ÉCOLE POLYTECHNIQUE DE L'UNIVERSITÉ DE NANTES
DÉPARTEMENT D'INFORMATIQUE

RAPPORT DE RECHERCHE ET DÉVELOPPEMENT

Analyse automatique des cybermenaces par Machine Learning sur des données de Honeypots

Luka FORTUN

October 25, 2025

encadré par Rebiha SOUADIH

— —

LABORATOIRE DES SCIENCES DU NUMÉRIQUES DE NANTES
CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

coordinateur : Vincent RICORDEL

Avertissement

Toute reproduction, même partielle, par quelque procédé que ce soit, est interdite sans autorisation préalable.

Une copie par xérographie, photographie, photocopie, film, support magnétique ou autre, constitue une contrefaçon passible des peines prévues par la loi.

Analyse automatique des cybermenaces par Machine Learning sur des données de Honeypots

Luka FORTUN

Résumé

Face à l'automatisation croissante des cyberattaques, les honeypots sont devenus des outils incontournables pour la collecte de renseignement sur les menaces (Cyber Threat Intelligence). Cependant, le volume massif et l'hétérogénéité des données générées allant de simples métadonnées réseau à des commandes Shell offusquées rendent l'analyse humaine impossible. Ce projet de Recherche Développement vise à concevoir un pipeline d'analyse automatisé et hybride. En s'appuyant sur une étude approfondie de la littérature récente (2020-2025), nous proposons une architecture distinguant deux flux de traitement. D'une part, l'utilisation d'algorithmes de Machine Learning supervisé (Random Forest) pour la classification rapide des flux structurés et des malwares connus. D'autre part, l'intégration innovante de Grands Modèles de Langage (LLM) et de techniques RAG (Retrieval-Augmented Generation) pour l'analyse sémantique et l'explicabilité des commandes complexes issues de honeypots à haute interaction (ex : Cowrie). Ce rapport présente l'état de l'art, les choix architecturaux et la méthodologie retenue pour combiner performance temps-réel et profondeur d'analyse.

Catégories et descripteurs de sujets : K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Invasive software, Unauthorized access*; I.2.6 [**Artificial Intelligence**]: Learning—*Concept learning, Induction*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*; H.2.8 [**Database Management**]: Database Applications—*Data mining*

Termes généraux : Security, Algorithms, Experimentation, Measurement, Performance.

Mots-clés additionnels et phrases : Honeypot, Machine Learning, Random Forest, LLM, Cyber Threat Intelligence, Log Analysis.

Remerciements

Table des matières

1 Introduction

Ce premier chapitre pose le cadre général du projet de recherche. Face à la recrudescence et à la complexité des cyberattaques, l'analyse manuelle des données de sécurité atteint ses limites. Nous débuterons par une description du problème (Section 1.1), en détaillant le contexte des honeypots et les verrous technologiques liés au traitement massif des logs. Nous définirons ensuite les objectifs opérationnels et scientifiques (Section 1.2) du projet. La section suivante présentera le travail réalisé (Section 1.3) durant cette première phase d'étude. Enfin, nous présenterons l'organisation globale du rapport (Section 1.4).

1.1 Description du problème

1.1.1 Contexte

Dans un monde de plus en plus numérisé, la sécurité des systèmes d'information constitue un enjeu majeur pour les entreprises, les institutions publiques et les infrastructures critiques. Les cyberattaques sont désormais quotidiennes et se caractérisent par une grande diversité de techniques : exploitation de failles, attaques par dictionnaire, injections de code, ou encore détournement de protocoles. Afin d'étudier et de comprendre ces menaces, les chercheurs et ingénieurs en cybersécurité déploient des environnements de surveillance spécifiques appelés honeypots.

Un honeypot est un système volontairement vulnérable, configuré pour attirer les attaquants et enregistrer leurs activités. Ces dispositifs permettent de collecter des données réelles sur les comportements des cybercriminels et d'enrichir la connaissance en matière de menaces. Les journaux d'activité (logs) générés par les honeypots sont une source d'information précieuse : ils contiennent des traces détaillées d'intrusions, de commandes exécutées, de connexions réseau et d'autres événements liés aux attaques.

Cependant, ces données sont également massives, non structurées et souvent bruitées. Leur analyse manuelle est longue, complexe et sujette à l'erreur humaine. Dans ce contexte, l'usage de l'intelligence artificielle et plus

particulièrement du Machine Learning constitue une approche prometteuse pour automatiser l'analyse et la classification des incidents de sécurité. Le Machine Learning permet en effet d'identifier des motifs récurrents dans les données, de détecter des anomalies et d'extraire des connaissances exploitables à partir de volumes importants de logs.

Le projet s'inscrit dans cette perspective : il vise à concevoir et évaluer un pipeline complet de data science capable d'exploiter les données issues de honeypots pour identifier, classer et prioriser automatiquement les cyberattaques. Une extension de cette approche, via le Deep Learning, pourra être envisagée pour les cas plus complexes, notamment ceux impliquant des séquences temporelles de commandes ou de comportements malveillants.

1.1.2 Problématique

Les honeypots génèrent des volumes considérables de données brutes, mais transformer ces informations en renseignement de sécurité exploitable demeure un défi majeur. Les approches manuelles atteignent rapidement leurs limites face à la quantité, la variabilité et la complexité des données capturées.

La problématique générale de ce projet peut ainsi être formulée de la manière suivante :

Comment concevoir et mettre en œuvre un système automatisé d'analyse de données issues de honeypots, basé sur des techniques de Machine Learning, afin d'améliorer la détection et la classification des cyberattaques ?

Cette question regroupe plusieurs enjeux techniques et méthodologiques :

Définir un pipeline d'analyse capable de transformer des logs bruts en données exploitables ;

Identifier les algorithmes de Machine Learning les plus performants selon la nature des attaques observées ;

Évaluer la pertinence et la robustesse des modèles obtenus à l'aide de métriques standardisées ;

Explorer, le cas échéant, la valeur ajoutée du Deep Learning pour certains types de données ou de comportements.

La finalité du projet est de proposer une solution d'ingénierie reproductible et évolutive, qui pourrait être intégrée dans des outils de détection ou de veille de sécurité.

1.1.3 Problèmes identifiés

Afin de répondre à cette problématique, plusieurs axes de travail et verrous techniques ont été identifiés :

1. **Préparation et structuration des données**
   Les données issues de honeypots, souvent sous forme de logs textuels, doivent être nettoyées, normalisées et structurées avant toute analyse. Cette étape comprend le prétraitement (suppression du bruit, horodatage, extraction de champs pertinents) et le feature engineering (création de variables statistiques ou comportementales). La qualité de cette phase conditionne la performance des modèles d'apprentissage.
2. **Sélection et implémentation des modèles de Machine Learning**

Différents algorithmes (Random Forest, SVM, k-NN, Gradient Boosting, etc.) devront être développés, entraînés et comparés afin d'identifier celui offrant le meilleur compromis entre précision, temps d'exécution et capacité de généralisation. L'objectif est de classifier les attaques selon leur nature ou leur pattern comportemental.

3. **Évaluation des performances et comparaison avec l'état de l'art**

   Les modèles développés seront évalués à l'aide de métriques standards telles que la précision, le rappel et le F1-score. Une analyse critique des résultats permettra de situer la performance obtenue par rapport aux approches déjà publiées dans la littérature scientifique.

4. **Exploration du Deep Learning (si applicable)**

   Si les résultats du Machine Learning sont encourageants et que le calendrier du projet le permet, une étude complémentaire portera sur l'application du Deep Learning (RNN, LSTM, etc.) pour analyser des séquences temporelles de commandes ou de connexions. Cette approche pourrait permettre de détecter des comportements plus complexes et d'améliorer la précision globale du système.

5. **Intégration et perspectives d'exploitation**

   Enfin, une réflexion sera menée sur la mise en production d'un tel système : comment intégrer ce pipeline d'analyse dans un outil de surveillance automatisée ? Quels seraient les besoins en termes de maintenance, de mise à jour des modèles et de compatibilité avec des plateformes existantes de cybersécurité (SIEM, IDS, etc.) ?

1.2 Objectifs

L'objectif principal de ce projet est de concevoir et d'évaluer un pipeline automatisé pour l'analyse de données issues de honeypots à l'aide de techniques de Machine Learning. L'ambition est d'extraire automatiquement, à partir de journaux (logs) bruts, des informations exploitables sur les cybermenaces, de classifier les attaques par type et de prioriser les incidents de sécurité détectés.

Afin d'atteindre cet objectif général, plusieurs objectifs spécifiques ont été définis :

1. **Réaliser une étude bibliographique approfondie**

   Identifier et analyser les travaux existants portant sur l'analyse de logs de sécurité à l'aide de méthodes de Machine Learning et de Deep Learning, en particulier dans le contexte des honeypots.

