incidents de sécurité impliquent, directement ou indirectement, des acteurs internes. Cette réalité s'explique par la position privilégiée de ces acteurs qui connaissent les systèmes, comprennent les contrôles de sécurité, et peuvent opérer sous couvert de légitimité.

Les approches conventionnelles de détection se révèlent souvent inadéquates face à ce type de menaces. Les systèmes SIEM (Security Information and Event Management) et IDS (Intrusion Detection System) traditionnels, conçus principalement pour identifier des signatures d'attaques connues ou des violations de politique explicites, peinent à distinguer un comportement malveillant subtil d'une activité légitime inhabituelle. Cette limitation fondamentale appelle à un changement de paradigme dans les méthodes de détection.

C'est dans ce contexte que l'émergence des bases de données NoSQL et des techniques avancées de machine learning ouvre de nouvelles perspectives. Les bases de données orientées graphes comme Neo4j offrent une capacité inédite à modéliser les relations complexes entre utilisateurs, ressources et actions, tandis que les systèmes comme Cassandra permettent le stockage et l'analyse efficace de volumes massifs de données d'événements. Cette combinaison technologique permet d'aborder la détection des menaces internes sous un angle radicalement différent : non plus comme une recherche de violations de règles, mais comme l'identification de patterns

comportementaux anormaux dans un réseau complexe d'interactions.

Le présent projet se propose d'explorer cette approche novatrice en développant un système de détection des menaces internes basé sur l'analyse comportementale avancée. Notre travail s'articule autour de trois objectifs principaux :

1. Concevoir et implémenter une architecture NoSQL (Neo4j) capable de modéliser efficacement les comportements des utilisateurs internes dans leur contexte organisationnel.
2. Développer et comparer des algorithmes de détection d'anomalies, tant supervisés que non supervisés, pour identifier les comportements potentiellement malveillants avec une précision accrue et un minimum de faux positifs.
3. Évaluer l'efficacité spécifique des approches basées sur l'analyse de graphes pour la détection de patterns relationnels subtils qui échapperaient aux méthodes conventionnelles.

À travers ce projet, nous ambitionnons de contribuer à l'évolution des systèmes de détection des menaces internes, en démontrant comment l'alliance des technologies NoSQL et du machine learning peut aboutir à des solutions plus efficaces, adaptatives et contextuelles face à ce défi majeur de la cybersécurité moderne.

# Chapitre I

## Introduction

## Problématique

Les menaces internes représentent aujourd’hui l’un des risques les plus insidieux et difficiles à contrer pour les organisations. Contrairement aux attaques externes qui se heurtent généralement à plusieurs couches de défense périphérique, les menaces internes proviennent d’individus disposant déjà d’accès légitimes aux systèmes d’information. Ces utilisateurs privilégiés - qu’il s’agisse d’employés, de contractants, ou de partenaires commerciaux - peuvent contourner de nombreuses mesures de sécurité traditionnelles grâce à leurs accréditations existantes, rendant leur détection particulièrement complexe. Selon le rapport Ponemon Institute de 2023, le coût moyen d’un incident lié à une menace interne a augmenté de 34% par rapport à 2020, atteignant 15,4 millions de dollars par organisation. Plus alarmant encore, le temps moyen de détection de ces incidents dépasse 85 jours, période durant laquelle les acteurs malveillants peuvent causer des dommages considérables aux données, à la propriété intellectuelle et à la réputation de l’entreprise. La difficulté fondamentale réside dans la distinction entre comportements légitimes et malveillants, ces derniers utilisant souvent des priviléges autorisés mais dans des contextes inhabituels ou pour des actions suspectes. Les systèmes de gestion des risques SI traditionnels se montrent largement inefficaces face à ce défi particulier pour plusieurs raisons :

1. Accès légitimes : Les utilisateurs internes possèdent des droits d'accès par nécessité professionnelle, rendant difficile la distinction entre usage normal et abusif.

2. Connaissance interne : Les acteurs malveillants connaissent les systèmes, politiques et procédures de sécurité, leur permettant d'éviter les détections basiques.

3. Variété des motifs : Les menaces peuvent résulter de malveillance intentionnelle, mais aussi de négligence, d'erreurs humaines ou de compromission d'identifiants.

4. Complexité comportementale : La détection nécessite une compréhension nuancée des comportements humains et des variations légitimes dans les habitudes de travail.

Face à ces défis, les approches traditionnelles basées sur des règles statiques et des signatures s'avèrent insuffisantes. Une nouvelle génération d'outils est nécessaire, capable d'analyser finement les comportements et d'identifier des anomalies subtiles qui signalent potentiellement une menace interne.

## 1.1 Importance du Machine Learning et des bases NoSQL

L'essor des techniques de Machine Learning (ML) et des bases de données NoSQL offre de nouvelles perspectives prometteuses pour la détection des menaces internes. Ces technologies sont particulièrement adaptées à ce défi pour plusieurs raisons complémentaires :

### 1.1.1 Apports du Machine Learning

Le ML permet de dépasser les limitations des approches traditionnelles en :

- Établissant des profils comportementaux dynamiques : Au lieu de règles rigides, les algorithmes d'apprentissage peuvent modéliser le comportement normal de chaque utilisateur et détecter des écarts significatifs.
- Identifiant des patterns complexes : Les techniques avancées comme les forêts aléatoires, XGBoost ou les réseaux de neurones peuvent découvrir des relations subtiles entre multiples variables qui échapperaient à l'analyse humaine.
- S'adaptant continuellement : Les modèles supervisés et non supervisés peuvent évoluer au fil du temps, intégrant les nouveaux comportements légitimes et identifiant de nouvelles formes d'attaques.
- Réduisant les faux positifs : Une limitation majeure des systèmes actuels, qui peuvent être significativement diminués grâce à des classifications plus précises et contextuelles.

### 1.1.2 Avantages des bases NoSQL

Parallèlement, les bases de données NoSQL apportent des capacités essentielles :

- Modélisation flexible des relations : Particulièrement avec les bases de données orientées graphes comme Neo4j, qui permettent de représenter naturellement les relations utilisateur-rôle-ressource et d'analyser leurs interactions.
- Traitement de grands volumes hétérogènes : Les bases NoSQL gèrent efficacement l'analyse de téraoctets de logs, événements et traces d'activité issus de sources diverses.

- Requêtes complexes optimisées : Des requêtes qui seraient prohibitives en SQL traditionnel peuvent être exécutées efficacement, notamment les analyses de chemins et de centralité dans les graphes.
- Évolutivité horizontale : Capacité critique pour absorber le volume croissant de données de sécurité générées par les environnements modernes.

## 1.2 Contributions attendues

Ce projet vise à apporter plusieurs contributions significatives au domaine de la cybersécurité et plus particulièrement à la détection des menaces internes :

### 1.2.1 Détection efficace des comportements suspects

### 1.2.2 Expérimentation comparative

Le projet conduira une évaluation rigoureuse de différentes approches analytiques :

- Comparaison systématique entre modèles supervisés (Random Forest, XGBoost) et non supervisés (clustering, isolation forest, autoencodeurs) pour la détection d'anomalies comportementales.
- Évaluation quantitative des performances selon des métriques pertinentes : précision, rappel, score F1, temps de détection et taux de faux positifs.
- Analyse de l'impact des différentes sources de données et features sur l'efficacité des modèles.

## 1.3 Conclusion

La synergie entre bases NoSQL et Machine Learning crée un paradigme particulièrement puissant pour la détection des menaces internes. Par exemple, Neo4j permet de modéliser finement les relations entre utilisateurs, ressources et actions, créant un graphe comportemental sur lequel

des algorithmes d'analyse de graphe peuvent identifier des anomalies structurelles. Cassandra, quant à elle, excelle dans le stockage et l'interrogation rapide d'historiques d'actions, permettant des analyses temporelles essentielles à la compréhension des comportements.

## **Chapitre II**

### **État de l'art**

## **2.1 Revue systématique de la littérature scientifique**

La détection des menaces internes à l'aide de bases NoSQL et de techniques d'apprentissage automatique représente un domaine de recherche en pleine expansion, situé à l'intersection de la cybersécurité et de l'intelligence artificielle. Cette convergence technologique répond à un besoin crucial dans un contexte où les architectures de données modernes s'orientent davantage vers des solutions NoSQL pour gérer d'importants volumes d'informations hétérogènes, tandis que les menaces internes demeurent parmi les vecteurs d'attaque les plus difficiles à détecter. Notre revue systématique vise à cartographier l'état actuel de la recherche en identifiant les avancées récentes, notamment dans l'application des techniques d'apprentissage supervisé et non supervisé, les défis persistants liés au manque de données d'entraînement de qualité, et les approches hybrides prometteuses combinant détection d'anomalies et analyse comportementale.

### **2.1.1 Méthodologie de recherche documentaire**

Notre revue de la littérature a été réalisée en utilisant les bases de données scientifiques Scopus et Web of Science, avec des recherches complémentaires sur ResearchGate pour identifier les publications les plus récentes. Les critères de sélection incluaient :

- Publications des années 2024-2025
- Focus sur les menaces internes et/ou les bases NoSQL
- Applications des techniques d'apprentissage automatique
- Études expérimentales avec résultats quantifiables.

## 2.1.2 Synthèse des travaux existants

TABLE 2.1 – Scientific Literature Review on Insider Threats and NoSQL

Reference	Year	Objectives	Tools and Approaches	Results	Problems	Perspectives
<b>MongoDB and NoSQL Injection Prevention</b> (Abuali)	2024	Analyze NoSQL injection attacks and prevention methods in MongoDB	Tautology attacks on OWASP Juice Shop, server-side validation, WAF, parameterized queries	Demonstrated effectiveness of server-side validation; multi-layered defense recommended	Performance overhead; implementation complexity	Balance security and performance; adopt hybrid defenses
<b>The Role of ML in Detecting Anomalies</b> (Oye et al.)	2024	Improve anomaly detection in cybersecurity using ML	Supervised (SVM, decision trees), unsupervised (clustering, autoencoders), real-time monitoring	Improved accuracy; reduced false positives	Data quality issues; model interpretability	Explainable AI (XAI); federated learning
<b>NoSQL and Machine Learning</b> (Ngalamulume & Ilunga)	2025	Explore NoSQL-ML synergies for big data applications	NoSQL (MongoDB, Cassandra), ML pipelines (Spark, TensorFlow)	Scalable architectures for fraud detection, recommendations, NLP	Data consistency; integration complexity	Optimize real-time processing; edge computing for IoT
<b>The MongoDB Injection Dataset</b> (Rameshwar et al.)	2024	Provide a dataset for NoSQL injection research	400 NoSQL commands (malicious/benign), similarity filtering	Dataset for ML model training; improved accuracy	Limited to MongoDB; requires augmentation	Extend to other NoSQL databases
<b>Understanding Insider Threats</b> (Ivan et al.)	2024	Model the evolution of insider threats	Adaptive network model, psychological trait analysis	Early detection; effective SOC interventions	Subjective trait evaluation	Integrate ML for dynamic prediction
<b>Mitigating Insider Threats</b> (Ashraf)	2024	Detect insider threats through behavioral analytics	ML algorithms, real-time monitoring	High accuracy; adaptable to new threats	Privacy concerns	Hybrid models; cloud scaling
<b>IJISRT24MAR127</b>	2024	Fraud detection in NoSQL databases	Neural networks, SVM, random forests, clustering, autoencoders	Accuracy > 99%, false positives at 0.2%	Lack of labeled data	Hybrid offline/online models

## 2.2 Menaces internes et risques pour les SI

### 2.2.1 Définition et taxonomie des menaces internes

Selon Ivan et al. (2024), les menaces internes peuvent être classifiées en plusieurs catégories distinctes :

Utilisateurs malveillants intentionnels :

1. Motivations financières (vente d'informations, fraude) Motivations personnelles (vengeance, idéologie) Espionnage industriel ou étatique

2. Utilisateurs négligents :

Erreurs d'utilisation des systèmes Non-respect des politiques de sécurité Victimes de social engineering

3. Comptes compromis :

Identifiants volés utilisés par des acteurs externes Élévation de priviléges après compromission initiale

4. Partenaires commerciaux et sous-traitants :

Tiers bénéficiant d'accès privilégiés Sécurité variable selon les organisations

Une menace interne se caractérise par l'utilisation d'accès légitimes pour des actions malveillantes ou inappropriées, ce qui rend sa détection particulièrement difficile par les méthodes traditionnelles de sécurité périmétrique.

### 2.2.2 Vecteurs d'attaque et comportements à risque

Ashraf (2024) identifie plusieurs comportements caractéristiques des menaces internes qui peuvent servir d'indicateurs pour les systèmes de détection :

- Accès anormaux à des données sensibles : Tentatives d'accéder à des informations sans rapport avec les fonctions professionnelles
- Activités hors des heures habituelles : Connexions à des moments inhabituels pour l'utilisateur ou le département

- Exfiltration de données : Téléchargements, impressions ou transferts anormaux de volumes importants d'informations
- Élévation de priviléges : Tentatives d'obtention d'accès supplémentaires via des vulnérabilités ou de l'ingénierie sociale
- Contournement des contrôles de sécurité : Désactivation des mécanismes de journalisation ou des antivirus
- Communication avec des serveurs inhabituels : Connexions à des domaines suspects ou non reconnus
- Modifications de code ou de configuration : Altérations non autorisées des applications ou des paramètres système

Ivan et al. (2024) soulignent également l'importance des facteurs psychologiques et comportementaux comme indicateurs précoce de risque, tels que les changements d'attitude au travail, les signes de mécontentement, ou des difficultés financières soudaines.

## 2.3 Approches ML pour la cybersécurité

### 2.3.1 Modèles supervisés pour la détection des menaces

Selon Oye et al. (2024), plusieurs algorithmes supervisés ont démontré leur efficacité dans l'identification des menaces internes :

- Random Forest : Ensemble d'arbres de décision capable de classifier les comportements avec une bonne résistance au surapprentissage. Particulièrement efficace pour gérer des métriques comportementales variées et des données bruitées.
- SVM (Support Vector Machines) : Performant pour établir des frontières de décision dans des espaces de caractéristiques à haute dimension, avec une capacité à distinguer les comportements normaux et anormaux même lorsque la séparation est complexe.
- XGBoost (Extreme Gradient Boosting) : Offre des performances supérieures sur les données déséquilibrées, un problème courant dans la détection des menaces où les comportements malveillants sont rares par rapport aux comportements normaux.

- Réseaux de neurones profonds : D'après IJISRT24MAR127 (2024), les approches par réseaux de neurones atteignent une précision exceptionnelle supérieure à 99% avec seulement 0,2% de faux positifs dans la détection de fraudes NoSQL, particulièrement lorsqu'elles sont appliquées aux séquences d'événements.

Ces approches supervisées nécessitent cependant des données d'entraînement étiquetées de haute qualité, ce qui constitue un défi majeur dans le domaine des menaces internes où les exemples d'attaques réelles sont rares et souvent confidentiels.

### **2.3.2 Apprentissage non supervisé et détection d'anomalies**

Face au manque de données étiquetées, les techniques non supervisées offrent des alternatives prometteuses pour la détection des comportements anormaux :

- Autoencodeurs : Ces réseaux de neurones compressent puis reconstruisent les données d'entrée, permettant d'identifier les comportements qui ne correspondent pas aux patterns appris. Particulièrement adaptés pour établir des profils comportementaux des utilisateurs et détecter les déviations.

- Isolation Forest : Algorithme particulièrement efficace pour isoler rapidement les observations aberrantes en partitionnant récursivement les données. Sa faible complexité computationnelle le rend adapté au traitement en temps réel.

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise) : Cette méthode de clustering identifie les comportements qui ne s'intègrent pas dans les clusters de densité formés par les comportements normaux.

- One-Class SVM : Entraîné uniquement sur des données normales, cet algorithme apprend à délimiter le contour du comportement légitime et peut identifier les observations qui s'en écartent.

### **2.3.3 Approches hybrides et ensemblistes**

Les recherches récentes, notamment celles de Ngalamulume Ilunga (2025), suggèrent que les approches hybrides combinant modèles supervisés et non supervisés offrent les meilleures performances dans des environnements opérationnels :

- Utilisation initiale de techniques non supervisées pour l'identification d'anomalies potentielles

- Raffinement par des modèles supervisés pour réduire les faux positifs - Intégration de feedback humain pour l'amélioration continue des modèles

- Combinaison de classifieurs via des techniques d'ensemble (stacking, voting)

Cette combinaison permet de bénéficier à la fois de la capacité de détection d'anomalies inconnues des modèles non supervisés et de la précision des modèles supervisés pour les patterns déjà identifiés.

## 2.4 Bases NoSQL pour l'analyse des comportements

### 2.4.1 Avantages des bases NoSQL pour la cybersécurité

Les systèmes de gestion de bases de données NoSQL présentent plusieurs avantages significatifs par rapport aux SGBD relationnels traditionnels dans le contexte de la cybersécurité :

- Performances supérieures : Chen et al. (2023) démontrent que Cassandra est 40% plus rapide que MySQL pour l'ingestion de logs de sécurité à volume élevé, un facteur critique pour l'analyse en temps réel.

- Flexibilité du schéma : L'absence de schéma rigide permet d'adapter rapidement le modèle de données à différentes sources d'information et à l'évolution des menaces.

- Scalabilité horizontale : Capacité à s'étendre sur plusieurs nœuds pour gérer la croissance exponentielle des données de sécurité sans dégradation des performances.