2. **Acquérir et explorer un jeu de données représentatif**

   Sélectionner un jeu de données public de logs de honeypots (par exemple, Cowrie), puis effectuer une analyse exploratoire afin d'en comprendre la structure, les caractéristiques principales et les défis potentiels (bruit, déséquilibre des classes, volume, etc.).

3. **Mettre en œuvre une phase de prétraitement et de feature engineering**

Développer une méthodologie de nettoyage, de normalisation et de transformation des logs bruts en données exploitables par des algorithmes d'apprentissage automatique. Concevoir des représentations pertinentes pour différents types de modèles (traditionnels et profonds).

4. **Développer et évaluer des modèles de Machine Learning et de Deep Learning**

   Implémenter plusieurs classifieurs (Random Forest, SVM, Régression Logistique, etc.) et, si le temps le permet, expérimenter des approches de Deep Learning (LSTM, CNN) pour l'analyse séquentielle des logs. Évaluer leurs performances à l'aide de métriques standards telles que la précision, le rappel et le score F1.

5. **Analyser et interpréter les résultats obtenus**

   Comparer les performances des différents modèles, identifier leurs points forts et leurs limites, et évaluer la valeur ajoutée de l'automatisation pour la production de renseignement sur les menaces (threat intelligence).

6. **Comparer les résultats aux objectifs initiaux et documenter les limites**

   Mettre en perspective les résultats atteints par rapport aux attentes de départ, en identifiant les contraintes rencontrées et les perspectives d'amélioration futures.

1.3 Travail réalisé

La première phase de ce projet s'est concentrée sur la consolidation des bases théoriques et la définition de l'architecture technique. Les principales réalisations sont les suivantes :

— Analyse systématique de l'état de l'art (2020-2025) : Une revue exhaustive de la littérature a été menée, couvrant plus d'une dizaine d'articles de référence. Ce travail a permis d'identifier les limites du Deep Learning pour les logs simples et d'isoler le Random Forest comme l'algorithme de référence pour la performance.
— Étude comparative des architectures de Honeypots : Nous avons caractérisé les distinctions fondamentales entre les honeypots à faible interaction (type Dionaea) et à haute interaction (type Cowrie), justifiant la nécessité d'une approche analytique différenciée.
— Définition de la stratégie Hybride : Sur la base des travaux de Lanka et al. (2024) et Ali et al. (2025) , nous avons conçu une proposition d'architecture combinant la rapidité du ML statistique pour le filtrage et la puissance des LLM pour l'analyse sémantique des commandes inconnues.
— Planification et Gestion de projet : Établissement d'un calendrier prévisionnel, suivi hebdomadaire de l'avancement et identification des verrous technologiques (accès aux API LLM, choix du dataset).

1.4 Report organisation

Ce document est structuré de la manière suivante :

— Le Chapitre 2 dresse un État de l'art complet. Il définit la typologie des honeypots, analyse les défis liés aux données de cybersécurité, et compare de manière critique les performances des algorithmes de Machine Learning, Deep Learning et d'IA Générative appliqués à ce domaine.

— Le Chapitre 3 détaille nos Propositions. Il présente l'architecture hybride retenue (Random Forest + LLM), justifie le choix des technologies (Cowrie, Dionaea, Mistral/GPT) et décrit le pipeline de traitement des données envisagé.

2

État de l'art : Analyse automatisée des données de Honeypots par l'Intelligence Artificielle

L'automatisation de la Threat Intelligence nécessite une compréhension fine à la fois des sources de données et des algorithmes disponibles. Ce chapitre propose une analyse critique de la littérature scientifique récente (2020-2025). Nous établirons d'abord les fondements techniques et théoriques (Sections 2.2 et 2.3), en définissant les architectures de honeypots et la typologie des données collectées. Après avoir défini les métriques de performance et protocoles de validation (Section 2.4) essentiels à notre évaluation, nous analyserons comparativement trois familles d'approches : le Machine Learning supervisé (Section 2.5) pour la rapidité, le Deep Learning (Section 2.6) pour la séquentialité, et enfin l'IA Générative (Section 2.7) pour l'analyse sémantique. Le chapitre se conclut par une synthèse méthodologique (Section 2.8) justifiant nos choix technologiques.

2.1 Introduction : Le Défi de l'Automatisation en Cyber-Intelligence

Dans l'écosystème actuel de la cybersécurité, les honeypots (systèmes leurres) sont devenus des piliers de la stratégie de défense proactive. Contrairement aux systèmes de détection d'intrusion (IDS) qui cherchent des signatures connues dans le trafic de production, les honeypots capturent par définition une activité suspecte, éliminant théoriquement les faux positifs liés au trafic légitime. Cependant, la démocratisation des outils d'attaque automatisés (botnets, scripts kiddies) et l'émergence d'attaques sophistiquées (APT) génèrent un volume de données hétérogènes (logs systèmes, captures PCAP, binaires, séquences de commandes) qui dépasse les capacités d'analyse humaine.

L'automatisation de l'analyse via l'Intelligence Ar-

tificielle (IA) n'est plus une option, mais une nécessité opérationnelle pour transformer ces données brutes (*Raw Data*) en renseignement actionnable (*Threat Intelligence*). Ce chapitre propose une analyse critique et exhaustive de la littérature scientifique récente (2020-2025), en examinant successivement la typologie des données, les approches éprouvées par Machine Learning supervisé, et les perspectives offertes par le Deep Learning et l'IA générative.

2.2 Fondements Techniques : Architecture des Honeypots

Avant d'aborder l'analyse des données, il est nécessaire de définir l'architecture des systèmes de collecte. La littérature scientifique classifie les honeypots selon leur **niveau d'interaction**. Ce critère définit le degré de liberté laissé à l'attaquant et conditionne directement la nature des logs que notre système devra traiter.

2.2.1 Honeypots à Faible Interaction (Low-Interaction)

Ces systèmes simulent uniquement les services réseaux et les protocoles (TCP/IP, ICMP) sans offrir de véritable système d'exploitation à l'attaquant. Ils ne permettent pas d'exécution de code arbitraire.

— **Exemple technique :** *Dionaea*, utilisé dans l'étude de Firmansyah [FZ], est conçu pour capturer des malwares via des vulnérabilités spécifiques (SMB, HTTP, FTP).

— **Nature de la donnée :** Ils génèrent des données volumineuses mais très structurées : métadonnées de connexion (IP, Port, Timestamp) et empreintes de fichiers (Hashes MD5/SHA).

— **Pertinence pour le projet :** La structure rigide de ces logs les rend idéaux pour une classification rapide par des algorithmes de Machine Learning statistique comme le *Random Forest*.

2.2.2 Honeypots à Haute Interaction (High-Interaction)

À l'opposé, ces honeypots exposent un système d'exploitation réel ou une émulation avancée, permettant à l'attaquant d'interagir avec le système, de télécharger des outils et d'exécuter des commandes.

— **Exemple technique :** *Cowrie*, largement cité dans les travaux récents [LGV], émule un environnement UNIX complet et intercepte les sessions SSH et Telnet.

— **Nature de la donnée :** Ils capturent des données non-structurées et sémantiques : séquences de commandes Shell (ex : `rm -rf /`, `wget`), scripts offusqués et frappes clavier (keystrokes).

— **Pertinence pour le projet :** La complexité et la variabilité de ces commandes (souvent offusquées) justifient l'usage de techniques avancées comme les LLM (Large Language Models) pour en saisir le sens (Sémantique) là où le ML classique échoue.

Le choix de l'architecture relève d'un compromis. Les systèmes à faible interaction sont performants et peu risqués, mais "aveugles" aux détails de l'attaque. Les systèmes à haute interaction offrent une visibilité totale (Full Visibility), mais au prix d'une consommation de ressources élevée et d'un risque accru de compromission du système hôte.

2.3 Fondements Théoriques : Données et Benchmarks

L'efficacité d'un système de détection basé sur l'apprentissage automatique dépend intrinsèquement de la qualité et de la nature des données qui alimentent ses modèles.

2.3.1 Typologie des données de Honeypots

La littérature récente, notamment la revue systématique de **Kubba et al. (2025)**, classifie les données collectées en trois catégories distinctes, chacune offrant un niveau de granularité différent sur le comportement de l'attaquant [KNEAT] :

— **Logs Structurés et Métadonnées Réseau :** Il s'agit du niveau le plus basique de collecte, généralement issu de honeypots à faible interaction ou de pare-feux. Ces données incluent les adresses IP sources, les horodatages (*timestamps*), les ports ciblés et les protocoles de transport utilisés. Bien que pauvres en contexte sémantique, ces données tabulaires sont essentielles pour l'analyse statistique temporelle et la détection de tendances volumétriques.

— **Payloads et Artefacts Binaires :** Les honeypots à moyenne et haute interaction (comme *Dionaea*) acceptent les transferts de fichiers. Ces données comprennent les malwares téléchargés (exécutables Windows .exe, binaires Linux .elf). L'analyse de ces données nécessite des techniques d'extraction de caractéristiques (*feature engineering*) spécifiques pour convertir le code binaire en vecteurs exploitables.

— **Commandes Shell et Interactions Textuelles :** Cette catégorie représente la donnée la plus riche mais la plus complexe. Issue des honeypots simulant des services terminaux (SSH, Telnet), elle contient les séquences de commandes brutes saisies par l'attaquant (ex : `rm -rf /`, `wget`). Ces données non-structurées nécessitent souvent des techniques de Traitement du Langage Naturel (NLP).

2.3.2 Les Jeux de Données de Référence (Benchmarks)

Pour comparer objectivement les performances des algorithmes, la communauté scientifique s'appuie sur des jeux de données standardisés. L'étude extensive menée par **Ali et al. (2025)** recense les datasets dominants pour la détection d'anomalies dans les logs [ABB+] :

1. **HDFS (Hadoop Distributed File System) :** Généré sur une infrastructure Amazon EC2, ce dataset

contient plus de 11 millions de messages de logs. Étiqueté au niveau de la session, il constitue le standard *de facto* pour évaluer la détection d'anomalies séquentielles.

2. **BGL (BlueGene/L) :** Issu d'un supercalculateur (4,7 millions de logs), il présente des anomalies matérielles et logicielles étiquetées. Sa particularité réside dans un déséquilibre de classe modéré (environ 8% d'anomalies).

3. **Thunderbird & Spirit :** Ces datasets massifs (>200 millions de logs) sont principalement utilisés pour tester la scalabilité des algorithmes de Deep Learning face au "Big Data".

2.3.3 Les Données "In the Wild" (Collecte Terrain)

Les benchmarks académiques souffrant parfois d'obsolescence, des études récentes privilégient la collecte en temps réel pour garantir la pertinence face aux menaces actuelles :

— **Firmansyah & Zahra (2023)** utilisent un dataset hybride combinant *ClaMP* (malwares publics) et des données capturées par un honeypot *Dionaea* (2 169 échantillons récents) [FZ].

— **Lanka et al. (2024)** ont déployé des honeypots SSH sur le cloud AWS, capturant 47 764 commandes brutes provenant de 1 154 adresses IP attaquantes distinctes [LGV].

2.4 Métriques de Performance et Protocoles d'Évaluation

L'évaluation d'un système de détection d'intrusions (IDS) basé sur le Machine Learning ne peut se limiter à une simple mesure d'exactitude globale. En cybersécurité, les jeux de données sont intrinsèquement déséquilibrés (*imbalanced*) : le trafic légitime est majoritaire, tandis que les attaques représentent souvent moins de 1% des événements [ABB+].

Cette section définit les indicateurs mathématiques et les protocoles de validation retenus pour ce projet, en s'alignant sur les standards académiques identifiés dans l'état de l'art.

2.4.1 La Matrice de Confusion

La base de l'évaluation repose sur la matrice de confusion, qui croise les classes réelles avec les classes prédites par le modèle. Dans un contexte binaire (Attaque vs Normal), elle se définit ainsi :

	Prédit : Attaque	**Prédit : Normal**
Réel : Attaque	Vrai Positif (TP)	Faux Négatif (FN)
Réel : Normal	Faux Positif (FP)	Vrai Négatif (TN)

TABLE 2.1 – Matrice de Confusion pour la détection d'intrusions

— **Vrai Positif (TP) :** Une attaque correctement bloquée. C'est l'objectif principal.

— **Faux Négatif (FN) :** Une attaque non détectée. C'est le cas le plus critique en sécurité ("Missed Detection"), car le système est compromis.
— **Faux Positif (FP) :** Une fausse alerte. Un taux élevé entraîne une "fatigue de l'alerte" pour les analystes [noa].
— **Vrai Négatif (TN) :** Trafic normal correctement ignoré.

2.4.2 Métriques Dérivées

Pour dépasser le biais de l'exactitude (*Accuracy*), nous utiliserons les métriques suivantes, recommandées par Firmansyah (2023) et Ali et al. (2025) :

1. La Précision (Precision)

Elle mesure la qualité des alertes générées.

$$\text{Précision} = \frac{TP}{TP + FP} \tag{2.1}$$

Une précision de 100% signifie que chaque alerte est une véritable attaque. Une précision faible indique un système "bruyant".

2. Le Rappel (Recall / Sensitivity / TPR)

Il mesure la capacité du système à trouver toutes les attaques.

$$\text{Rappel} = \frac{TP}{TP + FN} \tag{2.2}$$

Dans notre contexte de honeypot, maximiser le Rappel est souvent prioritaire pour ne manquer aucune compromission potentielle.

3. Le F1-Score

C'est la moyenne harmonique de la Précision et du Rappel. C'est la métrique de référence pour comparer les modèles sur des datasets déséquilibrés, car elle pénalise les modèles qui favorisent une métrique au détriment de l'autre.

$$F1 = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}} = \frac{2TP}{2TP + FP + FN} \tag{2.3}$$

L'étude de **Firmansyah et Zahra (2023)** montre que le Random Forest atteint un F1-Score de **99,36%** sur la classification de malwares, ce qui servira de *baseline* pour notre projet [FZ].

4. Taux de Faux Positifs (FPR - False Positive Rate)

Il mesure la proportion de trafic légitime incorrectement classé comme attaque.

$$FPR = \frac{FP}{TN + FP} \tag{2.4}$$

L'objectif, selon **Alshahrani (2023)**, est de maintenir ce taux sous les 5% pour garantir l'opérabilité du système [noa].

2.4.3 Métriques de Performance Computationnelle

Outre la qualité de la détection, l'efficacité du modèle est un critère clé pour les environnements contraints (IoT) ou les flux massifs.

— **Temps d'Entraînement (T_{train}) :** Temps nécessaire pour construire le modèle. Comme le souligne **Ali et al. (2025)**, cette métrique disqualifie souvent les modèles de Deep Learning lourds (ex : NeuralLog nécessitant 15 heures) face au Random Forest (quelques minutes) pour des mises à jour fréquentes [ABB+].

— **Temps d'Inférence (T_{infer}) :** Temps nécessaire pour classifier une nouvelle instance. Pour une détection en temps réel, ce temps doit être inférieur à l'intervalle d'arrivée des paquets/logs. **Lee et al. (2021)** valident l'usage du Random Forest avec un temps d'inférence moyen de **0,1 ms** sur Raspberry Pi [LAJK].

2.4.4 Protocole de Validation

Pour garantir la robustesse des résultats et éviter le surapprentissage, il existe une méthode.

K-Fold Cross-Validation ($k = 10$) : Le jeu de données est divisé en k sous-ensembles. Le modèle est entraîné sur $k - 1$ parties et testé sur la partie restante. Ce processus est répété k fois. La performance finale est la moyenne des k itérations. Ce protocole, utilisé par la majorité des études de référence, assure que chaque échantillon a été utilisé à la fois pour l'entraînement et le test.

2.5 Approches par Machine Learning Supervisé : Analyse Approfondie

L'apprentissage supervisé constitue le socle de la détection d'intrusions moderne. Contrairement aux systèmes experts basés sur des règles statiques (ex : Snort), ces algorithmes apprennent une fonction de mappage $f : X \rightarrow Y$ à partir d'un jeu de données étiqueté $D = \{(x_i, y_i)\}_{i=1}^{N}$, où x_i représente le vecteur de caractéristiques d'un événement honeypot et y_i sa classe (Bénin, Malveillant, Type d'attaque).

Cette section analyse en détail les trois familles d'algorithmes dominantes dans la littérature récente (2020-2025) : les Arbres de Décision, les Méthodes d'Ensemble (Random Forest) et les Machines à Vecteurs de Support (SVM).

2.5.1 Prétraitement et Ingénierie des Caractéristiques (Feature Engineering)

L'efficacité des algorithmes de Machine Learning dépend moins du choix du modèle que de la qualité de la représentation des données. Les logs de honeypots étant hétérogènes, la littérature identifie deux stratégies majeures de transformation.