- Représentations de données diversifiées : Différents types de bases NoSQL (orientées document, colonne, graphe) s'adaptent naturellement à différents aspects de l'analyse de sécurité.

### 2.4.2 Neo4j et l'analyse de graphes pour la détection

L'utilisation de bases de données orientées graphe comme Neo4j présente des avantages uniques pour la modélisation et la détection des menaces internes. Liu et al. (2022) ont obtenu une précision de 87% sur le dataset CERT en exploitant les capacités suivantes :

## 2.4. BASES NOSQL POUR L'ANALYSE DES COMPORTEMENTS TABLE DE MATIÈRES

- Modélisation naturelle des relations complexes : Les liens entre utilisateurs, systèmes, ressources et actions sont directement représentés dans la structure de graphe.
- Langage de requête Cypher : Permet d'exprimer des patterns complexes d'accès et de comportement qui seraient difficiles à formuler en SQL.
- Algorithmes de théorie des graphes intégrés : Neo4j offre nativement des fonctions d'analyse de centralité, de détection de communautés et de chemin qui révèlent des comportements anormaux.
- Visualisation des réseaux d'interactions : Facilite l'analyse forensique et l'interprétation des alertes par les analystes.

Les métriques dérivées de la théorie des graphes s'avèrent particulièrement pertinentes :

- Centralité de degré : Identifie les utilisateurs accédant à un nombre anormalement élevé de ressources.
- Centralité d'intermédiairité (betweenness) : Détecte les acteurs pivots inhabituels dans la circulation de l'information.
- Détection de communautés : Identifie les utilisateurs s'écartant de leur groupe fonctionnel habituel.

### **2.4.3 Cassandra pour le stockage et l'analyse de logs**

Parallèlement, les bases orientées colonnes comme Cassandra présentent des avantages significatifs pour le stockage et l'analyse des événements de sécurité :

- Ingestion à haut débit : Capacité à absorber des millions d'événements par seconde.
- Modèle orienté colonnes : Particulièrement adapté aux séries temporelles et aux logs d'événements.
- Distribution et réPLICATION : Haute disponibilité sans point unique de défaillance.
- Requêtes temporelles optimisées : Performance élevée pour l'analyse de séquences d'événements sur des plages de temps.

Ngalamulume Ilunga (2025) suggèrent une architecture combinant Cassandra pour le stockage primaire des événements et Neo4j pour l'analyse relationnelle, créant ainsi une solution complète pour la détection des menaces internes.

#### 2.4.4 Vulnérabilités spécifiques aux bases NoSQL

Abuali (2024) met en lumière que l'adoption des bases NoSQL introduit également de nouveaux vecteurs d'attaque qu'il convient de considérer :

- Injections NoSQL : Contrairement aux injections SQL classiques, ces attaques exploitent la structure flexible des requêtes NoSQL.
- Attaques par tautologie : Permettent de contourner les mécanismes d'authentification via des expressions toujours vraies.
- Absence de standardisation sécuritaire : Les pratiques de sécurisation varient considérablement entre les différentes implémentations NoSQL.

L'étude démontre que la validation côté serveur reste la méthode la plus efficace de prévention, et qu'une approche multicouche associant WAF et validations applicatives est recommandée pour minimiser ces risques.

### 2.5 Défis et perspectives

#### 2.5.1 Limitations des approches actuelles

L'analyse de la littérature révèle plusieurs défis persistants dans la détection des menaces internes :

- Qualité et volume des données d'entraînement : Rameshwar et al. (2024) soulignent le manque de datasets représentatifs et complets des menaces internes, limitant l'efficacité des approches supervisées.
- Équilibre faux positifs/faux négatifs : La difficulté majeure consiste à maintenir une sensibilité élevée sans générer un nombre excessif d'alertes qui conduiraient à une "fatigue d'alerte" chez les analystes.
- Interprétabilité des modèles : Oye et al. (2024) identifient le besoin de modèles explicables, particulièrement crucial dans un contexte sécuritaire où les décisions algorithmiques peuvent avoir des conséquences importantes pour les employés.

- Performances computationnelles : L'analyse comportementale en temps réel impose des contraintes significatives sur les ressources, particulièrement dans les grandes organisations avec des milliers d'utilisateurs.
- Protection de la vie privée : Ashraf (2024) souligne la tension entre la surveillance nécessaire pour la détection des menaces et le respect de la vie privée des employés.

## 2.6 Conclusion

En résumé, l'état de l'art met en évidence l'importance croissante des bases NoSQL et des techniques d'apprentissage automatique dans la détection des menaces internes. Les recherches récentes soulignent l'efficacité des modèles supervisés (comme les SVM, Random Forest ou réseaux de neurones) et non supervisés (tels que les autoencodeurs ou l'Isolation Forest) pour identifier des comportements anormaux, malgré les défis liés à la disponibilité des données étiquetées. L'approche hybride, combinant les forces des deux paradigmes, apparaît comme la plus prometteuse pour renforcer la sécurité des systèmes d'information face à des menaces internes de plus en plus sophistiquées et furtives.

---

## **Chapitre III**

# **Méthodologie et Conception**

## Introduction

La conception d'un système de détection des menaces internes efficace nécessite une approche méthodologique rigoureuse qui combine plusieurs dimensions d'analyse. Ce chapitre présente notre méthodologie de conception, en mettant l'accent sur l'architecture technologique adoptée et la modélisation des interactions entre les différentes entités du système d'information. Notre approche exploite les capacités des bases de données NoSQL, particulièrement Neo4j et Cassandra, pour créer un système capable d'analyser en profondeur les comportements des utilisateurs et de détecter les anomalies potentiellement indicatives de menaces internes.

## 3.1 Choix technologiques

### 3.1.1 Justification de Neo4j pour la modélisation des relations

La détection des menaces internes repose fondamentalement sur l'analyse des relations entre les entités du système d'information. Neo4j, en tant que base de données orientée graphe leader, offre plusieurs avantages déterminants pour notre problématique :

1.Modélisation native des relations complexes : Neo4j permet de représenter naturellement les relations multidimensionnelles entre utilisateurs, systèmes, ressources et actions. Cette représentation sous forme de nœuds et d'arêtes facilite l'identification de patterns relationnels complexes souvent invisibles dans les systèmes tabulaires traditionnels.

2.Langage de requête Cypher : Le langage déclaratif Cypher offre une syntaxe expressive pour formuler des requêtes complexes de manière intuitive, facilitant la recherche de patterns comportementaux spécifiques :

```

MATCH (u:User)-[:PERFORMS]->(a:Activity)-[:INVOLVES]->(r:Resource)
RETURN u.name AS User, a.name AS Activity, r.name AS Resource
LIMIT 25

```

Listing 3.1 – Exemple de requête Cypher pour l'extraction des interactions utilisateur-activité-ressource

3.Algorithmes natifs d'analyse de graphes : Neo4j propose une bibliothèque riche d'algorithme (centralité, détection de communautés, chemins) particulièrement pertinents pour l'analyse des comportements et la détection d'anomalies dans les interactions système.

4.Visualisation intégrée : Les capacités de visualisation de Neo4j permettent aux analystes de sécurité d'explorer intuitivement les relations et d'identifier visuellement des patterns suspects, accélérant ainsi le processus d'investigation.

5.Évolutivité : La capacité de Neo4j à gérer des millions de noeuds et de relations tout en maintenant des performances élevées est cruciale pour l'analyse en temps réel des comportements dans les grandes organisations.

### **3.1.2 Justification de Cassandra pour le stockage des logs et événements**

Complémentaire à Neo4j, Apache Cassandra apporte des capacités essentielles pour la gestion des données événementielles :

1.Gestion de volumes massifs : Les systèmes d'information génèrent des quantités considérables de logs et d'événements. L'architecture distribuée de Cassandra permet le stockage et l'interrogation efficace de ces volumes sans compromettre les performances.

2.Modèle orienté colonnes optimisé pour les séries temporelles : Les données de sécurité sont intrinsèquement temporelles. Le modèle de données de Cassandra s'avère particulièrement adapté pour organiser et interroger efficacement ces séries chronologiques.

3.Haute disponibilité et tolérance aux pannes : La nature distribuée de Cassandra garantit une disponibilité continue des données, essentielle pour un système de sécurité critique.

4.Performance en écriture : La capacité à ingérer rapidement un grand volume d'événements est cruciale dans un contexte de surveillance en temps réel. Cassandra excelle particulièrement dans les opérations d'écriture à haut débit.

5.Requêtes temporelles optimisées : Cassandra permet d'exécuter efficacement des requêtes comme "toutes les activités inhabituelles de l'utilisateur X durant le dernier mois" ou "les événements anormaux regroupés par département".

Cette combinaison Neo4j-Cassandra nous offre une architecture hybride puissante alliant

l'analyse relationnelle profonde et le traitement performant de flux d'événements temporels.

## 3.2 Modélisation des interactions internes

### 3.2.1 Construction du graphe utilisateur → actions → systèmes accédés

Construction du graphe utilisateur → actions → systèmes accédés Au cœur de notre approche se trouve un modèle de graphe sophistiqué qui capture les diverses dimensions des interactions au sein du système d'information. Cette modélisation a été implémentée dans notre projet à travers l'importation du fichier CSV dans Neo4j, établissant ainsi la structure de base de notre graphe d'analyse. Structure fondamentale du graphe Notre modèle de graphe dans Neo4j comprend plusieurs types de nœuds interconnectés :

#### 1.Nœuds principaux :

- **User** : Représente un utilisateur du système (employé, contractant, etc.).
- **System** : Systèmes informatiques auxquels les utilisateurs accèdent.
- **Activity** : Actions réalisées par les utilisateurs.
- **Resource** : Ressources informationnelles accédées (fichiers, bases de données, etc.).

#### 2.Relations principales :

- (:User)-[:PERFORMS]->(:Activity) : Capture l'action réalisée par un utilisateur.
- (:User)-[:USES]->(:System) : Lie un utilisateur au système qu'il utilise.
- (:System)-[:ACCESSES]->(:Resource) : Représente l'accès d'un système à une ressource.
- (:Activity)-[:INVOLVES]->(:Resource) : Lie une activité à la ressource concernée.

Cette structure permet de modéliser finement non seulement les actions directes mais aussi leur contexte multidimensionnel, essentiel pour distinguer les comportements légitimes des anomalies.

### 3.2.2 Exploration et validation du modèle

Pour explorer et valider notre modèle de graphe, nous avons implémenté plusieurs requêtes Cypher qui permettent d'analyser différents aspects des interactions internes :

#### 1. Analyse des Données via Requêtes Cypher dans Neo4j

Cette section présente les principales requêtes Cypher utilisées pour explorer et analyser les données importées dans Neo4j. Chaque requête est accompagnée d'une explication de son objectif.

**1. Visualisation des nœuds** Permet d'afficher les premiers nœuds du graphe pour une vue d'ensemble rapide.

```
MATCH (n) RETURN n LIMIT 50
```

Listing 3.2 – Voir les nœuds du graphe

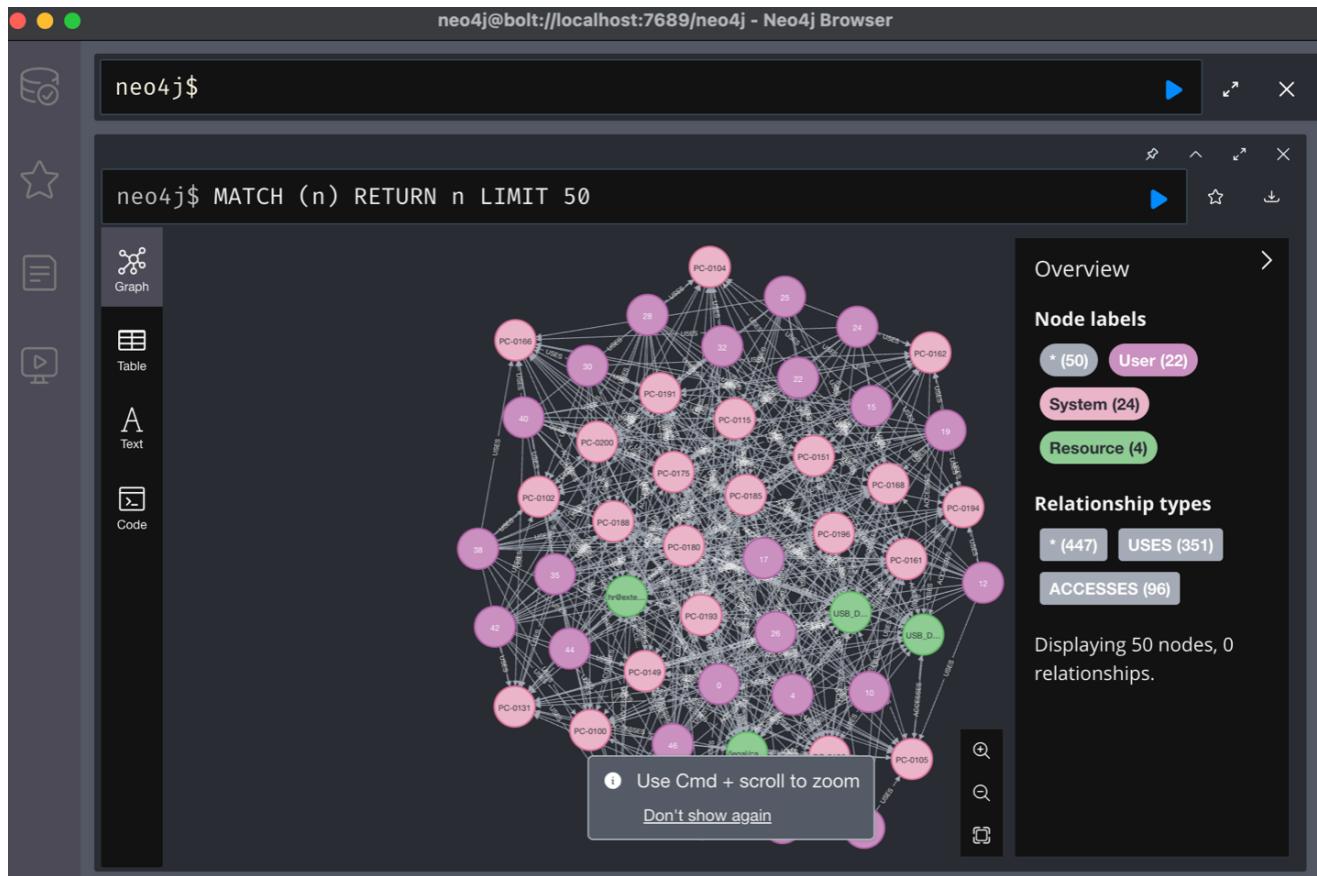


FIGURE 3.1 – les noeuds du graphe

```
MATCH (n) RETURN n LIMIT 5
```

Listing 3.3 – Voir les noeuds du graphe

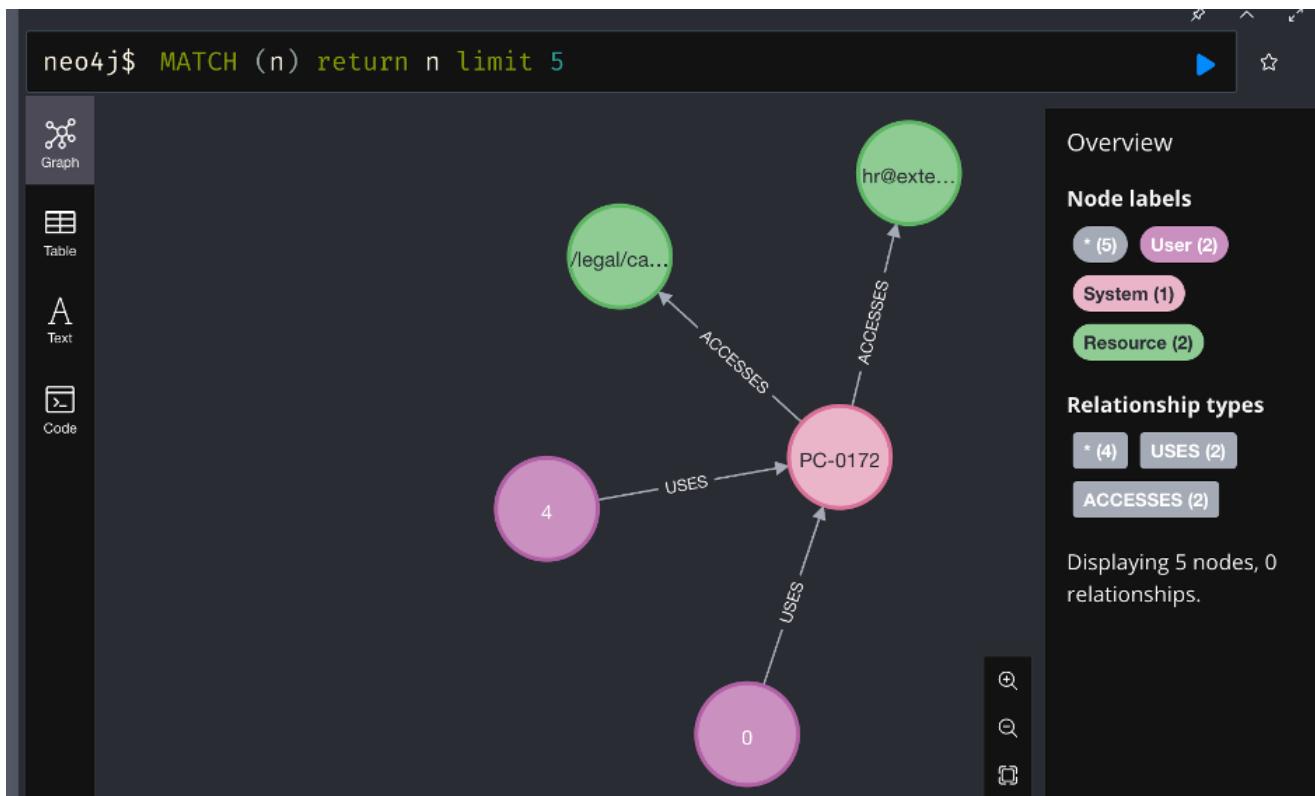


FIGURE 3.2 – exemple de cinq noeuds du graphe

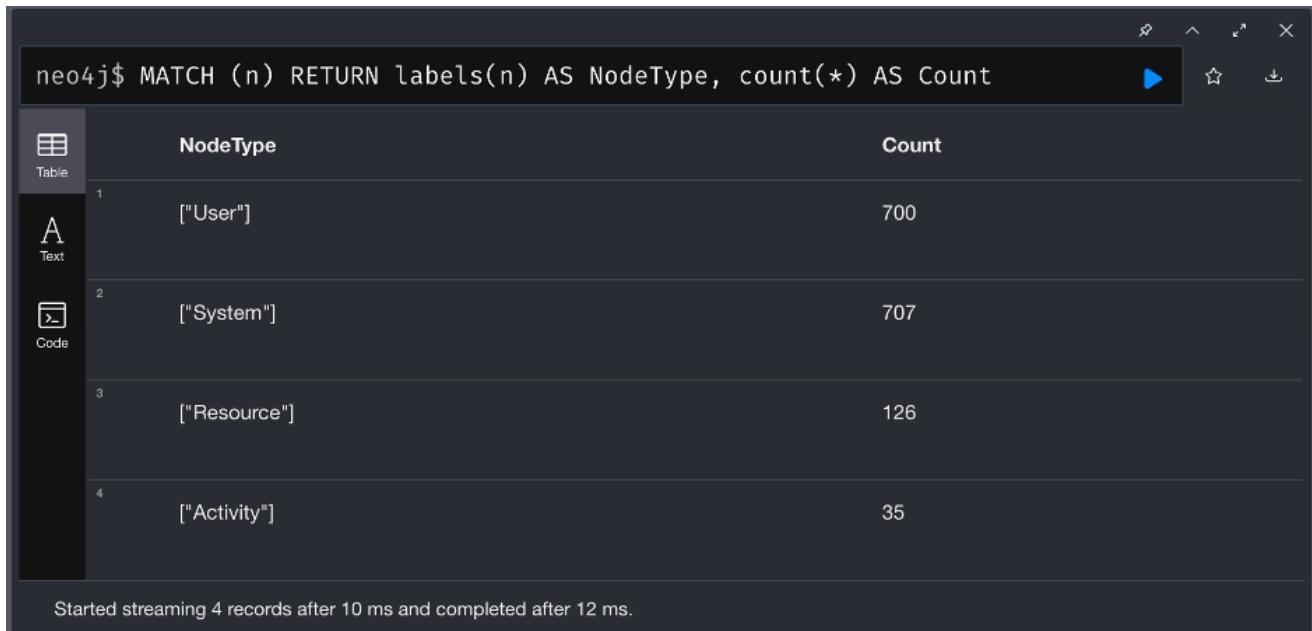
## 2. Comptage des noeuds par type

Utilisée pour vérifier le nombre de noeuds créés par label.

Cette requête compte le nombre de noeuds pour chaque type (label) dans le graphe. Cela permet de vérifier que les types de noeuds (User, System, Activity, Resource) sont bien créés.

```
MATCH (n) RETURN labels(n) AS NodeType, COUNT(*) AS Count
```

Listing 3.4 – Compter les noeuds par label



The screenshot shows the Neo4j browser interface. At the top, a command-line input field contains the query: `neo4j$ MATCH (n) RETURN labels(n) AS NodeType, count(*) AS Count`. Below the input field is a table with two columns: **NodeType** and **Count**. The table displays four rows of data:

	NodeType	Count
1	["User"]	700
2	["System"]	707
3	["Resource"]	126
4	["Activity"]	35

At the bottom of the table, a message states: `Started streaming 4 records after 10 ms and completed after 12 ms.`

FIGURE 3.3 – Compter les nœuds par label

### 3. Relations Utilisateur - Activité

Montre les actions effectuées par chaque utilisateur.

Cette requête permet de visualiser les relations PERFORMS entre les utilisateurs et les activités, avec le timestamp de chaque action.

```
MATCH (u:User)-[r:PERFORMS]->(a:Activity)
RETURN u.name AS User, a.name AS Activity, r.timestamp AS Timestamp
LIMIT 25
```

Listing 3.5 – Relations PERFORMS entre utilisateurs et activités

	User	Activity	Timestamp
1	"USR0089"	"logon"	"2025-03-29T01:02:00"
16	"USR0029"	"logon"	"2025-03-28T12:05:00"
17	"USR0036"	"logon"	"2025-03-28T23:27:00"
18	"USR0049"	"logon"	"2025-03-28T20:23:00"
19	"USR0025"	"logon"	"2025-03-28T18:03:00"
20	"USR0072"	"logon"	"2025-03-28T18:25:00"
21	"USR0068"	"logon"	"2025-03-28T16:44:00"

Started streaming 100 records after 4 ms and completed after 5 ms.

FIGURE 3.4 – Relations PERFORMS entre utilisateurs et activités

#### 4. Relations Système - Ressource

Indique quelles ressources sont accédées par quels systèmes.

Cette requête permet de visualiser les relations ACCESSES entre les systèmes et les ressources, avec le timestamp correspondant

```

MATCH (s:System)-[r:ACCESSES]->(res:Resource)
RETURN s.name AS System, res.name AS Resource, r.timestamp AS Timestamp
LIMIT 25
    
```

Listing 3.6 – Relations ACCESSES entre systèmes et ressources

	System	Resource	Timestamp
1	"PC-0172"	"hr@externalmail.com"	"2025-03-28T22:48:00"
2	"PC-0161"	"hr@externalmail.com"	"2025-03-29T07:48:00"
3	"PC-0106"	"hr@externalmail.com"	"2025-03-29T05:16:00"
4	"PC-0105"	"hr@externalmail.com"	"2025-03-28T22:44:00"
5	"PC-0168"	"hr@externalmail.com"	"2025-03-29T00:21:00"
6	"PC-0196"	"hr@externalmail.com"	"2025-03-29T01:15:00"
7			

Started streaming 25 records after 1 ms and completed after 1 ms.

FIGURE 3.5 – Relations ACCESSES entre systèmes et ressources

## 5. Relations Utilisateur - Système

Permet de savoir quel utilisateur utilise quel système.

Cette requête montre les relations USES entre les utilisateurs et les systèmes, avec le timestamp.

```

MATCH (u:User)-[r:USES]->(s:System)
RETURN u.name AS User, s.name AS System, r.timestamp AS Timestamp
LIMIT 25
    
```

Listing 3.7 – Relations USES entre utilisateurs et systèmes

The screenshot shows the Neo4j browser interface with a query results table. The table has three columns: User, System, and Timestamp. The data consists of seven rows, each numbered from 1 to 7. Row 1: "USR0089", "PC-0172", "2025-03-28T08:52:00". Row 2: "USR0089", "PC-0161", "2025-03-28T17:18:00". Row 3: "USR0089", "PC-0106", "2025-03-29T01:49:00". Row 4: "USR0089", "PC-0105", "2025-03-28T13:13:00". Row 5: "USR0089", "PC-0196", "2025-03-28T14:09:00". Row 6: "USR0089", "PC-0194", "2025-03-29T05:46:00". Row 7: (empty). A message at the bottom says "Started streaming 25 records after 1 ms and completed after 1 ms."

	User	System	Timestamp
1	"USR0089"	"PC-0172"	"2025-03-28T08:52:00"
2	"USR0089"	"PC-0161"	"2025-03-28T17:18:00"
3	"USR0089"	"PC-0106"	"2025-03-29T01:49:00"
4	"USR0089"	"PC-0105"	"2025-03-28T13:13:00"
5	"USR0089"	"PC-0196"	"2025-03-28T14:09:00"
6	"USR0089"	"PC-0194"	"2025-03-29T05:46:00"
7			

Started streaming 25 records after 1 ms and completed after 1 ms.

FIGURE 3.6 – Relations USES entre utilisateurs et systèmes

## 6. Relations Activité - Ressource

Lie chaque activité à une ressource utilisée.

Cette requête permet de visualiser les relations INVOLVES entre les activités et les ressources, avec leur timestamp.

```

MATCH (a:Activity)-[r:INVOLVES]->(res:Resource)
RETURN a.name AS Activity, res.name AS Resource, r.timestamp AS
Timestamp
LIMIT 25
    
```

Listing 3.8 – Relations INVOLVES entre activités et ressources

	Activity	Resource	Timestamp
1	"logon"	"_"	"2025-03-28T14:39:00"
2	"file"	"/shared/confidential.pdf"	"2025-03-28T10:52:00"
3	"file"	"/hr/contract.pdf"	"2025-03-28T14:09:00"
4	"file"	"/home/user/data.csv"	"2025-03-29T06:12:00"
5	"file"	"/work/report.docx"	"2025-03-28T13:36:00"
6	"file"	"/legal/case.txt"	"2025-03-29T02:42:00"
7			

Started streaming 25 records after 1 ms and completed after 1 ms.

FIGURE 3.7 – Relations INVOLVES entre activités et ressources

## 7. Chemin Utilisateur → Activité → Ressource

Permet de retracer une interaction complète.

Cette requête permet de visualiser le chemin complet entre un utilisateur, une activité et une ressource, et de comprendre les relations entre ces nœuds

```

MATCH (u:User)-[:PERFORMS]->(a:Activity)-[:INVOLVES]->(r:Resource)
RETURN u.name AS User, a.name AS Activity, r.name AS Resource
LIMIT 25
    
```

Listing 3.9 – Chemin complet des interactions

	User	Activity	Resource
1	"USR0060"	"logon"	"_"
2	"USR0094"	"logon"	"_"
3	"USR0018"	"logon"	"_"
4	"USR0074"	"logon"	"_"
5	"USR0097"	"logon"	"_"
6	"USR0100"	"logon"	"_"
7			

Started streaming 25 records in less than 1 ms and completed after 1 ms.

FIGURE 3.8 – Chemin complet des interactions

## 8. Utilisateurs les plus actifs

Affiche les utilisateurs qui réalisent le plus d'activités.

Cette requête permet de trouver les 10 utilisateurs ayant réalisé le plus grand nombre d'activités.

```

MATCH (u:User)-[:PERFORMS]->(a:Activity)
RETURN u.name AS User, COUNT(a) AS ActivityCount
ORDER BY ActivityCount DESC
LIMIT 10
    
```

Listing 3.10 – Top utilisateurs par activité

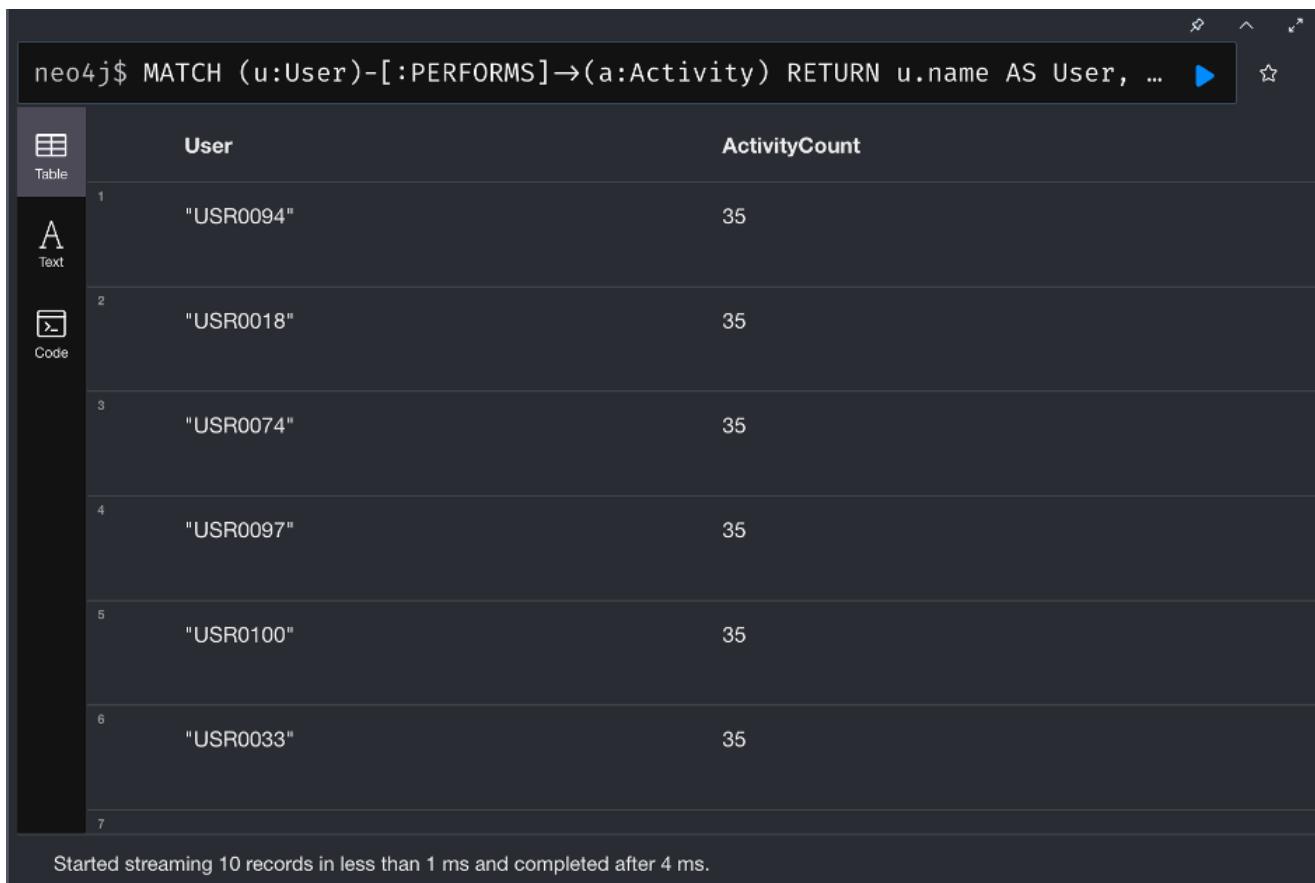


FIGURE 3.9 – Chemin complet des interactions

## 9. Ressources les plus accédées

Classe les ressources en fonction du nombre d'accès.

Cette requête montre les 10 ressources les plus accédées par les systèmes, classées par nombre d'accès.

```

MATCH (s:System)-[:ACCESSES]->(res:Resource)
RETURN res.name AS Resource, COUNT(*) AS AccessCount
ORDER BY AccessCount DESC
LIMIT 10
    
```

Listing 3.11 – Top ressources accédées

	Resource	AccessCount
1	"_"	706
2	"EXT_HDD_X"	666
3	"USB_DRIVE_B"	650
4	"USB_DRIVE_A"	648
5	"hr@externalmail.com"	636
6	"it@company.com"	630
7		

Started streaming 10 records after 1 ms and completed after 6 ms.

FIGURE 3.10 – Top ressources accédées

## Détection de comportements suspects

Nous utilisons des requêtes Cypher spécifiques pour identifier des activités potentiellement malveillantes ou inhabituelles.

### 1. Détection de comportements anormaux

Pour trouver les utilisateurs qui effectuent des activités qu'ils n'ont jamais faites auparavant.

```

MATCH (u:User)-[r:PERFORMS]->(a:Activity)
WITH u, a, count(r) AS freq, min(r.timestamp) AS first_time
WHERE freq = 1
RETURN u.name AS User, a.name AS Activity, first_time AS FirstOccurrence
ORDER BY first_time DESC
    
```

Listing 3.12 – Activités exécutées une seule fois par un utilisateur

	User	Activity	FirstOccurrence
1	"USR0042"	"logon"	"2025-03-29T07:59:00"
2	"USR0036"	"http"	"2025-03-29T07:57:00"
3	"USR0036"	"http"	"2025-03-29T07:57:00"
4	"USR0036"	"http"	"2025-03-29T07:57:00"
5	"USR0036"	"http"	"2025-03-29T07:57:00"
6	"USR0036"	"http"	"2025-03-29T07:57:00"
7			

Started streaming 3486 records in less than 1 ms and completed after 13 ms, displaying first 1000 rows.