Extraction Structurelle (Cas des fichiers Binaires)

Dans le cadre de l'analyse de malwares capturés (fichiers PE - Portable Executable), **Firmansyah et Zahra (2023)** détaillent un processus d'extraction rigoureux pour éviter l'exécution en sandbox [FZ]. Ils définissent

un vecteur de caractéristiques X composé de 33 dimensions, extraites via la librairie `pefile`. Les attributs les plus discriminants identifiés sont :

- **Métriques de taille :** `SizeOfRawData` vs `VirtualSize`. Un écart significatif indique souvent la présence d'un *packer* (technique d'offuscation utilisée pour cacher le code malveillant).
- **Intégrité :** `CheckSum`. Les malwares générés automatiquement ont souvent des sommes de contrôle incohérentes avec le contenu du fichier.
- **Dépendances :** Les importations de DLL (`Import Address Table`) révèlent les fonctionnalités du programme (ex : appels réseaux, chiffrement).

Vectorisation des Logs Textuels (NLP)

Pour les logs textuels (commandes Shell, requêtes HTTP), une simple conversion numérique ne suffit pas. **Ali et al. (2025)** comparent plusieurs techniques d'encodage [ABB+] :

- **TF-IDF (Term Frequency-Inverse Document Frequency) :** Pondère les mots rares (ex : "wget", "curl") plus lourdement que les mots fréquents. Utilisé par les approches classiques (SVM).
- **Semantic Embeddings (FastText/BERT) :** Transforme chaque log en un vecteur dense dans un espace continu. Bien que plus précis, Ali et al. démontrent que pour des logs très structurés, le gain de performance par rapport à un encodage fréquentiel simple est souvent marginal pour les algorithmes d'arbres.

2.5.2 Les Arbres de Décision (Decision Trees - DT)

Les arbres de décision (C4.5, CART) partitionnent l'espace des données en rectangles par des divisions récursives basées sur la maximisation du gain d'information (ou la minimisation de l'impureté de Gini).

Application à la Prédiction d'Attaques : Alshahrani (2023) utilise l'algorithme J48 (implémentation Java de C4.5) pour prédire les attaques sur un réseau de honeypots [**?**].

- **Performance :** Sur un jeu de données unique, le DT atteint **95,52%** de précision.
- **Avantage :** L'explicabilité. Le modèle produit des règles claires (ex : *SI port=22 ET protocol=SSH ET duration<1s ALORS Brute-Force*), cruciales pour les analystes SOC.
- **Limitation Critique (Scalabilité) :** L'auteur observe une chute de précision à **90,87%** lorsque le modèle est testé sur une architecture multi-honeypots distribuée. Cela illustre la tendance du DT au sur-apprentissage (*overfitting*) : il apprend "par cœur" les spécificités d'un capteur mais échoue à généraliser sur des données hétérogènes [noa].

2.5.3 Random Forest : L'État de l'Art Industriel

Pour pallier la variance élevée des arbres de décision, l'algorithme **Random Forest (RF)** utilise la technique du *Bagging* (Bootstrap Aggregating). Il entraîne N arbres sur des sous-ensembles aléatoires de données et agrège leurs votes (Figure 2.1).

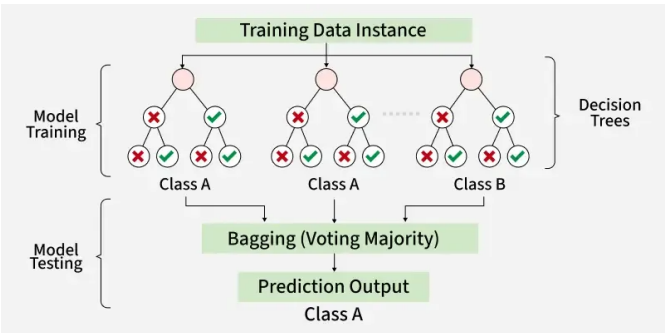


FIGURE 2.1 – Ilustration du fonctionnement de l'algorithme Random Forest *(source : geeksforgeeks.org)*

Validation Empirique Multi-Domaines : Le Random Forest s'impose dans la littérature 2020-2025 comme l'algorithme "roi" pour les honeypots, validé par trois études majeures aux contextes différents.

A. Pour la classification de Malwares (Firmansyah, 2023)

Sur le dataset hybride (ClaMP + Dionaea), Firmansyah rapporte les résultats suivants [FZ] : **Analyse :** Le RF

Algorithme	Précision	Rappel	F1-Score
Decision Tree	97.9%	98.6%	98.2%
SVM	87.1%	95.5%	87.9%
Random Forest	**99.2%**	**99.6%**	**99.3%**

TABLE 2.2 – Comparaison des performances sur la classification de malwares (Firmansyah, 2023)

élimine presque totalement les faux négatifs (Rappel de 99,6%), ce qui est la priorité absolue en cybersécurité (ne pas rater une infection).

B. Pour la rapidité dans l'IoT (Lee et al., 2021)

Dans les environnements industriels (Smart Factories), la latence est critique. Lee et al. ont implémenté un RF sur des Raspberry Pi [LAJK].

- **Précision :** 96,66% sur la détection de botnets Mirai.
- **Temps d'inférence : 0,1 ms** par échantillon.

Ce résultat prouve que le RF est compatible avec les contraintes temps-réel des réseaux haut débit, là où des réseaux de neurones profonds (DNN) introduiraient une latence de plusieurs millisecondes.

C. Comparaison face au Deep Learning (Ali et al., 2025)

L'étude de Ali et al. est décisive pour justifier l'usage du RF face à la "hype" du Deep Learning. Sur le dataset de référence HDFS (11 millions de logs), ils mesurent [ABB+] :

— **Performance :** RF (F1 : 99,8%) ≈ NeuralLog (F1 : 99,8%).

— **Coût d'entraînement :** RF (96 secondes) ≪ NeuralLog (57 497 secondes).

Conclusion : Pour des données tabulaires issues de logs, le Random Forest offre la même performance que les Transformers, pour un coût énergétique 600 fois inférieur.

2.5.4 Support Vector Machines (SVM) et Approches Probabilistes

Les SVM cherchent à trouver l'hyperplan séparateur qui maximise la marge entre les classes. **Rivas et al. (2019)** ont utilisé des SVM pour classifier des attaques DDoS sur des honeypots Cloud [RDO+]. Cependant, les études récentes montrent les limites de cette approche :

— **Sensibilité au bruit :** Dans l'étude de Firmansyah, le SVM chute à 87% de précision [FZ]. Contrairement au RF qui isole le bruit dans des branches profondes, le SVM tente de trouver une frontière géométrique globale, ce qui est difficile quand les données des attaquants sont très hétérogènes ou offusquées.

— **Complexité quadratique :** Le temps d'entraînement du SVM augmente quadratiquement avec le nombre d'échantillons ($O(n^2)$), le rendant inadapté aux flux massifs de "Big Data" générés par les honeypots modernes, contrairement au RF ($O(n \log n)$).

Taşçi et al. (2021) explorent également le **Naive Bayes** pour la détection d'attaques de mots de passe [TGB+]. Bien que très rapide (linéaire), cet algorithme suppose l'indépendance des caractéristiques, une hypothèse fausse en sécurité (ex : l'échec d'un login est corrélé à la tentative suivante dans une attaque brute-force), ce qui plafonne ses performances autour de 92-94%.

2.5.5 Synthèse des Algorithmes Supervisés

Le Tableau 2.3 synthétise les forces et faiblesses des algorithmes analysés dans le contexte spécifique des honeypots.

Algorithme	Précision Moyenne	Robustesse au Bruit	Coût Calcul	Verdi
Decision Tree	Haute (95-98%)	Faible (Overfitting)	Très Faible	Bon p
SVM	Moyenne (83-90%)	Faible	Élevé ($O(n^2)$)	Dépa
Naive Bayes	Moyenne (90-94%)	Moyenne	Très Faible	Utile
Random Forest	**Très Haute (>99%)**	**Forte**	**Faible**	**L'éta**

TABLE 2.3 – Comparaison critique des algorithmes de ML supervisé pour les Honeypots

2.6 Le Deep Learning (Gestion de la Séquentialité)

Le Deep Learning (DL) est souvent présenté comme la solution aux limites du ML classique, notamment pour gérer la séquentialité et les données non structurées. Cependant, la littérature récente invite à la nuance.

2.6.1 La Promesse : Modélisation Temporelle

Abdallah et al. (2024) soulignent que les attaques modernes (APT, exfiltration lente) ne sont pas des événements atomiques mais des séquences. Ils préconisent l'usage de réseaux **LSTM** (Long Short-Term Memory) pour leur capacité à retenir le contexte sur de longues périodes, et de **CNN** (Convolutional Neural Networks) pour l'analyse spatiale du trafic réseau [AAA+].