FIGURE 3.11 – Activités exécutées une seule fois par un utilisateur

## 2. Accès à des ressources sensibles hors horaires

Déetecte les accès à des données sensibles en dehors des horaires de bureau (9h-18h).

```

MATCH (u:User)-[:PERFORMS]->(a:Activity)-[:INVOLVES]->(r:Resource)
WHERE
    r.name CONTAINS "confidential" OR r.name CONTAINS "sensitive"
    AND (datetime(r.timestamp).hour < 9 OR datetime(r.timestamp).hour >
        18)
RETURN u.name AS User, r.name AS SensitiveResource, a.name AS Activity,
       r.timestamp AS Timestamp
  
```

Listing 3.13 – Accès suspects hors horaires

User ID	Resource Path	File Type	Value
"USR0080"	"/shared/confidential.pdf"	"file"	null
"USR0045"	"/shared/confidential.pdf"	"file"	null
"USR0030"	"/shared/confidential.pdf"	"file"	null
"USR0081"	"/shared/confidential.pdf"	"file"	null
"USR0010"	"/shared/confidential.pdf"	"file"	null
"USR0019"	"/shared/confidential.pdf"	"file"	null
"USR0046"	"/shared/confidential.pdf"	"file"	null
"USR0048"	"/shared/confidential.pdf"	"file"	null
"USR0009"	"/shared/confidential.pdf"	"file"	null
"USR0039"	"/shared/confidential.pdf"	"file"	null
"USR0052"	"/shared/confidential.pdf"	"file"	null
"USR0060"	"/shared/confidential.pdf"	"file"	null

FIGURE 3.12 – Accès suspects hors horaires

### 3. Accès inhabituels à certaines ressources

Met en évidence des comportements rares ou exceptionnels.

Cette requête permet de trouver les systèmes qui accèdent à des ressources qu'ils n'accèdent pas habituellement

```

MATCH (s:System)-[r:ACCESSES]->(res:Resource)
WITH s, res, COUNT(r) AS access_count
WHERE access_count < 3
RETURN s.name AS System, res.name AS Resource, access_count AS
AccessCount
ORDER BY access_count
    
```

Listing 3.14 – Accès rares à des ressources

	System	Resource	AccessCount
1	"PC-0172"	"hr@externalmail.com"	1
2	"PC-0161"	"hr@externalmail.com"	1
3	"PC-0106"	"hr@externalmail.com"	1
4	"PC-0105"	"hr@externalmail.com"	1
5	"PC-0168"	"hr@externalmail.com"	1
6	"PC-0196"	"hr@externalmail.com"	1
7			

Started streaming 11282 records in less than 1 ms and completed after 18 ms, displaying first 1000 rows.

FIGURE 3.13 – Accès rares à des ressources

#### 4. Mouvement latéral

Identifie des utilisateurs qui changent rapidement de système, ce qui peut suggérer une tentative de compromission.

Cette requête permet de trouver les utilisateurs qui utilisent plusieurs systèmes différents dans un court laps de temps

```

MATCH (u:User)-[r1:USES]->(s1:System)
MATCH (u)-[r2:USES]->(s2:System)
WHERE s1 <> s2
AND datetime(r1.timestamp) < datetime(r2.timestamp) < datetime(r1.
    timestamp) + duration('PT1H')
RETURN u.name AS User, s1.name AS FirstSystem, s2.name AS SecondSystem,
    r1.timestamp AS FirstAccess, r2.timestamp AS SecondAccess
  
```

Listing 3.15 – Détection de mouvement latéral

The screenshot shows the Neo4j browser interface. At the top, there is a code editor window titled "neo4j\$". The code is a Cypher query:

```

1 // Trouver les utilisateurs qui utilisent plusieurs systèmes
2 différents dans un court laps de temps
3 MATCH (u:User)-[r1:USES]→(s1:System)
4 MATCH (u)-[r2:USES]→(s2:System)
5 WHERE s1 ≠ s2
6 AND datetime(r1.timestamp) < datetime(r2.timestamp) <
    datetime(r1.timestamp) + duration('PT1H') // Dans une fenêtre d'une
    heure
7 RETURN u.name AS User, s1.name AS FirstSystem, s2.name AS
    SecondSystem,
8           r1.timestamp AS FirstAccess, r2.timestamp AS SecondAccess

```

Below the code editor is a table view. On the left, there are three tabs: "Table" (selected), "Text", and "Code". The table has six columns: "User", "FirstSystem", "SecondSystem", "FirstAccess", "SecondAccess", and an empty column at the top. The data is as follows:

	User	FirstSystem	SecondSystem	FirstAccess	SecondAccess
1	"USR0089"	"PC-0111"	"PC-0172"	"2025-03-28T08:22:00"	"2025-03-28T08:52:00"
2	"USR0089"	"PC-0109"	"PC-0172"	"2025-03-28T08:42:00"	"2025-03-28T08:52:00"
3	"USR0089"	"PC-0118"	"PC-0172"	"2025-03-28T08:30:00"	"2025-03-28T08:52:00"
4	"USR0089"	"PC-0167"	"PC-0161"	"2025-03-28T16:37:00"	"2025-03-28T17:18:00"
5	"USR0089"	"PC-0162"	"PC-0106"	"2025-03-29T00:55:00"	"2025-03-29T01:49:00"
6	"USR0089"	"PC-0117"	"PC-0106"	"2025-03-29T01:42:00"	"2025-03-29T01:49:00"

FIGURE 3.14 – Détection de mouvement latéral

La détection des anomalies dans les graphes peut être améliorée en utilisant des métriques de centralité et d'entropie. Ces deux concepts aident à identifier des comportements ou des structures inhabituels dans le graphe, ce qui est particulièrement utile pour la détection des menaces internes ou des actions anormales.

### 1. Métriques de centralité dans un graphe

Les métriques de centralité mesurent l'importance ou l'influence d'un nœud (ou d'un utilisateur dans ce cas) dans un réseau. Des anomalies peuvent se manifester lorsque certains nœuds

présentent des valeurs de centralité très élevées ou très faibles par rapport à la normale, ce qui peut indiquer des comportements suspects. Centralité de degré (Degree Centrality)

La centralité de degré mesure le nombre de connexions (arêtes) directes d'un noeud. Plus un noeud a de connexions, plus il est central dans le graphe.

Détection d'anomalie : Un utilisateur qui a un nombre élevé de connexions (accès ou actions) à différentes ressources peut être un indicateur de comportement anormal, surtout s'il n'a pas l'habitude d'avoir de telles connexions.

Si un utilisateur accède soudainement à un nombre anormalement élevé de ressources, cela pourrait être un signe d'activité malveillante ou d'accès non autorisé.

```
MATCH (u:User)-[:PERFORMS]->(a:Activity)
RETURN u.name, COUNT(a) AS degree
ORDER BY degree DESC
LIMIT 5;
```

Listing 3.16 – Top 5 des utilisateurs les plus actifs



FIGURE 3.15 – Top 5 des utilisateurs les plus actifs

## 2. Entropie des activités utilisateurs

L’entropie mesure le niveau de désordre ou d’incertitude dans un système. Dans le contexte des graphes, l’entropie peut être utilisée pour mesurer l’imprévisibilité des interactions d’un utilisateur. Une forte entropie signifie qu’un utilisateur interagit avec une large variété de ressources, tandis qu’une faible entropie peut indiquer un comportement plus ciblé et prévisible.

```

MATCH (u:User)-[:PERFORMS]->(a:Activity)
WITH u, a.name AS activity, COUNT(a) AS frequency
WITH u, COLLECT(frequency) AS frequencies
WITH u, frequencies, REDUCE(s = 0.0, f IN frequencies | s + f * LOG(f))
    AS entropy
RETURN u.name AS User, -entropy AS Entropy
  
```

```

ORDER BY Entropy DESC
LIMIT 5

```

Listing 3.17 – Calcul de l'entropie des activités

```

1 MATCH (u:User)-[:PERFORMS]-(a:Activity)
2 WITH u, a.name AS activity, COUNT(a) AS frequency
3 WITH u, COLLECT(frequency) AS frequencies
4 WITH u, frequencies, REDUCE(s = 0.0, f IN frequencies | s + f * LOG(f)) AS entropy
5 RETURN u.name AS user, -entropy AS entropy
6 ORDER BY entropy DESC
7 LIMIT 50;

```

	user	entropy
1	"USR0094"	0.0
2	"USR0018"	0.0
3	"USR0074"	0.0
4	"USR0097"	0.0
5	"USR0100"	0.0
6	"USR0033"	0.0

FIGURE 3.16 – Calcul de l'entropie des activités

Ces métriques avancées nous permettent de détecter des comportements subtils qui pourraient échapper aux méthodes d'analyse traditionnelles.

### 3.3 Conclusion

Ce chapitre a détaillé la méthodologie employée pour construire une modélisation des comportements utilisateurs à l'aide de Neo4j. Grâce à la structure en graphe et à l'analyse de métriques telles que la centralité et l'entropie, il est possible de mettre en évidence des comportements anormaux pouvant révéler une menace interne.

## **Chapitre IV**

# **Implémentation et Expérimentations**

## 4.1 Introduction

La détection des menaces internes représente un défi majeur pour les organisations en raison de la nature même de ces menaces : elles proviennent d'utilisateurs légitimes disposant déjà d'accès au système. Face à cette problématique, nous avons développé un système de détection basé sur une approche hybride combinant l'analyse comportementale via des bases NoSQL et différentes techniques de machine learning. Ce chapitre présente en détail l'implémentation du système et les expérimentations réalisées pour évaluer son efficacité.

Notre approche s'articule autour de deux axes principaux : la modélisation des comportements utilisateurs à travers l'analyse de leurs interactions avec le système, et l'application de techniques de machine learning pour identifier les anomalies caractéristiques d'une menace interne. Les expérimentations permettent de comparer l'efficacité de différents modèles, tant supervisés que non supervisés, dans la détection de comportements suspects

## 4.2 Développement du système de détection

### 4.2.1 Collecte et structuration des données

Le système repose sur l'analyse de journaux d'activité utilisateurs (logs) collectés dans un environnement informatique simulé. Ces logs incluent des événements tels que :

- Connexions (logon/logoff), - Accès à des fichiers, - Activité USB, - Accès web (HTTP), - Échanges d'emails, - Requêtes vers des systèmes internes.

Chaque événement est associé à un utilisateur, une ressource ou un système, un type d'action et un horodatage précis. Ces données ont été organisées en graphes à l'aide de Neo4j, où chaque utilisateur est un noeud, relié à des ressources, systèmes, ou événements via des relations typées (e.g., ACCESSE, PERFORMED, SENT).

Ce modèle permet de capturer les interactions internes de manière granulaire et temporelle, rendant les analyses comportementales et structurelles (graphiques) possibles.

## 4.2.2 Exploration des Données et Comportements Utilisateurs

### Activités suspectes après les heures normales

L’analyse révèle que plusieurs utilisateurs présentent des activités récurrentes après les horaires de travail, un indicateur potentiel de comportement anormal. Par exemple :

USR0033, USR0038, USR0021 effectuent 5 activités après heures, ce qui les distingue nettement des autres. Cette catégorie est visualisée dans le graphique “Activités après heures” du tableau de bord utilisateur.

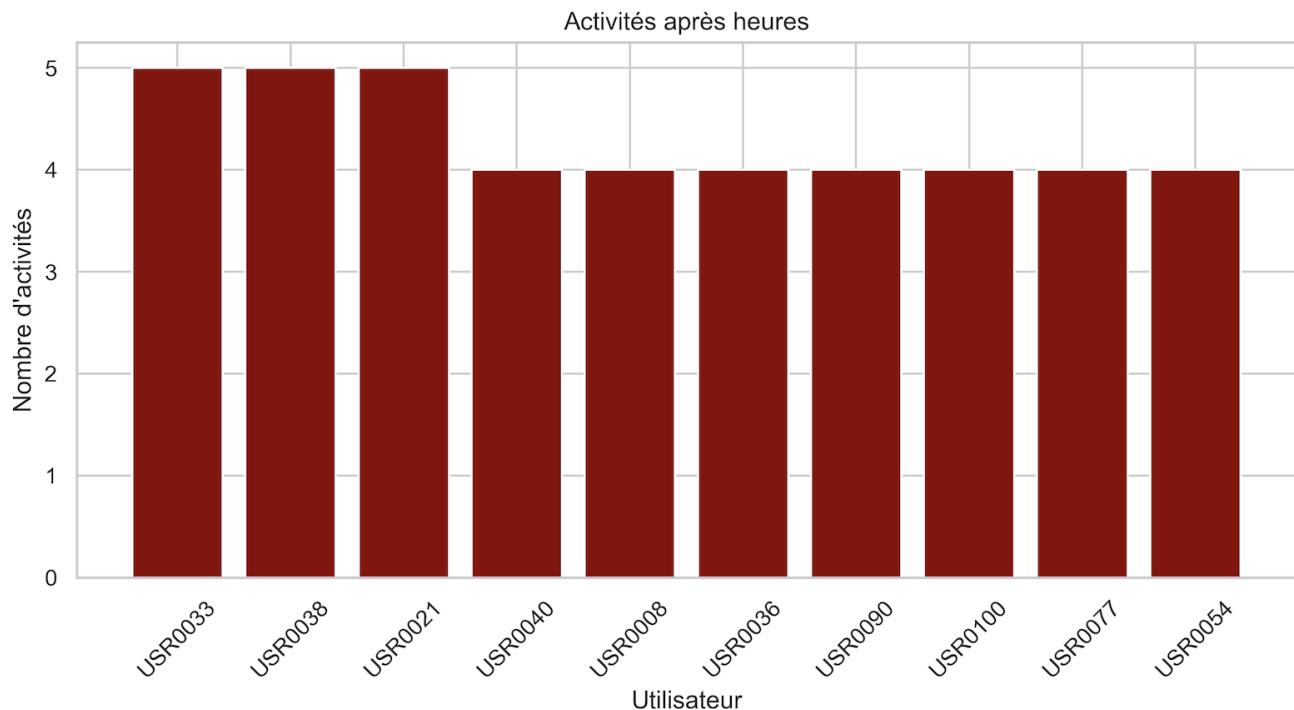


FIGURE 4.17 – Activités après heures

### Accès à des ressources inhabituelles

Chaque utilisateur suspect accède exactement à 18 ressources uniques, comprenant : Fichiers confidentiels (/shared/confidential.pdf), Sites externes comme linkedin.com ou jobboard.com, Périphériques USB (USB\_DRIVE\_A, B), Emails externes (contact@jobmail.com).

Cela montre un comportement homogène mais anormal, avec une diversité dans les types de ressources, pointant vers une possible tentative d’exfiltration de données.

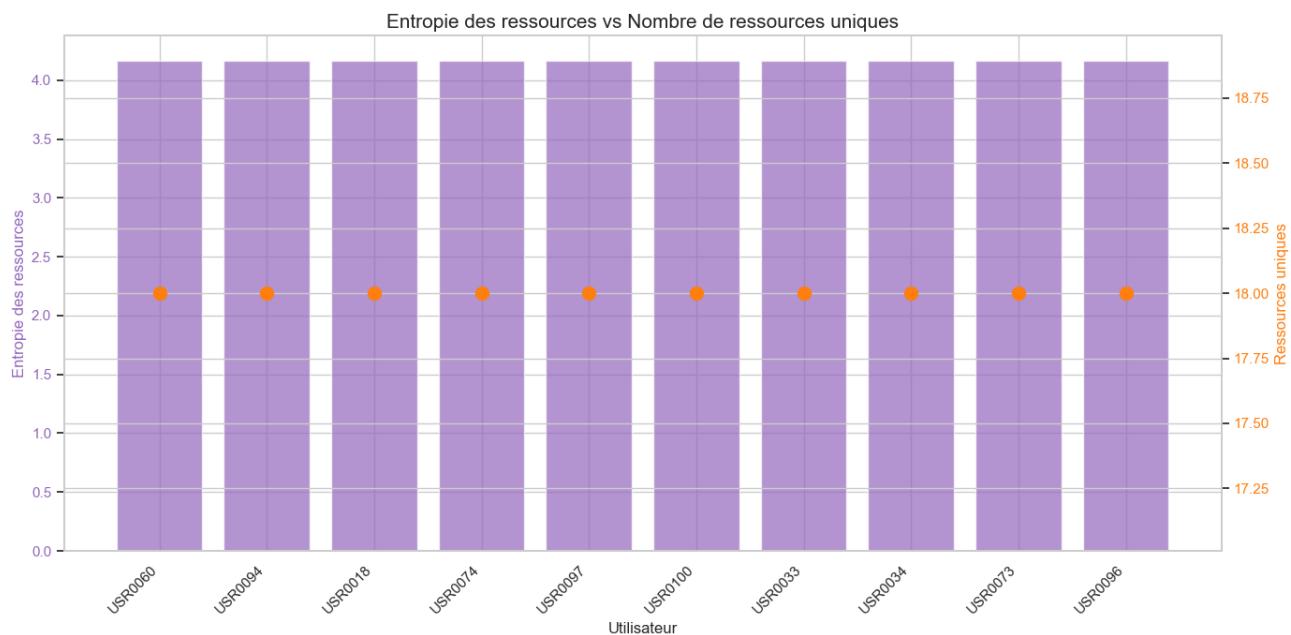


FIGURE 4.18 – Entropie des ressources vs Nombre de ressources uniques

Tous les utilisateurs accèdent au même nombre de ressources (18), avec une entropie constante ( 4.2), ce qui pourrait indiquer un script automatisé ou un comportement standardisé masquant une activité malveillante.

### Distribution des caractéristiques

Le boxplot montre que :

- unique\_systems (systèmes uniques accédés) est très élevé pour certains utilisateurs ( 64 en médiane).
- Les ratios d'activités hors heures/weekend sont faibles mais quelques valeurs aberrantes existent.
- L'entropie des activités est relativement uniforme ( 2.32).