2.6.2 La Réalité Empirique : L'Étude Critique de Ali et al. (2025)

L'étude massive de **Ali et al. (2025)** constitue un "Reality Check" indispensable pour tout projet R&D [ABB+]. En comparant 9 techniques sur 7 datasets, ils démontrent que :

1. **Performance Équivalente :** Sur des logs structurés (HDFS), le Random Forest atteint un F1-score de 99,8%, égalant les modèles DL complexes comme *NeuralLog* ou *LogRobust*.
2. **Coût Disproportionné :** Le temps d'entraînement sur GPU A100 varie de 96 secondes pour le Random Forest à plus de 57 000 secondes pour des modèles basés sur les Transformers.
3. **Sensibilité :** Les modèles DL sont extrêmement sensibles aux hyperparamètres, rendant leur déploiement industriel complexe.

Cette étude justifie de ne pas recourir au Deep Learning pour la simple classification de logs, mais de le réserver à des tâches où la modélisation séquentielle est critique.

2.6.3 L'Innovation : Honeypots Adaptatifs (Defense Active)

La véritable valeur ajoutée du Deep Learning réside dans l'adaptabilité. **Abdul Kareem et al. (2024)** proposent un "Honeypot Adaptatif" piloté par l'Apprentissage par Renforcement (Reinforcement Learning - RL) [AKSM]. L'agent RL modifie dynamiquement la configuration du leurre pour maximiser le temps de connexion de l'attaquant. Les résultats montrent une augmentation de 40% de la durée d'interaction et de 50% de la collecte de données. Cependant, cette intelligence a un coût : une surcharge de 30% des ressources système (CPU/RAM).

2.7 L'Ère Sémantique (IA Générative et LLM)

Depuis 2024, l'intégration des Grands Modèles de Langage (LLM) marque une rupture pour l'analyse des données non structurées, notamment les commandes

Shell, où les approches statistiques échouent face à l'obfuscation. Par exemple, la commande Shell suivante `rm -rf /` peut être obfusquée et rendue difficilement identifiable ainsi : `a="rm"; b="-rf"; $a $b /`.

2.7.1 Analyse Sémantique par RAG

L'étude de **Lanka, Gupta et Varol (2024)** est pionnière dans l'application du *Retrieval-Augmented Generation* (RAG) aux logs de honeypots SSH [LGV]. Leur pipeline se décompose en trois étapes :

1. **Normalisation :** Utilisation de *Bashlex* pour abstraire les variables (IP, fichiers) des commandes.
2. **Vectorisation (Embeddings) :** Transformation des commandes en vecteurs mathématiques via un modèle spécialisé (*Salesforce/SFR-Embedding-Mistral*).
3. **Interprétation :** Recherche de similarité vectorielle pour identifier des variantes d'attaques connues, suivie d'une qualification par un LLM (GPT-4) pour expliquer l'intention malveillante.

Cette approche permet non seulement de détecter, mais d'**expliquer** l'attaque, comblant le manque d'interprétabilité des réseaux de neurones classiques ("Black Box").

2.8 Synthèse et Conclusion Méthodologique

L'analyse de l'état de l'art révèle une dichotomie claire dans les approches technologiques, résumée dans le Tableau 2.4. De plus les différents benchmarks de l'étude d'Ali et al[ABB+] permet d'avoir une vue d'ensemble des performances de différentes méthodes d'analyse de logs (Figures 2.2 & 2.3).

Technologie	Cas d'Usage Optimal	Avantag
Random Forest (ML)	Classification de malwares, Détection d'intrusions IoT	Rapidité extrême, Faible coû
LSTM / CNN (DL)	Analyse de séquences longues, Trafic réseau brut	Gestion de la temporalité et des
LLM / RAG (GenAI)	Analyse de commandes Shell, Explicabilité	Compréhension sémantique, C

TABLE 2.4 – Synthèse comparative des approches d'IA pour les Honeypots

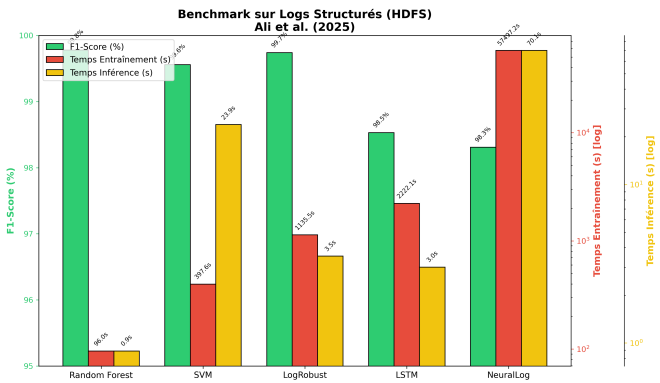


FIGURE 2.2 – Benchmark comparatif des performances (F1-Score) et des temps d'entraînement sur le dataset HDFS (logs structurés)[ABB+].

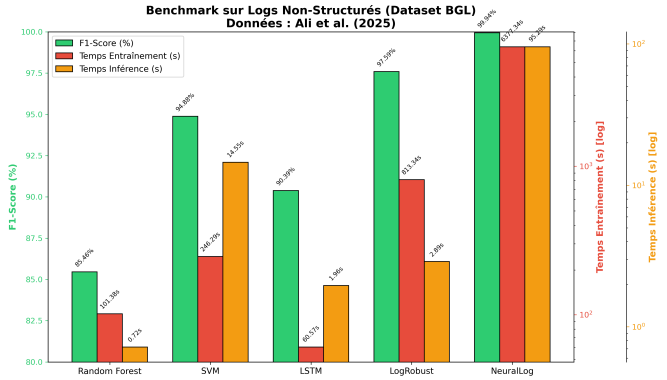


FIGURE 2.3 – Benchmark comparatif des performances (F1-Score) et des temps d'entraînement sur le dataset BGL (logs non-structurés)[ABB+].

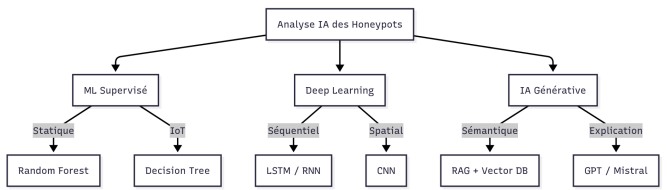


FIGURE 2.4 – Les différentes catégories d'IA utilisées dans le cadre des logs de Honeypots

Au regard de ces conclusions, ce projet adoptera une stratégie hybride. Le socle de détection sera basé sur le **Random Forest**, validé par Firmansyah (2023) et Ali (2025) comme la référence pour la classification rapide. En complément, une exploration des techniques d'**Embeddings et LLM** (inspirée de Lanka, 2024) sera menée pour traiter les commandes inconnues, apportant ainsi la dimension innovante attendue d'un projet R&D.

3

Proposition d'Architecture et Méthodologie

Ce chapitre formalise la solution technique envisagée. Nous y détaillons l'architecture d'une pipeline hybride, la stratégie de données (du développement à la mise à l'échelle) et la méthodologie d'extraction de caractéristiques.

Sur la base de l'analyse de l'état de l'art (Chapitre 2), et notamment des conclusions de Ali et al. (2025) sur le coût du Deep Learning, nous proposons une architecture de détection en "entonnoir" (*Funnel Architecture*). L'objectif est de concilier l'efficience industrielle du Machine Learning classique pour le filtrage massif et la puissance cognitive de l'IA générative pour l'analyse fine des menaces complexes (si le temps nous permet de bien l'intégrer).

Nous détaillerons dans un premier temps la stratégie d'acquisition et de gestion des données (Section 3.1), incluant la sélection des datasets Cowrie et la méthodologie d'étiquetage. La Section 3.2 présentera les travaux d'ingénierie des caractéristiques (Feature Engineering) nécessaires pour transformer des logs bruts en vecteurs exploitables. Le cœur de notre contribution, l'architecture du pipeline de traitement (Section 3.3), expliquera l'articulation entre le Random Forest et le module LLM. Enfin, nous définirons la stratégie de validation expérimentale (Section 3.4) qui sera mise en œuvre pour évaluer les performances du système.

3.1 Stratégie d'Acquisition et de Gestion des Données

L'entraînement de modèles de Machine Learning nécessite un volume conséquent de données qualifiées. Pour garantir la robustesse de notre approche, nous avons défini une stratégie de données à deux niveaux.

3.1.1 Sélection des Jeux de Données (Datasets)

Nous exploiterons deux sources distinctes de logs issus de honeypots **Cowrie** (standard industriel pour l'émulation SSH/Telnet), répondant à deux besoins différents du

projet.