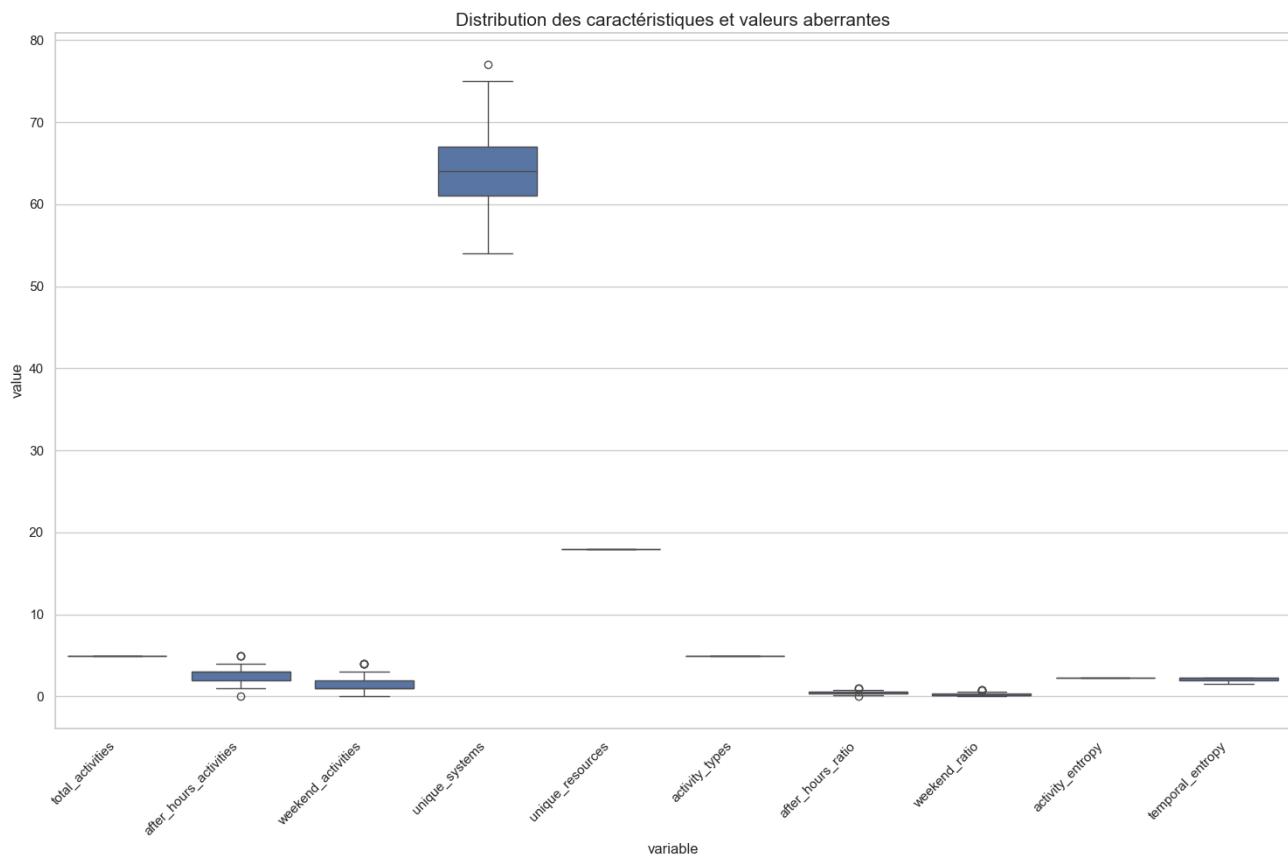


FIGURE 4.19 – Distribution des caractéristiques

Cela suggère :

Une hétérogénéité structurelle (certains utilisateurs accèdent à beaucoup plus de systèmes),  
Une homogénéité comportementale (mêmes types d'activités, mêmes entropies), facilitant l'identification des déviants.

### Corrélations entre comportements

- **Forte corrélation (1.0)** entre `total_activities` et `after_hours_ratio`.
- **Corrélations significatives** (environ 0.64) entre :
  - `after_hours_activities` et `weekend_activities`,
  - `after_hours_ratio` et `weekend_ratio`.

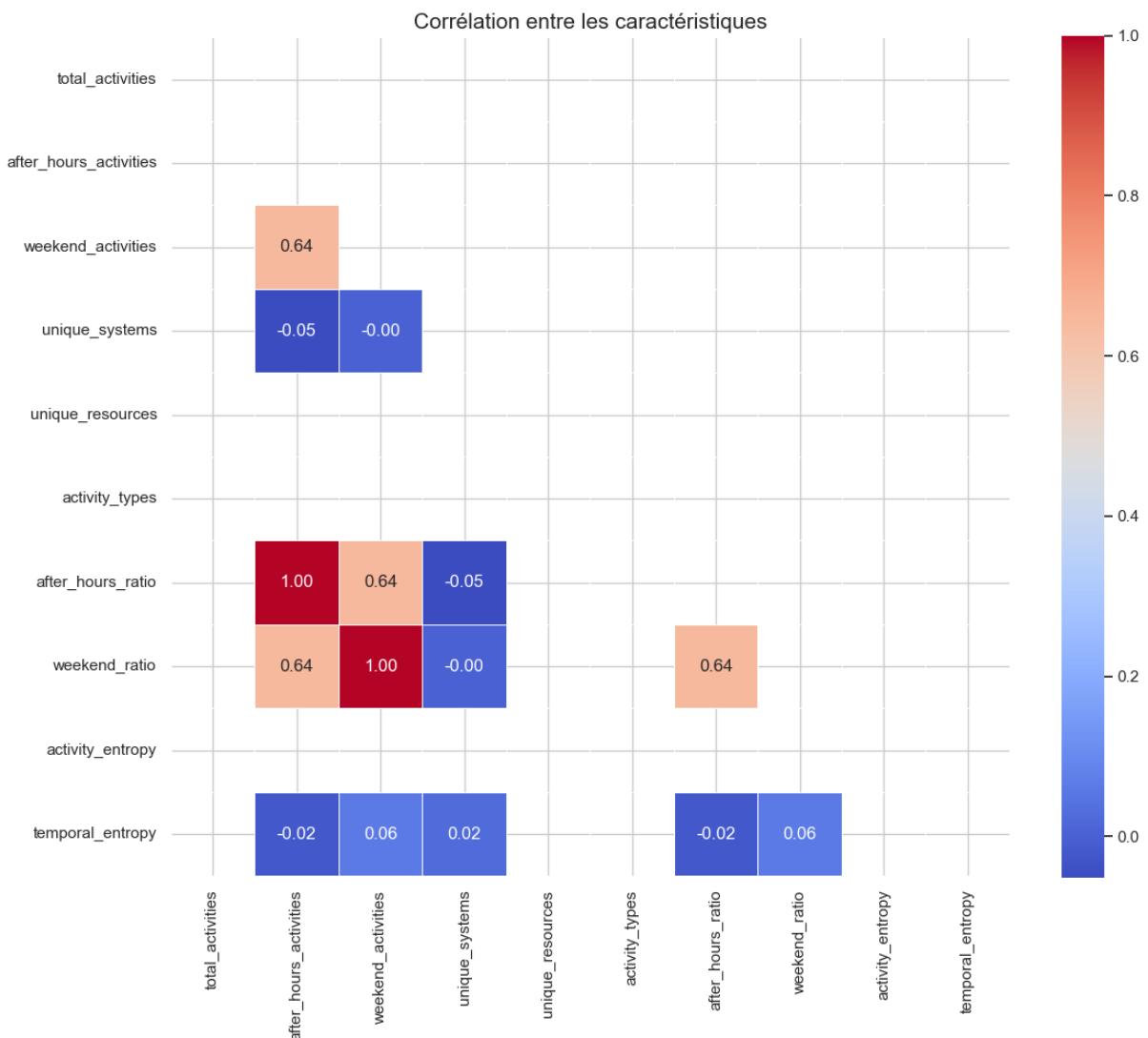


FIGURE 4.20 – Matrice de corrélation comportementale

Cela signifie que les utilisateurs les plus actifs sont aussi ceux qui travaillent souvent en dehors des heures normales, indiquant une potentielle catégorie de risque.

### Analyse de l'entropie et des ressources

Seul un point visible (5 activités, entropie = 2.32, ratio after hours = 0.6) indique un manque de variance, mais aussi une forte uniformité comportementale.

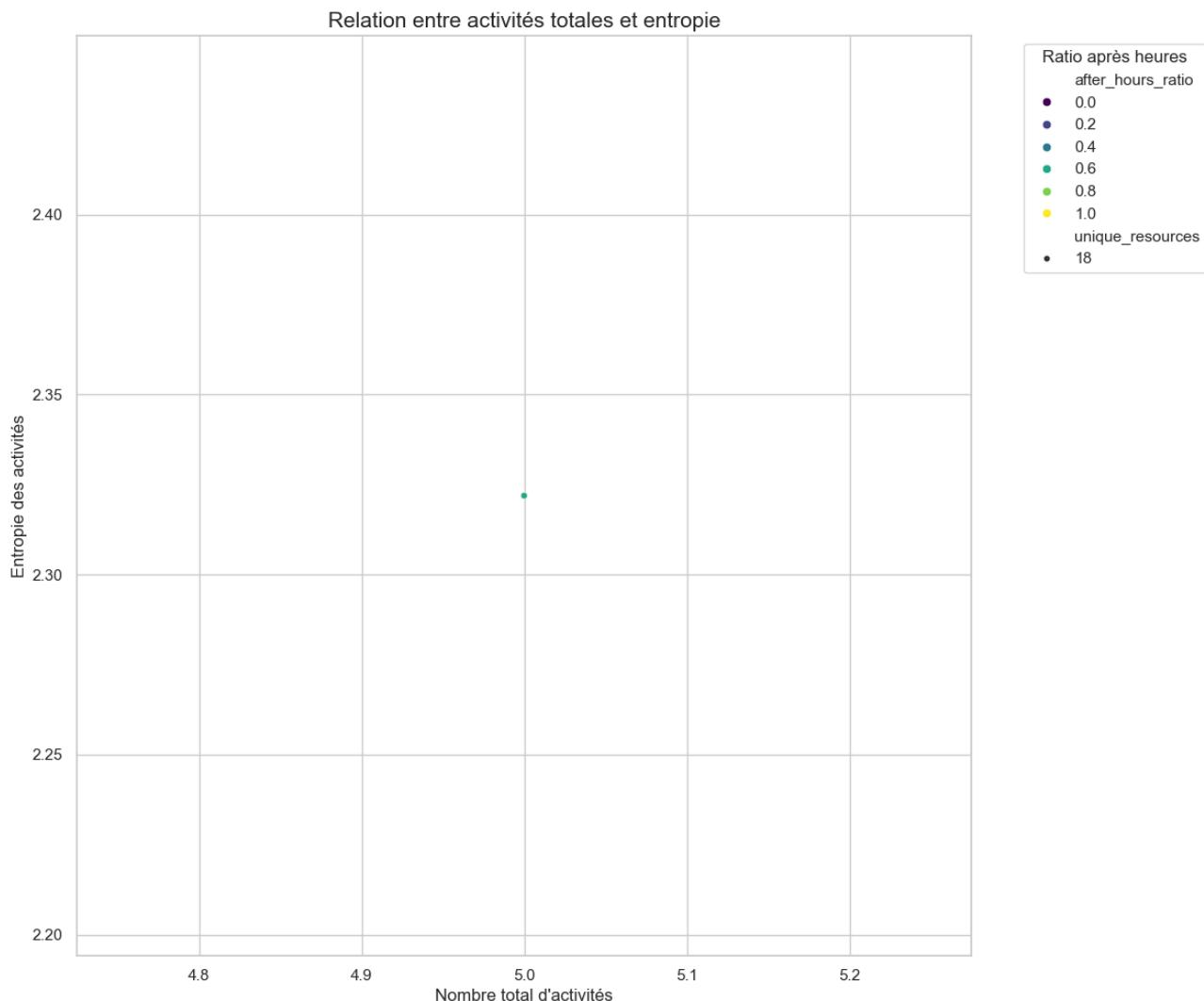


FIGURE 4.21 – Entropie vs Nombre d'activités

### Centralité dans le réseau

USR0060 et USR0031 sont les plus centraux (centralité 80), Ces utilisateurs sont potentiellement des hubs d'interaction, susceptibles de jouer un rôle de coordinateurs dans un comportement malveillant.

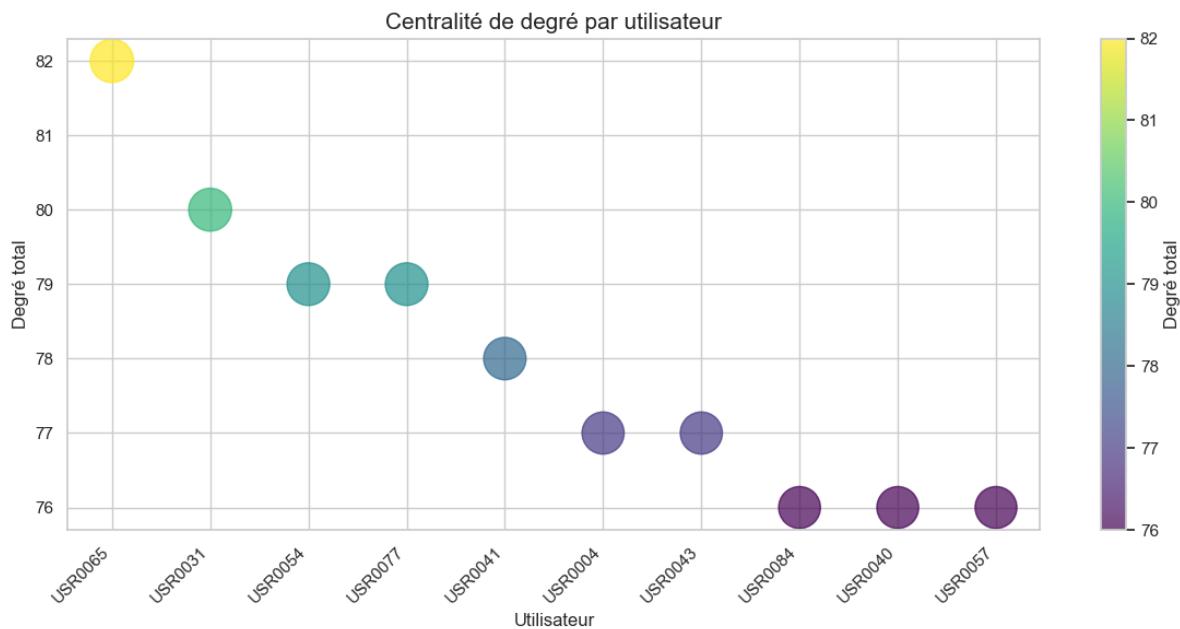


FIGURE 4.22 – Centralité de degré par utilisateur

Cette métrique permet d’identifier les utilisateurs les plus connectés, susceptibles de faciliter ou dissimuler des actions anormales.

### Synthèse Visuelle

Ce tableau de bord regroupe :

Les utilisateurs à haute activité, L’entropie des activités et des ressources, Les corrélations temporelles, La centralité.

Tableau de bord d'analyse des comportements utilisateurs

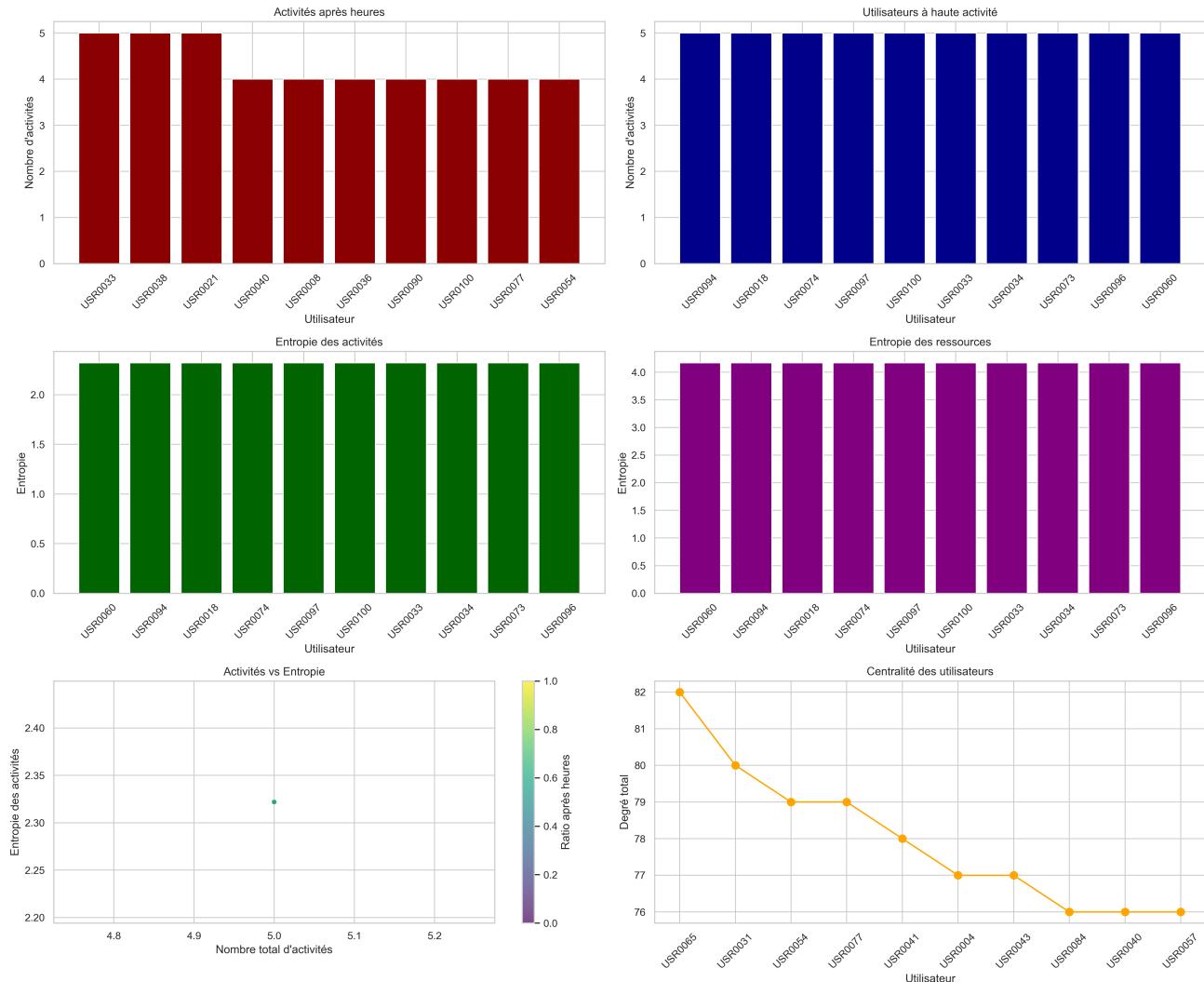


FIGURE 4.23 – Tableau de bord d'analyse des comportements utilisateurs

L'ensemble suggère une cohérence comportementale, sauf pour quelques utilisateurs centralisés, actifs après heures, et accédant à des ressources sensibles. Ces indices cumulés en font des candidats probables à une menace interne.