1. **Jeu de Données de Développement : "Cowrie-Honeypot" (Kaggle)**
   - **Source :** Logs d'un honeypot Cowrie exposé sur le cloud Azure pendant deux semaines (Auteur : xmlyna).
   - **Volumétrie :** Environ 16 000 sessions et 40 000 événements.
   - **Usage :** Ce dataset, de taille raisonnable, servira au **prototypage rapide** du pipeline, à la mise au point des scripts de feature engineering et à l'entraînement initial des modèles Random Forest. Sa structure propre facilite les itérations de développement.

2. **Jeu de Données de Scalabilité : "CyberLab Honeynet" (Zenodo)**
   - **Source :** Archive massive du *CyberLab*, couvrant une année complète d'activité de honeypots (Mai 2019 - Février 2020).
   - **Volumétrie :** Plus de 10 Go de logs compressés, représentant des millions d'interactions.
   - **Usage :** Ce dataset sera utilisé dans une seconde phase pour valider la **robustesse et la scalabilité** de notre modèle. Il permettra de vérifier si le Random Forest maintient ses performances (latence < 1ms) face à un flux massif, simulant un environnement de production réel.

3.1.2 Structure de la Donnée et Parsing

Le format d'entrée est le standard JSON de Cowrie. Chaque interaction est un événement discret. L'événement critique pour notre analyse est `cowrie.command.input`, qui contient la commande brute exécutée par l'attaquant.

```
{
  "eventid": "cowrie.command.input",
  "timestamp": "2025-10-24T03:15:22.123Z",
  "session": "a1b2c3d4",
  "src_ip": "192.168.1.55",
  "input": "wget http://malware.site/bot.sh -O /tmp/.x; chmo
  "message": "CMD: wget http://malware.site/bot.sh ..."
}
```

3.1.3 Stratégie d'Étiquetage (Ground Truth)

Les logs bruts ne disposant pas d'étiquette "Type d'attaque", nous appliquerons une méthodologie de **Weak Supervision** (Supervision Faible) pour constituer notre vérité terrain (*Ground Truth*). Nous développerons un module d'étiquetage heuristique basé sur des règles expertes pour annoter automatiquement le jeu d'entraînement :

- **Classe Download :** Déclenchée par `wget`, `curl`, `scp`, `ftp`, `tftp`.
- **Classe Execution :** Déclenchée par `chmod +x`, `./`, `sh`, `bash`, `python`.
- **Classe Reconnaissance :** Déclenchée par `uname`, `whoami`, `cat /proc/cpuinfo`, `id`.
- **Classe Destruction :** Déclenchée par `rm`, `mv`, `dd`, `> /dev/null`.

Le modèle Random Forest apprendra ensuite à généraliser ces classes au-delà des simples mots-clés, en identifiant des corrélations statistiques (longueur, entropie, contexte).

3.2 Ingénierie des Caractéristiques (Feature Engineering)

Pour permettre au modèle **Random Forest** de traiter ces données textuelles, nous devons transformer les commandes brutes en vecteurs numériques significatifs. Nous avons défini pour le moment quelques caractéristiques (*features*) à extraire pour créer nos vecteurs.

3.2.1 Caractéristiques Lexicales (Analyse de la forme)

Ces métriques visent à détecter la complexité et l'offuscation sans analyser le sens profond.

- **Longueur de la commande (int) :** Les payloads malveillants (téléchargement complexe, encodage) sont souvent significativement plus longs que les commandes d'administration standard.
- **Entropie de Shannon (float) :** Calculée sur la chaîne de caractères. Une entropie élevée (> 4.5) indique une distribution de caractères aléatoire ou encodée (Base64), typique des tentatives d'évasion pour cacher un payload.
- **Ratio de caractères spéciaux (float) :** Proportion de `;`, `|`, `$` par rapport aux lettres. Un ratio élevé signale souvent du "Command Chaining" ou des scripts complexes.
- **Nombre d'arguments (int) :** Indicateur de complexité de la commande.

3.2.2 Caractéristiques Sémantiques (Mots-clés)

Nous utiliserons une approche "Bag-of-Words" ciblée pour détecter l'intention via des indicateurs binaires (0 ou 1). Voici quelques exemples (pas forcément les plus pertinants pour le moment) :

- **Flags d'Action :** Présence de verbes d'action critiques (Download, Execution, Destruction).
- **Flag Network :** Présence d'outils réseaux suspects (`nc`, `netcat`, `ping`, `telnet`).
- **Flag Privilege :** Présence de tentatives d'élévation de privilèges (`sudo`, `su`, `root`).

3.2.3 Caractéristiques Contextuelles (Session)

Ces features agrègent l'historique de la session pour donner du contexte à la commande isolée.

- **Heure de l'attaque (catégorique) :** Heures ouvrées vs Nuit/Week-end.
- **Vitesse de frappe (float) :** Temps écoulé entre la connexion et la première commande. Les bots exécutent souvent leurs scripts dans les premières millisecondes (< 100ms), contrairement aux humains.
- **Position dans la session (int) :** Numéro de séquence de la commande (les commandes de reconnaissance arrivent souvent au début, l'exfiltration à la fin).

3.3 Architecture du Pipeline de Traitement

Notre solution technique repose sur une architecture modulaire (voir Figure 3.1), conçue pour optimiser le ratio coût/performance identifié comme critique par Ali et al. (2025).

3.3.1 Branche A : Classification Rapide (Random Forest)

Cette branche traite 100% du flux entrant.

— **Objectif :** Filtrer les attaques massives et connues (Bots, Brute Force).

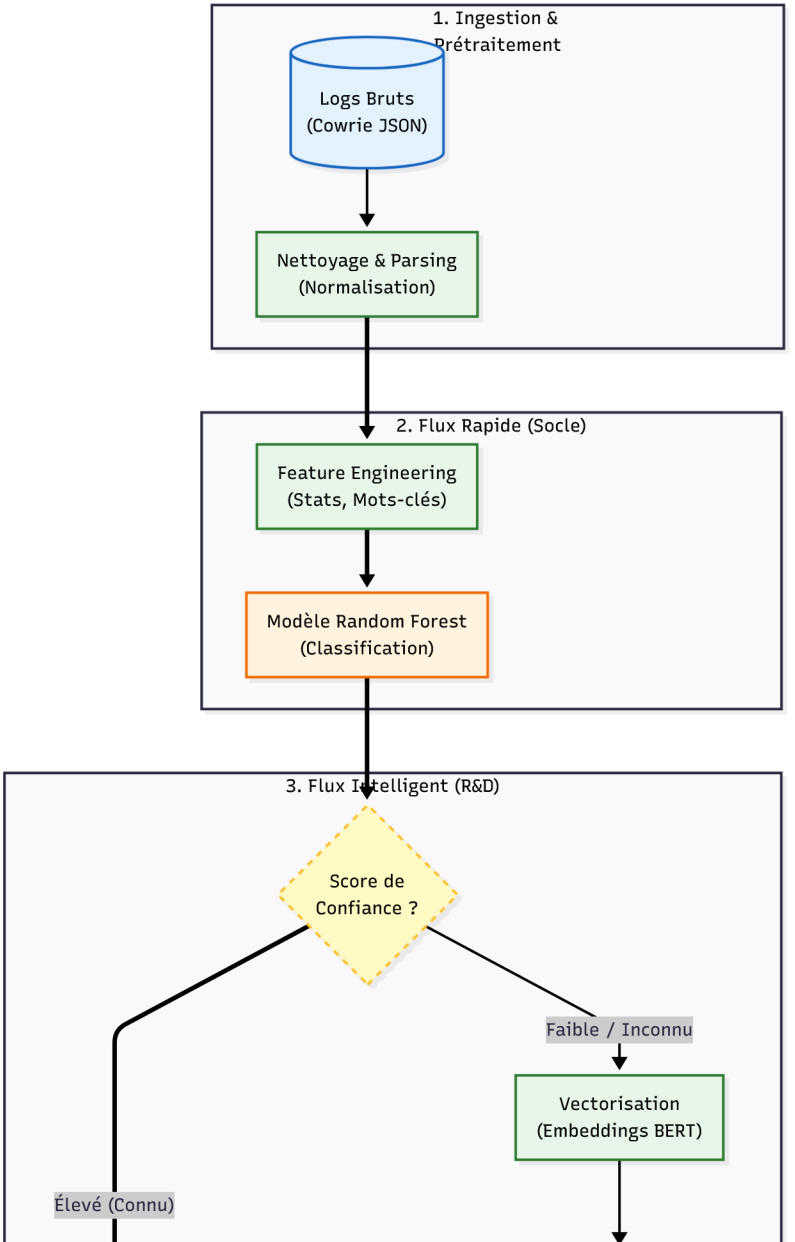
— **Algorithme :** Random Forest (Scikit-learn).

— **Justification :** Validé par Firmansyah (2023) pour sa précision (99.2%) et par Ali (2025) pour son coût d'entraînement 600 fois inférieur au Deep Learning.

— **Critère de décision :** Le modèle fournit une classe prédite et un **score de probabilité** (via `predict_proba`). Si ce score dépasse un seuil de confiance (ex : 0.8), la décision est acceptée immédiatement.

3.3.2 Branche B : Exploration Sémantique (LLM)

Activée uniquement si le Random Forest est "incertain" (Score de confiance < Seuil) ou classifie l'événement comme "Inconnu".



— **Technologie :** Vectorisation (Embeddings) + LLM (via modèle local).
— **Objectif :** Qualifier les attaques "Zero-Day" ou fortement offusquées. Le LLM génère une explication textuelle de l'attaque.

3.4 Stratégie de Validation Expérimentale

Pour évaluer la pertinence de notre architecture hybride, nous suivrons un protocole rigoureux.

3.4.1 Protocole d'Entraînement et de Test

— **Séparation des données :** Le dataset Kaggle sera divisé en deux sous-ensembles : 80% pour l'entraînement (*Training Set*) et 20% pour le test (*Testing Set*).
— **Validation Croisée :** Pour éviter le biais de sélection, nous appliquerons une **K-Fold Cross-Validation** ($k = 10$) sur les données d'entraînement.

3.4.2 Critères de Succès

Le succès de notre solution sera mesuré selon trois axes :

1. **Performance de Détection :** Atteindre un F1-Score $> 95\%$ sur la branche Random Forest.
2. **Taux de Faux Positifs (FPR) :** Maintenir le FPR inférieur à 1% pour garantir l'opérabilité en SOC.
3. **Performance Temps-Réel :** Rester sous la barre des **1 ms** de latence d'inférence pour le Random Forest (validé par Lee et al., 2021) et quantifier le surcoût du module LLM.

3.5 Conclusion

Cette architecture répond aux exigences de l'analyse de honeypots modernes : elle assure la rapidité nécessaire pour traiter des millions de logs (dataset Zenodo) tout en offrant une capacité d'investigation cognitive pour les menaces émergentes.

Bibliographie

[AAA+] Amira Abdallah, Aysha Alkaabi, Ghaya Alameri, Saida Rafique, Nura Musa, and Thangavel Murugan. Cloud network anomaly detection using machine and deep learning techniques - recent research advancements. PP :1–1. 21

[ABB+] Shan Ali, Chaima Boufaied, Domenico Bianculli, Paula Branco, and Lionel Briand. A comprehensive study of machine learning techniques for log-based anomaly detection. 30(5) :129. 14, 15, 17, 18, 20, 21, 22, 23, 31

[AKSM] Shaik Abdul Kareem, Ram Chandra Sachan, and Rajesh Kumar Malviya. AI-driven adaptive honeypots for dynamic cyber threats. 21

[ASB+] Ahmed Abdou, Ryan Sheatsley, Yohan Beugin, Tyler Shipp, and Patrick McDaniel. HoneyModels : Machine learning honeypots. In *MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM)*, pages 886–891. ISSN : 2155-7586.

[FZ] Diandra Firmansyah and Amalia Zahra. Honeypot-based thread detection using machine learning techniques. 71 :243–252. 13, 15, 16, 17, 19, 20

[KNEAT] Ahmed Kubba, Qassim Nasir, Omnia Elmutasim, and Manar Abu Talib. A systematic review of honeypot data collection, threat intelligence platforms, and ai/ml techniques. 14

[LAJK] Seungjin Lee, Azween Abdullah, Nz Jhanjhi, and Sh Kok. Classification of botnet attacks in IoT smart factory using honeypot combined with machine learning. 7 :e350. 17, 19, 20

[LGV] Phani Lanka, Khushi Gupta, and Cihan Varol. Intelligent threat detection—AI-driven analysis of honeypot data to counter cyber threats. 13(13) :2465. Publisher : Multidisciplinary Digital Publishing Institute. 13, 15, 22

[MBK+] Vishal Mehta, Pushpendra Bahadur, Manik Kapoor, Preeti Singh, and Subhadra Rajpoot. Threat prediction using honeypot and machine learning. In *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, pages 278–282.

[MR] Iik Muhamad Malik Matin and Budi Rahardjo. The use of honeypot in machine learning based on malware detection : A review. In *2020 8th International Conference on Cyber and IT Service Management (CITSM)*, pages 1–6.

[noa] Predication attacks based on intelligent honeypot technique. 12(3) :1631–1635. 16, 18, 20

[RDO+] Pablo Rivas, Casimer DeCusatis, Matthew Oakley, Alex Antaki, Nicholas Blaskey, Steven LaFalce, and Stephen Stone. Machine learning for DDoS attack classification using hive plots. In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 0401–0407. 20

[SPZK] Ali Hussain Shah, Daem Pasha, Esmaeil Habib Zadeh, and Savas Konur. Automated log analysis and anomaly detection using machine learning. In *Fuzzy Systems and Data Mining VIII*, pages 137–147. IOS Press.

[TGB+] Hatice Taşçi, Serkan Gönen, Mehmet Ali Barişkan, Gökçe Karacayilmaz, Birkan Alhan, and Ercan Nurcan Yilmaz. Password attack analysis over honeypot using machine learning password attack analysis. 13(2) :388–402. 20

Table des figures

Liste des tableaux

List of Algorithms

Fiche de lecture bibliographique

Référence complète	Firmansyah, Diandra & Zahra, Amalia. (2023). Honeypot-Based Thread Detection using Machine Learning Techniques. International Journal of Engineering Trends and Technology. 71. 243-252. 10.14445/22315381/IJETT-V71I8P221.
Type de document	Article de revue
Auteurs / Année	Amalia Zahra & Diandra Firmansyah
Titre	Honeypot-Based Thread Detection using Machine Learning Techniques
Source	10.14445/22315381/IJETT-V71I8P221
Objectif de l'étude	Proposer une approche de détection automatique de menaces à partir des données collectées par un honeypot, en appliquant différents algorithmes de Machine Learning et en comparant leurs performances afin d'améliorer la précision de classification des fichiers malveillants.

Contexte et motivation	L'analyse manuelle des journaux générés par les honeypots est chronophage et difficile à maintenir face à l'évolution rapide des attaques. Les auteurs cherchent à automatiser cette analyse et à améliorer la détection de malwares en combinant des données issues d'un honeypot Dionaea avec un jeu de données public (ClaMP).
Données utilisées	- Jeu de données ClaMP : 7 574 malwares et 4 989 fichiers bénins. - Données capturées via un honeypot Dionaea (2 169 échantillons collectés en 2023). Les deux sources combinées permettent de créer un ensemble représentatif de fichiers exécutables (PE).
Méthodologie / Modèles	Extraction de 33 caractéristiques statiques issues des en-têtes PE (Portable Executable). Évaluation de trois modèles : Decision Tree, Random Forest, et Support Vector Machine (SVM). Séparation des données 90/10 (entraînement/test) et mesure de performance via précision, rappel et F1-score.

Résultats principaux	- Random Forest : précision de 99,4 % et F1-score de 99,4 %. - Decision Tree : 99,2 %. - SVM : 87 %. Le modèle Random Forest est jugé le plus performant et robuste après sélection des 5 attributs les plus pertinents.
Forces de l'étude	- Données réelles issues d'un honeypot et validation croisée avec un dataset public. - Méthodologie claire et reproductible. - Excellentes performances obtenues sans recours au deep learning.
Limites / Biais	- Analyse statique uniquement (pas d'observation du comportement dynamique). - Données limitées à un seul honeypot et une seule région géographique (Indonésie). - Peu de modèles explorés (pas d'algorithmes profonds).
Lien avec mon projet	L'étude illustre concrètement comment exploiter les logs de honeypots à l'aide du Machine Learning. Elle sert de base méthodologique pour concevoir un pipeline d'analyse automatisée de logs dans mon PRED.

Synthèse personnelle	Cette étude montre qu'une approche de classification supervisée appliquée aux données de honeypots peut atteindre une très forte précision. Elle met en évidence la pertinence du Random Forest pour la détection de malwares, tout en soulignant la nécessité de tests à plus grande échelle et de méthodes plus adaptatives (Deep Learning).