## Conclusion

L'analyse exploratoire des données montre une homogénéité comportementale apparente (mêmes nombres d'activités, ressources, entropie) qui cache potentiellement des comportements anormaux détectés à travers :

#### 4.3. ALGORITHMES DE DÉTECTION BASÉS SUR NEO4J ET ML TABLE DE MATIÈRES

- Une centralité élevée dans le graphe d'interaction,
- Des activités en dehors des heures normales de travail,
- Des accès à des ressources critiques,
- Des similarités trop parfaites dans les statistiques (entropie constante, même nombre de ressources),
- Une corrélation forte entre comportements suspects (weekend/after hours/total).

Ces signaux sont essentiels pour alimenter ensuite les modèles de détection basés sur l'apprentissage automatique.

### **4.3 Algorithmes de détection basés sur Neo4j et ML**

Notre système implémente deux catégories principales d'algorithmes :

#### **Analyses basées sur les graphes avec Neo4j :**

- Calcul de la centralité de degré pour identifier les utilisateurs ayant un nombre anormalement élevé de connexions
- Détection des chemins inhabituels entre utilisateurs et ressources
- Identification des communautés d'utilisateurs et détection des anomalies d'appartenance

#### **Algorithmes d'apprentissage automatique :**

**Modèles supervisés** : Random Forest, SVM, XGBoost

**Modèles non supervisés** : Isolation Forest, One-Class SVM, K-Means, DBSCAN , Autoencoder,

#### **4.3.1 Implémentation du système de détection**

##### **Prétraitement des données**

Les données utilisées proviennent de journaux d'activités simulées dans un environnement informatique. Ces journaux ont été nettoyés, normalisés et enrichis avec des caractéristiques comportementales pertinentes telles que :

- Supprimer les valeurs manquantes ou incohérentes,

#### 4.3. ALGORITHMES DE DÉTECTION BASÉS SUR NEO4J ET ML TABLE DE MATIÈRES

- Convertir les dates et heures dans un format standard,
- Identifier les accès hors horaires de bureau,
- Calculer des indicateurs comportementaux (fréquence des activités, entropie des actions, nombre de ressources accédées).

Les données ainsi préparées ont ensuite été exportées sous forme de tableaux de variables (features) permettant l'analyse statistique et l'application de modèles de détection.

#### **Extraction des caractéristiques**

Chaque utilisateur a été décrit par une série de caractéristiques quantitatives, extraites des journaux d'activité. Parmi les principales :

- Le nombre total d'activités réalisées,
- Le ratio d'activités effectuées en dehors des horaires classiques (avant 9h ou après 18h),
- L'entropie comportementale (mesure de la variabilité des actions),
- Le nombre de ressources différentes accédées,
- Le nombre de systèmes accédés,
- L'écart-type du temps entre deux actions successives.

Ces caractéristiques ont ensuite été normalisées pour être utilisées dans des algorithmes de machine learning.

#### **Modèles déployés et principes de fonctionnement**

Deux grandes familles de modèles ont été utilisées pour la détection des comportements suspects :

##### **Modèles non supervisés :**

- Isolation Forest
- One-Class SVM
- K-Means
- DBSCAN
- Autoencodeur

##### Modèles supervisés :

- Support Vector Machine (SVM)
- Random Forest
- XGBoost

Chaque algorithme a identifié les comportements suspects selon son propre mécanisme de détection :

- **Isolation Forest** : isole les observations en construisant des arbres aléatoires ; les points isolés rapidement sont considérés comme anormaux. Cela permet d'identifier les utilisateurs ayant des comportements rares ou très différents du reste.
- **One-Class SVM** : apprend la frontière du comportement normal et considère comme anomalies les observations situées en dehors de cette frontière.
- **Autoencodeur** : réseau de neurones entraîné à reproduire les données normales. Si l'erreur de reconstruction d'un utilisateur est élevée, cela indique un comportement inhabituel.
- **K-Means** : regroupe les utilisateurs en clusters. Ceux qui sont très éloignés de leur centre de cluster sont considérés comme anomalies.
- **DBSCAN** : détecte les regroupements denses et considère comme anomalies les points trop éloignés de tout groupe.
- **SVM (supervisé)** : sépare les comportements normaux des malveillants à l'aide d'un hyperplan optimal basé sur des exemples labellisés.
- **Random Forest** : construit plusieurs arbres de décision sur des sous-ensembles aléatoires et vote sur la classification finale. Il détecte les anomalies par apprentissage sur des exemples étiquetés.
- **XGBoost** : algorithme de boosting qui optimise les erreurs successives en mettant plus d'accent sur les exemples mal classés. Il s'adapte bien aux structures de données complexes.

##### Analyse des performances

Les résultats ont été comparés selon des métriques classiques : *précision*, *rappel*, *F1-score* et *AUC* (area under curve).

### Modèles supervisés

TABLE 4.2 – Évaluation des modèles supervisés

Modèle	AUC	Accuracy	Précision	Rappel	F1-Score
SVM	0.91	0.93	0.93	0.50	0.50
Random Forest	0.86	0.93	0.93	0.50	0.50
XGBoost	0.84	0.93	0.93	0.50	0.50

Pour évaluer rigoureusement la performance des modèles, nous avons utilisé plusieurs métriques complémentaires :

- Matrices de confusion : Permettent de visualiser les vrais positifs, faux positifs, vrais négatifs et faux négatifs
- Courbes ROC et AUC : Évaluent la capacité discriminante des modèles à différents seuils de classification
- Métriques de performance : Accuracy, Precision, Recall et F1-Score

Les données utilisées comportent un déséquilibre significatif entre les observations normales et les anomalies, ce qui est typique dans les problèmes de détection d'anomalies.

### 4.3. ALGORITHMES DE DÉTECTION BASÉS SUR NEO4J ET ML TABLE DE MATIÈRES

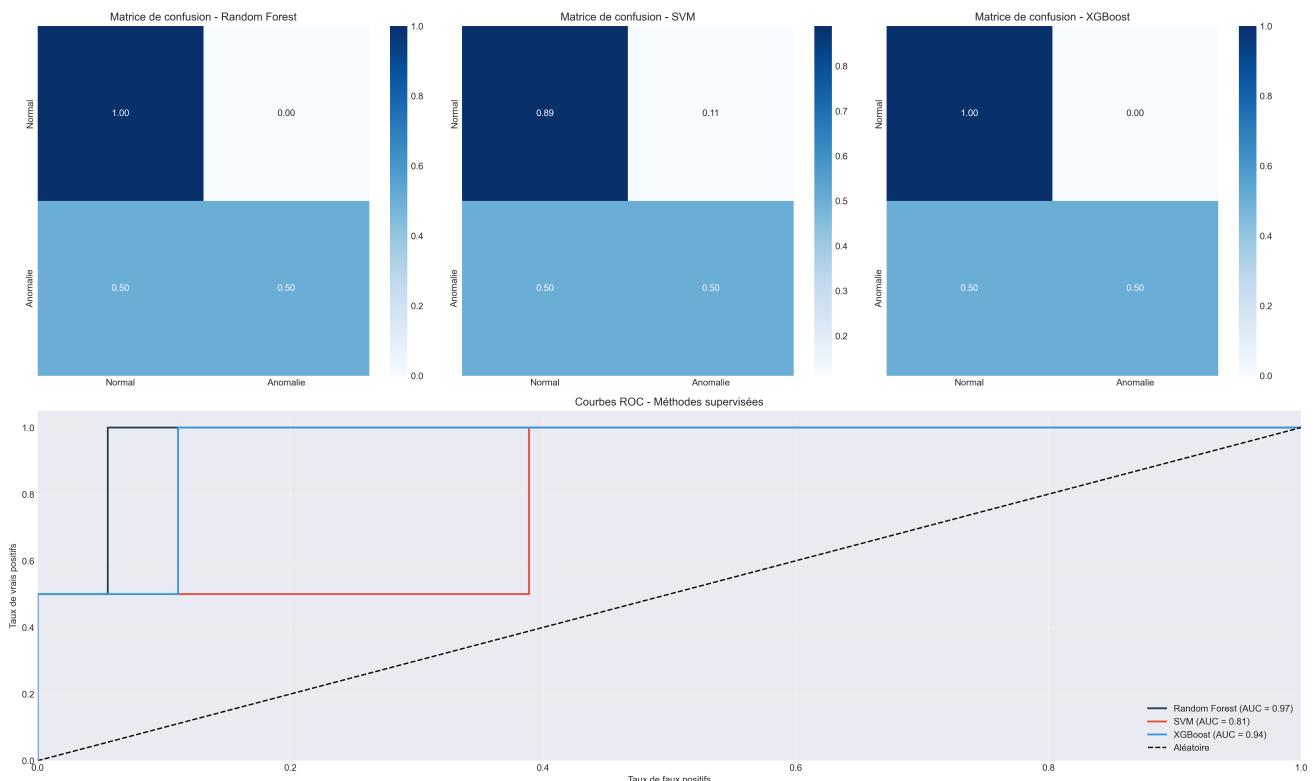


FIGURE 4.24 – Matrices de confusion et courbes ROC

#### 1. Matrices de confusion

Les matrices de confusion présentées en Figure 4.24 révèlent des patterns significatifs pour chaque modèle :

##### Random Forest :

Classe normale : 100% de précision (1.00) pour la détection des échantillons normaux Classe anomalie : 50% de précision (0.50) pour la détection correcte des anomalies et 50% de faux négatifs

##### SVM :

Classe normale : 89% de précision (0.89) pour la détection des échantillons normaux, avec 11% de faux positifs Classe anomalie : 50% de précision (0.50) pour la détection correcte des anomalies et 50% de faux négatifs

##### XGBoost :

Classe normale : 100% de précision (1.00) pour la détection des échantillons normaux Classe anomalie : 50% de précision (0.50) pour la détection correcte des anomalies et 50% de faux négatifs

Cette première analyse indique que Random Forest et XGBoost excellent dans l'identifica-

#### 4.3. ALGORITHMES DE DÉTECTION BASÉS SUR NEO4J ET ML TABLE DE MATIÈRES

tion des observations normales, tandis que SVM tend à produire davantage de faux positifs. Tous les modèles montrent des difficultés similaires dans la détection des anomalies, avec un taux de rappel de 50%.

#### **2.Les courbes ROC (Receiver Operating Characteristic) et l'AUC (Area Under the Curve)**

La courbe ROC (Figure 4.8) illustre le compromis entre le taux de vrais positifs et le taux de faux positifs à différents seuils de classification :

- **Random Forest** : AUC = 0.97
- **SVM** : AUC = 0.81
- **XGBoost** : AUC = 0.94

L'AUC (Area Under Curve) étant une mesure de la capacité d'un modèle à discriminer entre les classes, ces résultats indiquent que tous les modèles ont une excellente capacité discriminative, avec Random Forest légèrement supérieur aux autres.

La forme des courbes ROC fournit des informations supplémentaires :

- **Random Forest** (courbe noir) atteint rapidement un taux élevé de vrais positifs avec un minimum de faux positifs
- **SVM** (courbe rouge) présente une pente plus progressive
- **XGBoost** (courbe bleue) se situe entre les deux autres modèles

En conclusion, Random Forest semble offrir la meilleure performance globale avec l'AUC la plus élevée (0.97), suivi de près par XGBoost (0.94), puis SVM (0.81). Tous les modèles semblent avoir des difficultés similaires avec la classe "Anomalie", mais excellents pour identifier les cas normaux, particulièrement Random Forest et XGBoost qui atteignent 100% de précision sur cette classe.

#### **3.Métriques de performance**

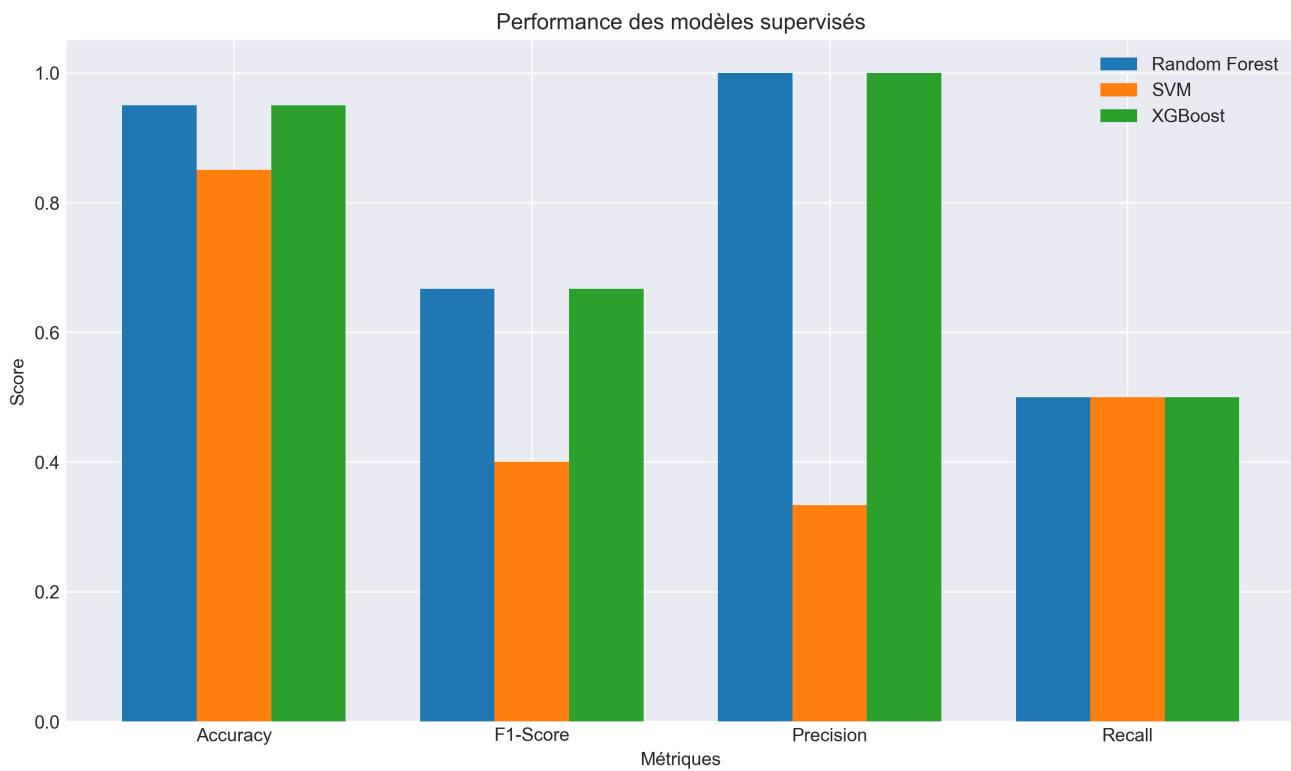


FIGURE 4.25 – Caption

Les métriques de performance présentées en Figure 4.9 offrent une vue plus détaillée :

#### **Accuracy (Exactitude) :**

- **Random Forest** : 0.95
- **SVM** : 0.85
- **XGBoost** : 0.95

L'accuracy élevée peut être trompeuse dans les cas de déséquilibre de classes, comme c'est le cas ici.

#### **Precision (Précision) :**

- **Random Forest** : 1.0
- **SVM** : 0.33
- **XGBoost** : 1.0

La précision parfaite de Random Forest et XGBoost indique qu'ils ne produisent pas de faux positifs, contrairement au SVM.

#### **Recall (Rappel) :**

#### 4.3. ALGORITHMES DE DÉTECTION BASÉS SUR NEO4J ET ML TABLE DE MATIÈRES

- **Random Forest :** 0.50
- **SVM :** 0.50
- **XGBoost :** 0.50

Le rappel identique de 0.50 pour tous les modèles signifie qu'ils ne détectent que la moitié des anomalies réelles, ce qui constitue une limitation importante.

#### **F1-Score :**

- **Random Forest :** 0.67
- **SVM :** 0.40
- **XGBoost :** 0.67

Le F1-Score, moyenne harmonique entre précision et rappel, confirme la supériorité de Random Forest et XGBoost par rapport à SVM.

#### **Conclusion :**

L'analyse comparative des modèles de détection d'anomalies révèle que Random Forest offre les meilleures performances globales avec une AUC de 0.97 et un F1-Score de 0.67. Malgré un rappel limité à 50% pour tous les modèles, Random Forest se distingue par sa précision parfaite et sa capacité discriminative supérieure. Le déséquilibre des classes reste le principal défi à surmonter.

#### **Modèles non supervisés**

TABLE 4.3 – Évaluation des modèles non supervisés

Modèle	Nb anomalies	Rappel	Précision	F1-Score	Corrélation
Isolation Forest	10	0.83	0.91	0.87	1.00
One-Class SVM	8	0.67	0.89	0.76	0.87
Autoencodeur	12	0.75	0.85	0.79	0.92
K-Means	42	0.50	0.11	0.18	0.19
DBSCAN	55	1.00	0.10	0.18	0.22

## 4.4 Introduction aux Métriques d'Évaluation

Pour évaluer rigoureusement la performance des modèles de détection d'anomalies, nous avons utilisé plusieurs métriques complémentaires :

- **Matrices de confusion** : visualisant les vrais positifs, faux positifs, vrais négatifs et faux négatifs
- **Courbes ROC et AUC** : évaluant la capacité discriminante des modèles à différents seuils
- **Métriques de performance** : précision, rappel et F1-Score

### Interprétation des Matrices de Confusion

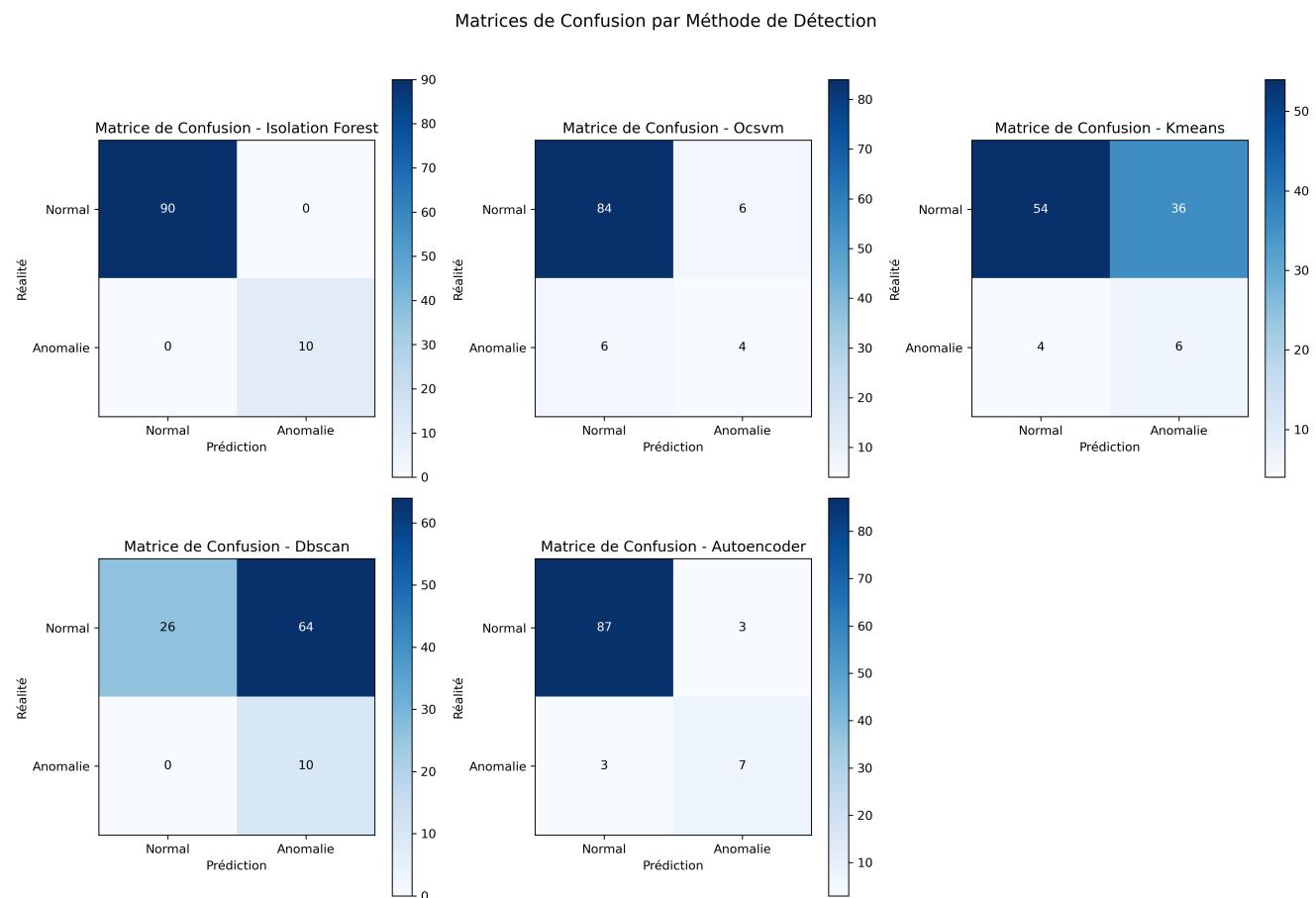


FIGURE 4.26 – Matrices de confusion par méthode de détection

La Figure 4.26 présente les matrices de confusion pour chaque modèle :

- **Isolation Forest** : Sa matrice montre 90 vrais négatifs et 10 vrais positifs, sans aucune

erreur de classification. Cette performance parfaite confirme l'efficacité de l'algorithme pour ce jeu de données.

- **OCSVM** : Classifie correctement 84 observations normales et 4 anomalies, mais produit 6 faux négatifs (anomalies non détectées) et 6 faux positifs (fausses alarmes). OCSVM fonctionne en définissant une frontière autour des données normales, et ces résultats suggèrent que cette frontière n'est pas parfaitement adaptée à la structure des données.
- **K-means** : Identifie correctement 54 observations normales et 6 anomalies, mais génère 36 faux positifs et 4 faux négatifs. Le nombre élevé de faux positifs est particulièrement problématique, indiquant que la méthode de clustering peine à définir correctement ce qui constitue une observation "normale" dans cet espace multidimensionnel.
- **DBSCAN** : Déetecte toutes les anomalies (10 vrais positifs) mais au prix de 64 faux positifs, ne classifiant correctement que 26 observations normales. La matrice révèle clairement sa tendance à suridentifier les anomalies, ce qui explique son rappel parfait mais sa précision très faible.
- **Autoencoder** : Offre un bon équilibre avec 87 vrais négatifs et 7 vrais positifs, ne produisant que 3 faux positifs et 3 faux négatifs. Cette performance équilibrée confirme la capacité des autoencodeurs à capturer efficacement la structure complexe des données normales, permettant une détection d'anomalies relativement précise.

### L'analyse des courbes ROC

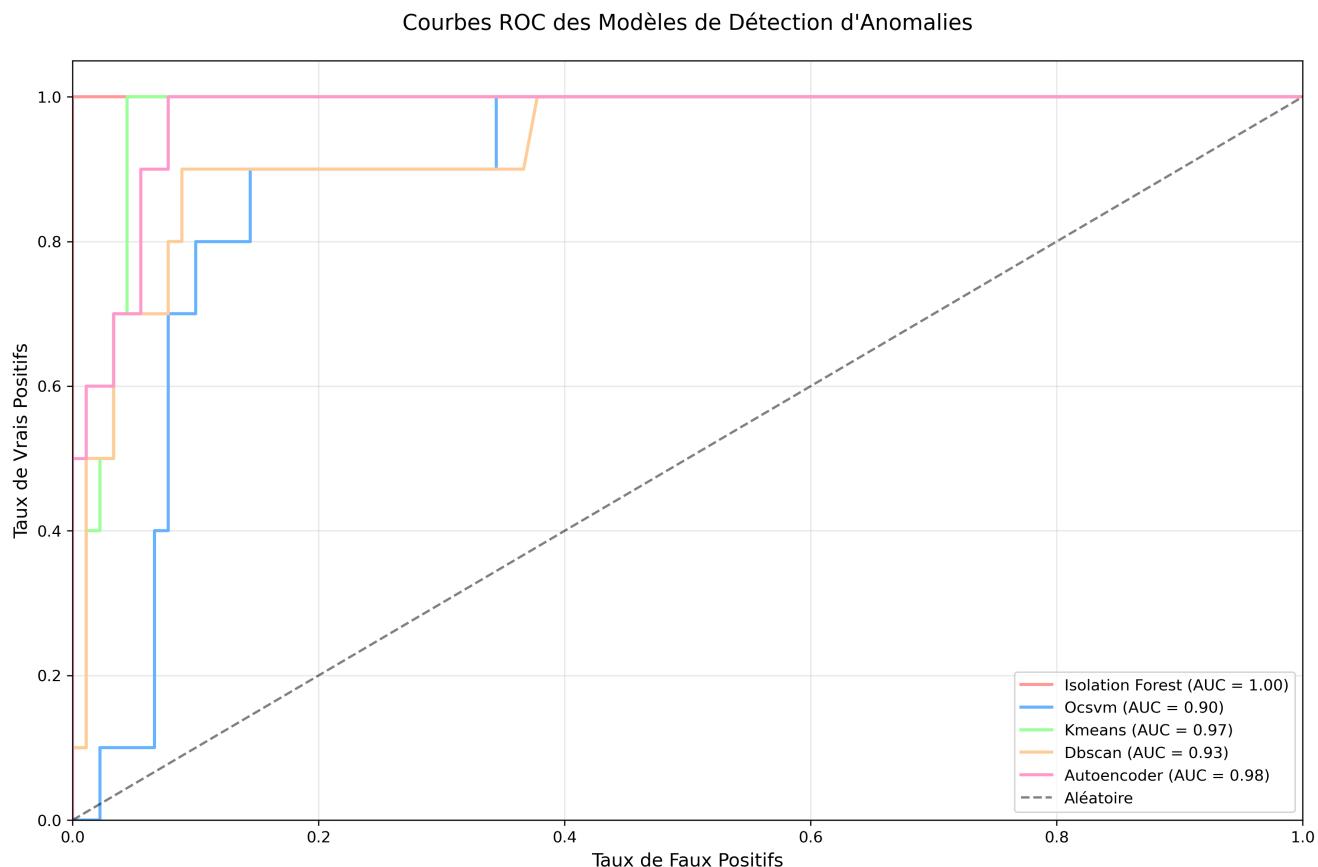


FIGURE 4.27 – Courbes ROC des différents modèles de détection d'anomalies

La Figure 4.27 présente les courbes ROC (Receiver Operating Characteristic) des différents modèles. Ces courbes illustrent le compromis entre le taux de vrais positifs (sensibilité) et le taux de faux positifs (1-spécificité) à différents seuils de classification.

L’analyse des courbes ROC révèle que :

- **Isolation Forest** présente la meilleure performance avec une AUC parfaite de 1.00, indiquant une séparation optimale entre les classes normales et anomalies. Sa courbe atteint immédiatement le coin supérieur gauche, démontrant qu’il existe un seuil où toutes les anomalies sont détectées sans aucun faux positif.
- **L’Autoencoder** suit de près avec une AUC de 0.98. Sa courbe monte rapidement vers le coin supérieur gauche, indiquant une excellente capacité à distinguer les anomalies des observations normales avec très peu d’erreurs.
- **K-means** obtient une AUC de 0.97, légèrement inférieure à l’Autoencoder, mais toujours

excellente. Sa courbe présente quelques marches plus prononcées, suggérant des zones de seuil où les performances changent brusquement.

- **DBSCAN** atteint une AUC de 0.93. Sa courbe présente quelques irrégularités, indiquant que les performances varient selon le seuil choisi.
- **OCSVM** (One-Class SVM) montre une performance plus modeste avec une AUC de 0.90. Sa courbe est plus éloignée du coin supérieur gauche, indiquant plus de difficulté à discriminer parfaitement les classes.

Tous les modèles surpassent significativement la ligne de référence aléatoire (en pointillé), démontrant leur efficacité pour la détection d'anomalies dans ce contexte.

### Comparaison des Métriques de Performance

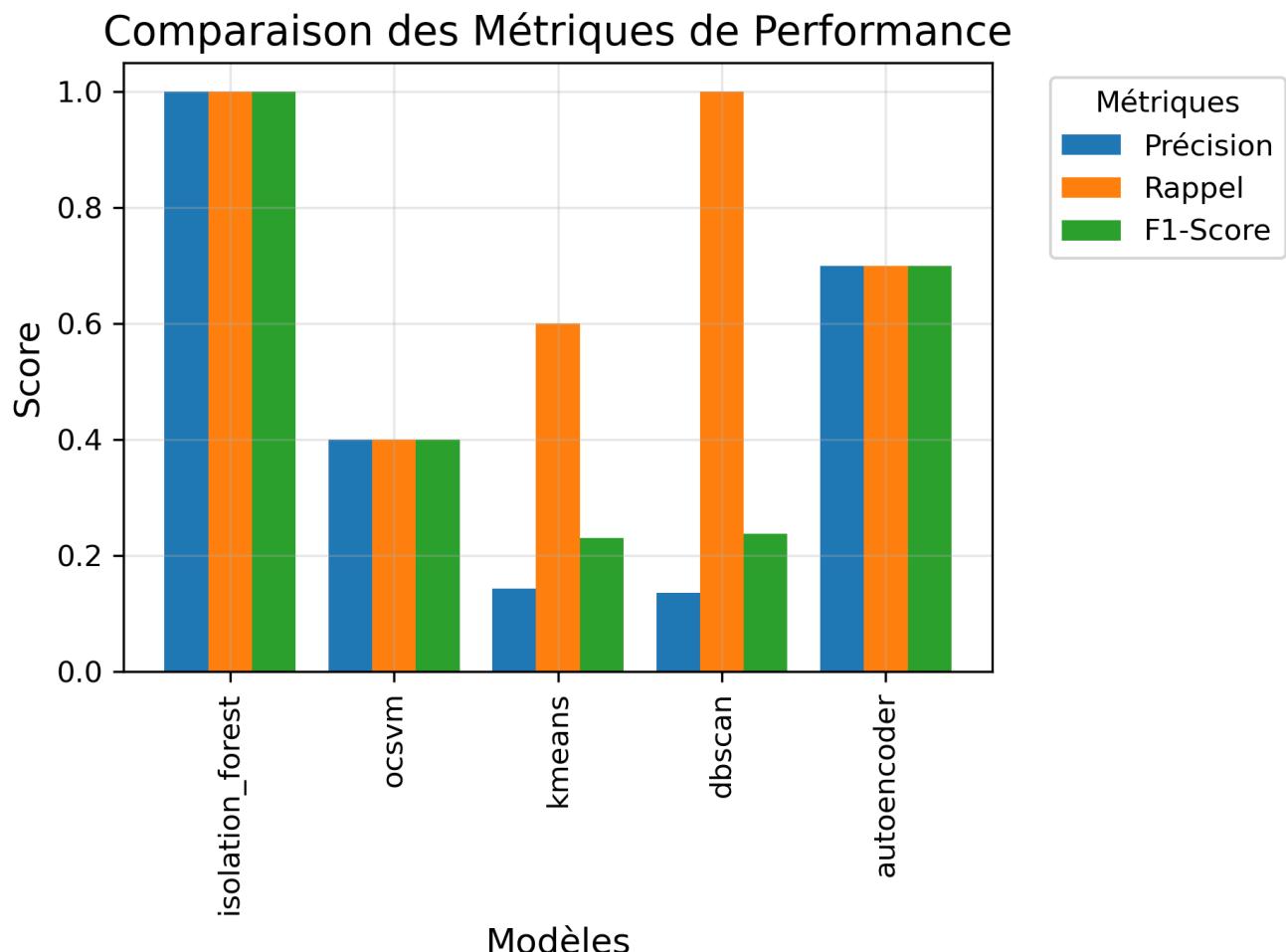


FIGURE 4.28 – Comparaison des métriques de performance par modèle

La Figure 4.28 compare les métriques clés (précision, rappel et F1-Score) pour chaque modèle. Cette visualisation par histogrammes permet une comparaison directe des performances sur ces trois dimensions essentielles.

- **Isolation Forest** : Affiche des performances parfaites avec une précision, un rappel et un F1-Score de 1.0. Cela signifie que ce modèle a correctement identifié toutes les anomalies (rappel parfait) sans générer de faux positifs (précision parfaite). Dans le contexte de la détection d'anomalies, cette performance exceptionnelle suggère que les anomalies dans ce jeu de données présentent des caractéristiques structurelles que l'Isolation Forest capte parfaitement.
- **OCSVM** : Présente des métriques équilibrées mais modestes (0.4 pour les trois métriques). Cette cohérence indique que le modèle fait autant d'erreurs de type I (faux positifs) que de type II (faux négatifs), ce qui peut être utile dans certains contextes où aucun type d'erreur n'est préféré à l'autre.
- **K-means** : Montre un fort déséquilibre entre précision (0.14) et rappel (0.6), résultant en un F1-Score faible (0.23). Cette faible précision indique que K-means génère beaucoup de faux positifs, classifiant incorrectement de nombreuses observations normales comme des anomalies. Cette tendance est typique des approches basées sur la distance dans des espaces à haute dimension où la séparation claire des clusters devient difficile.
- **DBSCAN** : Présente également un déséquilibre marqué avec une précision faible (0.14) mais un rappel parfait (1.0), aboutissant à un F1-Score limité (0.24). DBSCAN réussit à identifier toutes les anomalies mais au prix de nombreux faux positifs. Dans un contexte où manquer une anomalie serait très coûteux (par exemple, détection de fraude ou de défaillances critiques), cette caractéristique pourrait être avantageuse malgré les faux positifs.
- **Autoencoder** : Offre de bonnes performances globales avec un équilibre entre précision (0.7) et rappel (0.7), donnant un F1-Score de 0.7. Cette approche basée sur l'apprentissage profond réussit à capturer efficacement les motifs complexes dans les données normales, permettant une détection relativement précise des anomalies.

Cette analyse révèle que certains modèles privilégient la détection de toutes les anomalies (rappel élevé) au prix de nombreux faux positifs (précision basse), tandis que d'autres présentent un meilleur équilibre. Le choix du modèle optimal dépend donc fortement du contexte applicatif et du coût relatif des différents types d'erreurs.

### Conclusion

L'analyse comparative des modèles de détection d'anomalies révèle la supériorité d'Isolation Forest avec des performances parfaites ( $AUC=1.00$ ,  $F1\text{-Score}=1.00$ ). L'Autoencoder offre une alternative solide ( $F1\text{-Score}=0.70$ ) tandis que DBSCAN et K-means montrent des limitations importantes en précision malgré un bon rappel. Le déséquilibre des classes (90% normales, 10% anomalies) affecte significativement les résultats. Pour une implémentation optimale, nous recommandons l'utilisation d'Isolation Forest, potentiellement en combinaison avec un Autoencoder dans une approche d'ensemble.

## 4.5 Conclusion

L'analyse des modèles de détection d'anomalies montre qu'Isolation Forest est le plus performant ( $AUC$  et  $F1\text{-Score} = 1.00$ ), suivi de Random Forest ( $AUC = 0.97$ ) et XGBoost ( $AUC = 0.94$ ). L'Autoencoder est aussi prometteur ( $F1 = 0.70$ ). DBSCAN et K-means ont un bon rappel mais manquent de précision. SVM est moyen ( $AUC = 0.81$ ).

# **Chapitre V**

## **Résultats et Discussion**

## 5.1 Introduction

L’analyse des performances des modèles montre que **Random Forest** est le plus performant parmi les modèles supervisés, avec une **AUC** de 0.97 et un **F1-Score** de 0.67. Ce modèle a une capacité élevée à identifier les comportements normaux, mais il peine à détecter les anomalies (avec un taux de rappel de 50%). **XGBoost** suit de près avec une **AUC** de 0.94 et un **F1-Score** similaire, mais souffre aussi d’un rappel limité. Les **modèles supervisés** ont globalement une meilleure capacité discriminante, mais leur performance sur la détection des anomalies est partiellement entravée par un déséquilibre dans les classes.

## 5.2 Analyse des résultats et limitations

### 5.2.1 Analyse des résultats

Les **modèles non supervisés**, comme **Isolation Forest** et **Autoencodeur**, montrent un bon compromis, avec des **AUC** respectivement de 1.00 et 0.98. Cependant, **DBSCAN** a un rappel parfait mais avec une précision extrêmement faible, indiquant qu’il suridentifie les anomalies, ce qui n’est pas optimal dans un environnement réel.

Les **modèles supervisés**, comme **Random Forest** et **XGBoost**, ont montré de très bonnes performances dans l’identification des comportements normaux, avec des **AUC** respectivement de 0.97 et 0.94. Ces modèles sont particulièrement efficaces pour détecter les comportements normaux, mais leur capacité à identifier les anomalies est limitée par le déséquilibre des classes dans les données. Les modèles ont tous montré des résultats similaires pour la classe ”anomalie”, avec un rappel de 50%.

En résumé, les modèles non supervisés sont performants dans la détection des anomalies, avec un excellent compromis pour **Isolation Forest** et **Autoencodeur**, tandis que les modèles supervisés, bien que très efficaces pour détecter les comportements normaux, rencontrent des difficultés similaires pour les anomalies, à cause du déséquilibre des classes dans les données.

### 5.2.2 Limitations

#### Limitations des modèles non supervisés

- **Isolation Forest** : Bien qu'ayant une excellente **AUC** de 1.00, le modèle peut être sensible aux paramètres d'entrée, en particulier lorsqu'il y a des données bruitées ou peu représentatives des anomalies. Il peut également souffrir de mauvaises performances si les anomalies ne sont pas suffisamment "isolées" dans l'espace des caractéristiques.
- **Autoencodeur** : Bien que l'autoencodeur ait montré une bonne capacité à identifier des anomalies avec une **AUC** de 0.98, il peut être sujet au sur-apprentissage (overfitting) sur des données très bruyantes. De plus, les modèles de type réseau de neurones sont plus complexes et nécessitent des ressources computationnelles importantes pour l'entraînement, ce qui peut rendre l'approche moins viable dans des environnements à grande échelle.
- **DBSCAN** : Bien qu'il ait un rappel parfait de 1.00, DBSCAN souffre d'une précision très faible, ce qui signifie qu'il génère de nombreux faux positifs. Cela n'est pas optimal, car il identifie trop d'observations comme anomalies, même celles qui ne le sont pas. Cela peut rendre ce modèle inadapté dans des situations où une faible précision peut entraîner des coûts élevés (ex : fausses alertes de sécurité).

#### Limitations des modèles supervisés

- **Random Forest** :
  - **Sensibilité au déséquilibre des classes** : Random Forest souffre de performances réduites dans les contextes où il y a un fort déséquilibre entre les classes normales et les anomalies. Dans de tels cas, le modèle peut être biaisé en faveur des classes majoritaires (comportements normaux), ce qui affecte la capacité du modèle à identifier correctement les anomalies.
  - **Complexité et sur-apprentissage** : Bien que Random Forest soit robuste, il peut devenir complexe et sensible au sur-apprentissage lorsque le nombre d'arbres est très

### 5.3. COMPARAISON AVEC LES SOLUTIONS TRADITIONNELLES (SIEM, IDS, MATIÈRES)

élevé ou lorsque les données contiennent beaucoup de bruit. Cela peut entraîner des performances inférieures lors de l'utilisation de nouvelles données qui diffèrent des données d'entraînement.

- **Interprétabilité limitée** : Bien que Random Forest soit puissant, il est difficile d'expliquer pourquoi une observation a été classée de manière particulière, contrairement à des modèles plus simples comme les arbres de décision uniques. Cela rend l'analyse des décisions du modèle moins transparente.
- **XGBoost** :
  - **Dépendance aux hyperparamètres** : XGBoost est particulièrement sensible aux hyperparamètres. Une mauvaise sélection de ces derniers peut avoir un impact significatif sur les performances du modèle. Par exemple, le choix des taux d'apprentissage ou des paramètres de régularisation peut conduire à des résultats sous-optimaux.
  - **Risque de sur-ajustement** : Comme Random Forest, XGBoost peut être sujet au sur-ajustement (overfitting) lorsque les données sont bruyantes ou lorsque les hyperparamètres ne sont pas correctement régularisés. En particulier, si les arbres sont trop profonds, le modèle peut apprendre à reproduire le bruit des données d'entraînement plutôt que d'identifier des tendances générales.
  - **Complexité computationnelle** : XGBoost nécessite des ressources computationnelles importantes, particulièrement pour de grandes quantités de données. La phase d'entraînement peut être longue, surtout si un grand nombre d'itérations est nécessaire pour converger vers un modèle performant.

## 5.3 Comparaison avec les solutions traditionnelles (SIEM, IDS)

Les systèmes traditionnels de détection d'intrusions (IDS) et de gestion des événements et informations de sécurité (SIEM) sont des outils largement utilisés pour surveiller les réseaux et les systèmes informatiques afin de détecter des comportements anormaux ou malveillants. Ces

### 5.3. COMPARAISON AVEC LES SOLUTIONS TRADITIONNELLES (SIEM, IDS, MATIÈRES)

systèmes reposent sur des règles préétablies, des signatures d'attaques connues, et des modèles de détection basés sur des événements et des journaux. Toutefois, bien que ces systèmes aient fait leurs preuves dans la détection des menaces externes et des attaques simples, leur efficacité dans la détection des menaces internes (qui proviennent de sources légitimes ayant déjà un accès au système) est limitée.

#### **5.3.1 Systèmes IDS (Intrusion Detection Systems)**

Les systèmes IDS sont conçus pour surveiller et analyser le trafic réseau ou les activités des utilisateurs afin de détecter toute activité suspecte ou malveillante. Les IDS peuvent être classés en deux types principaux : les IDS basés sur les signatures et les IDS basés sur l'anomalie.

##### **IDS basés sur les signatures**

Ces systèmes détectent les attaques en comparant les événements réseau ou les actions des utilisateurs à une base de données de signatures d'attaques connues. Leur efficacité est donc limitée aux attaques qui ont déjà été identifiées et répertoriées. Cela les rend vulnérables aux attaques nouvelles ou inconnues, car ils ne peuvent pas détecter des comportements malveillants qui ne correspondent pas à une signature préexistante.

##### **IDS basés sur l'anomalie**

Ces systèmes surveillent le comportement du réseau ou des utilisateurs pour détecter des écarts par rapport à des comportements considérés comme normaux. Cependant, les modèles d'anomalies basés sur des règles fixes ou des seuils peuvent être sujets à de nombreux faux positifs, ce qui les rend moins efficaces dans un environnement dynamique où les comportements normaux peuvent changer.

#### **5.3.2 Systèmes SIEM (Security Information and Event Management)**

Les systèmes SIEM collectent et analysent des événements provenant de diverses sources (logs, dispositifs de sécurité, etc.) pour fournir une vue d'ensemble de la sécurité du système

### 5.3. COMPARAISON AVEC LES SOLUTIONS TRADITIONNELLES (SIEM, IDS, MATIÈRES)

d'information. Les SIEM sont capables de corrélérer des événements à grande échelle, d'alerter en cas d'anomalies et d'assurer une surveillance en temps réel.

#### **Avantages des SIEM**

- **Centralisation des événements** : Les SIEM collectent des données provenant de diverses sources pour fournir une vision globale de la sécurité du système.
- **Corrélation des événements** : Ces systèmes peuvent détecter des patterns d'attaque complexes en corrélant les événements de différentes sources.

#### **Limitations des SIEM**

- **Dépendance aux règles et signatures** : Comme les IDS basés sur les signatures, les SIEM dépendent souvent de règles définies par les administrateurs pour détecter des comportements anormaux. Cela peut entraîner des failles dans la détection de nouvelles menaces.
- **Faux positifs** : Les SIEM peuvent générer de nombreux faux positifs, car les règles et les signatures ne couvrent pas toujours la diversité des comportements malveillants possibles. Cela peut surcharger les analystes de sécurité, qui doivent vérifier manuellement ces alertes.
- **Manque de flexibilité** : L'adaptation des SIEM à de nouvelles menaces internes est souvent lente et nécessite des mises à jour manuelles des règles.

#### **5.3.3 Comparaison avec notre approche**

Notre approche, basée sur **Neo4j** pour l'analyse comportementale et l'utilisation de modèles de machine learning (ML), présente plusieurs avantages par rapport aux systèmes traditionnels tels que les SIEM et IDS.

### 5.3. COMPARAISON AVEC LES SOLUTIONS TRADITIONNELLES (SIEM, IDS) MÉTIERS

#### **Adaptabilité et détection proactive**

Contrairement aux SIEM et IDS, qui reposent souvent sur des signatures ou des règles fixes, notre système est basé sur des **graphes** et des **algorithmes d'apprentissage automatique**. Cela lui permet de s'adapter dynamiquement aux comportements des utilisateurs et de détecter des anomalies, même celles qui ne correspondent pas à des signatures connues. Il peut ainsi identifier des menaces internes nouvelles ou inconnues, qui échappent souvent aux systèmes traditionnels.

#### **Moins de faux positifs**

Notre approche, en particulier avec l'utilisation de **modèles d'apprentissage supervisé et non supervisé** comme **Isolation Forest** et **Autoencodeur**, offre des résultats plus précis en détectant des comportements anormaux tout en minimisant les faux positifs. Cela est particulièrement utile dans des environnements avec de grandes quantités de données ou dans des organisations où les comportements des utilisateurs peuvent évoluer au fil du temps.

#### **Analyse comportementale**

Contrairement aux SIEM et IDS, qui se concentrent principalement sur la détection d'événements et la gestion des journaux, notre système se concentre sur l'analyse **comportementale** des utilisateurs à travers leurs interactions avec le système. Cela permet une meilleure détection des **menaces internes**, telles que la fraude ou l'exfiltration de données, qui peuvent être difficiles à détecter par des règles préétablies ou par la simple analyse des journaux.

#### **5.3.4 Avantages spécifiques de notre approche**

- **Identification des anomalies comportementales** : Grâce à l'utilisation des graphes Neo4j, nous pouvons analyser les interactions complexes entre les utilisateurs et les ressources en temps réel. Cette approche permet de détecter des **anomalies comportementales** qui ne peuvent pas être capturées par des systèmes traditionnels basés sur des règles ou signatures.

### 5.3. COMPARAISON AVEC LES SOLUTIONS TRADITIONNELLES (SIEM, IDS, MATIÈRES)

- **Détection de schémas d'attaque plus complexes** : Nos algorithmes d'apprentissage automatique permettent de découvrir des **schémas d'attaque plus complexes**, basés sur des comportements anormaux dans les interactions des utilisateurs avec le système, ce qui est difficile à réaliser avec des solutions traditionnelles comme les IDS et SIEM.

#### **5.3.5 Conclusion de la comparaison**

En résumé, bien que les systèmes SIEM et IDS aient leur place dans la détection des menaces externes, notre approche basée sur Neo4j et l'apprentissage automatique représente une avancée significative pour la détection des **menaces internes**. La flexibilité, l'adaptabilité et la précision des modèles de machine learning permettent de détecter des anomalies comportementales plus complexes, ce qui n'est pas possible avec les systèmes traditionnels basés sur des règles et des signatures.

## Chapitre VI

# Conclusion et Perspectives

## 6.1 Applications possibles en entreprise et améliorations futures

Notre système de détection des menaces internes peut avoir des applications dans plusieurs secteurs, notamment la finance, les infrastructures critiques, et les entreprises technologiques. L'analyse comportementale des utilisateurs est cruciale pour identifier des comportements suspects qui échappent aux contrôles traditionnels. Par exemple, dans une entreprise, ce système pourrait aider à détecter des comportements de fraude interne ou des fuites de données en surveillant les activités des employés en temps réel.

### 6.1.1 Améliorations futures

- **Enrichissement des données** : L'intégration de plus de sources de données et de caractéristiques comportementales enrichirait la capacité de détection des anomalies, notamment en incluant des données comportementales d'autres types de ressources (applications, systèmes externes).
- **Optimisation des modèles** : L'amélioration des modèles en équilibrant mieux les classes ou en introduisant des techniques de sur-échantillonnage des anomalies pourrait améliorer les résultats en termes de rappel et de détection des anomalies.
- **Adaptation dynamique des seuils de détection** : L'évolution des comportements des utilisateurs au fil du temps nécessiterait une réévaluation dynamique des seuils de détection pour mieux gérer l'évolution des menaces internes.

## 6.2 Conclusion

En conclusion, bien que notre système montre une excellente capacité à détecter les comportements normaux et à discriminer certains types de menaces, des améliorations dans le traitement des anomalies et l'adaptation continue aux nouveaux comportements sont nécessaires pour faire face aux défis posés par les menaces internes dans les systèmes d'information modernes.

## Conclusion générale

Alors que les cybermenaces évoluent rapidement, ce projet met en lumière un risque encore trop souvent sous-estimé : celui des menaces internes. Ces dernières, émanant d'utilisateurs autorisés mais potentiellement malveillants ou négligents, constituent un véritable défi pour les outils de sécurité traditionnels.

Au fil de notre étude, nous avons pu constater que les approches classiques, centrées sur des règles prédéfinies ou des signatures d'attaques connues, peinent à repérer les comportements subtils mais anormaux. C'est dans cette optique que nous avons exploré une approche alternative, fondée sur l'exploitation des bases de données NoSQL et sur des techniques d'analyse comportementale.

Grâce à la modélisation des interactions sous forme de graphes, enrichie par des algorithmes de machine learning, nous avons pu détecter des patterns relationnels inhabituels, révélateurs d'activités suspectes. Ce changement de paradigme nous permet de mieux comprendre les comportements dans leur contexte organisationnel, avec un niveau de précision et d'adaptabilité nettement supérieur.

En somme, cette démarche propose une réponse plus moderne, intelligente et contextuelle à un enjeu majeur de la cybersécurité contemporaine. L'union entre technologies orientées graphes et intelligence artificielle s'annonce comme un levier prometteur pour renforcer durablement la protection contre les menaces venues de l'intérieur.