Fiche de lecture bibliographique

Référence complète	Abdul Kareem, Shaik and Sachan, Ram Chandra and Malviya, Rajesh Kumar, AI-Driven Adaptive Honeypots for Dynamic Cyber Threats (September 17, 2024). Available at SSRN : https ://ssrn.com/abstract=4966935 or http ://dx.doi.org/10.2139/ssrn.4966935
Auteurs / Année	Shaik Abdul Kareem & Ram Chandra Sachan & Rajesh Kumar Malviya
Titre	AI-Driven Adaptive Honeypots for Dynamic Cyber Threats
Source	10.2139/ssrn.4966935

Objectif de l'étude	Proposer et évaluer un honeypot adaptatif piloté par l'intelligence artificielle, capable d'ajuster dynamiquement son comportement face aux cyberattaques en cours. L'objectif est d'améliorer la détection des menaces avancées (APT, zero-day) et d'augmenter la durée et la richesse des interactions avec les attaquants.
Contexte et motivation	Les honeypots classiques, statiques et facilement identifiables, deviennent inefficaces face à des attaquants de plus en plus sophistiqués. L'étude vise à combler cette limite en introduisant des mécanismes d'adaptation automatique grâce à des modèles d'apprentissage profond et par renforcement, afin de rendre le leurre plus crédible et plus réactif.
Données utilisées	Données générées par des attaques simulées sur Google Cloud Platform : environ 100 Go de trafic et 50 000 scénarios d'attaque labellisés (DDoS, SQL injection, brute force, ransomware). Les informations collectées incluent le trafic réseau, les charges utiles, les adresses IP et les commandes exécutées.
Méthodologie / Modèles	Architecture en trois couches : 1. Collecte des logs et du trafic. 2. Moteur d'adaptation utilisant des réseaux de neurones convolutifs (CNN) pour classifier les attaques et un apprentissage par renforcement pour modifier dynamiquement la configuration du honeypot. 3. Environnement d'interaction simulant divers services (HTTP, SQL, SSH). Comparaison d'un honeypot statique et du honeypot adaptatif IA.
Résultats principaux	L'honeypot adaptatif améliore la durée moyenne d'interaction de 40 %, collecte 50 % de données supplémentaires et augmente le taux de détection à 90 % contre 65 % pour le modèle statique. La précision d'adaptation atteint 85 %, au prix d'une hausse de 30 % de la consommation de ressources.

Forces de l'étude	- Première conception détaillée d'un honeypot réellement dynamique. - Gains mesurables et reproductibles. - Combinaison innovante de Deep Learning et d'apprentissage par renforcement. - Architecture modulaire facilement intégrable à d'autres outils de cybersécurité.
Limites / Biais	- Coût computationnel élevé (CPU et mémoire). - Données générées en environnement simulé : pas encore testées sur des réseaux réels. - Nécessité d'un entraînement long et dépendance à la qualité de labellisation des attaques.
Lien avec mon projet	Cette étude illustre l'évolution vers une automatisation complète de l'analyse et de la détection des menaces. Elle renforce la pertinence d'exploiter le Machine Learning pour analyser les logs de honeypots et justifie l'exploration future de modèles adaptatifs ou séquentiels (LSTM, RL) dans le cadre de mon PRED.
Synthèse personnelle	Ce travail montre que l'intelligence artificielle peut transformer les honeypots en systèmes proactifs et auto-évolutifs. Il ouvre la voie à une cybersécurité prédictive basée sur l'adaptation continue des défenses. Pour mon projet, il constitue une référence inspirante sur la conception de pipelines automatisés et adaptatifs d'analyse de logs.

Fiche de lecture bibliographique

Référence complète	A. Abdou, R. Sheatsley, Y. Beugin, T. Shipp and P. McDaniel, "Honey-Models : Machine Learning Honeypots," MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM), San Diego, CA, USA, 2021, pp. 886-891, doi : 10.1109/MILCOM52596.2021.9652947.

Auteurs / Année	Ahmed Abdou & Ryan Sheatsley & Yohan Beugin & Tyler Shipp† & Patrick McDaniel
Titre	HoneyModels : Machine Learning Honeypots
Source	10.1109/MILCOM52596.2021.9652947
Objectif de l'étude	Proposer un nouveau mécanisme de défense pour les modèles de Machine Learning appelé *HoneyModels*, inspiré du concept de honeypot. L'objectif est de détecter les attaques adversariales dirigées contre des réseaux de neurones en intégrant un watermark (marque secrète) dans le modèle afin de repérer les comportements suspects.
Contexte et motivation	Les modèles de Machine Learning, notamment les réseaux de neurones profonds, sont vulnérables aux attaques adversariales. Ces attaques consistent à apporter de légères perturbations à un échantillon pour tromper le modèle. Les défenses actuelles (distillation, adversarial training, etc.) sont coûteuses, peu généralisables et faciles à contourner. Les auteurs s'inspirent des honeypots pour créer un modèle « piégé » qui attire et révèle les attaquants sans altérer la performance du modèle légitime.
Données utilisées	Deux jeux de données d'images standards utilisés pour l'évaluation : - **MNIST** (chiffres manuscrits) - **CIFAR-10** (images multi-classes). Les attaques adversariales évaluées incluent : Carlini-Wagner (CW), Projected Gradient Descent (PGD), et Jacobian Saliency Map Attack (JSMA).

Méthodologie / Modèles	- Création d'un « watermark » secret appliqué à certaines caractéristiques du modèle. - Entraînement d'un réseau neuronal altéré via un processus de *self-poisoning* : une fraction des données est modifiée (poisonnée) pour forcer le modèle à intégrer le watermark dans sa structure interne. - À l'inférence, un détecteur (logistic regression) compare le watermark attendu à celui observé pour identifier les échantillons adversariaux. - Trois critères d'évaluation : **Viabilité** (maintien des performances), **Détection** (taux de détection élevé) et **Indistinguishabilité** (impossibilité pour un attaquant de repérer la marque).
Résultats principaux	- Détection jusqu'à 69,5 % des attaques adversariales, avec un taux de faux positifs de 14,1 %. - Maintien d'une précision élevée du modèle d'origine (perte d'environ 12 % seulement pour 90 % de données empoisonnées). - Les adversarial samples générés à partir de HoneyModels sont **indiscernables** de ceux produits par des modèles non protégés, empêchant l'attaquant d'identifier la présence du watermark.
Forces de l'étude	- Approche innovante qui détourne les techniques d'attaque pour améliorer la sécurité. - Faible coût computationnel comparé aux défenses classiques. - Compatible avec différents jeux de données et architectures. - Bon équilibre entre robustesse et précision.
Limites / Biais	- Méthode testée uniquement sur des modèles d'image simples (pas de NLP ou de contexte industriel). - Le taux de détection (70 %) reste insuffisant pour un déploiement opérationnel. - L'approche dépend d'un hyperparamétrage fin (taille et amplitude du watermark, taux de poison).

Lien avec mon projet	Cette étude démontre qu'un modèle de Machine Learning peut être « piégé » pour repérer des interactions malveillantes, exactement comme un honeypot réseau. Elle illustre la fusion entre le concept de honeypot et l'analyse ML, renforçant la pertinence d'exploiter le Machine Learning pour détecter automatiquement les menaces dans les données issues de honeypots.
Synthèse personnelle	L'article propose une idée originale : transposer la logique du honeypot dans l'univers du Machine Learning. Il offre une vision complémentaire à l'analyse de logs en cherchant à identifier les comportements adversariaux directement au niveau du modèle. Pour mon PRED, il constitue une référence clé sur la dimension défensive et proactive de l'apprentissage automatique appliqué à la cybersécurité.

Fiche de lecture bibliographique

Référence complète	A. Abdou, R. Sheatsley, Y. Beugin, T. Shipp and P. McDaniel, "Honey-Models : Machine Learning Honeypots," MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM), San Diego, CA, USA, 2021, pp. 886-891, doi : 10.1109/MILCOM52596.2021.9652947.
Auteurs / Année	Ahmed Abdou & Ryan Sheatsley & Yohan Beugin & Tyler Shipp† & Patrick McDaniel
Titre	HoneyModels : Machine Learning Honeypots
Source	10.1109/MILCOM52596.2021.9652947

Objectif de l'étude	Proposer un nouveau mécanisme de défense pour les modèles de Machine Learning appelé *HoneyModels*, inspiré du concept de honeypot. L'objectif est de détecter les attaques adversariales dirigées contre des réseaux de neurones en intégrant un watermark (marque secrète) dans le modèle afin de repérer les comportements suspects.
Contexte et motivation	Les modèles de Machine Learning, notamment les réseaux de neurones profonds, sont vulnérables aux attaques adversariales. Ces attaques consistent à apporter de légères perturbations à un échantillon pour tromper le modèle. Les défenses actuelles (distillation, adversarial training, etc.) sont coûteuses, peu généralisables et faciles à contourner. Les auteurs s'inspirent des honeypots pour créer un modèle « piégé » qui attire et révèle les attaquants sans altérer la performance du modèle légitime.
Données utilisées	Deux jeux de données d'images standards utilisés pour l'évaluation : - **MNIST** (chiffres manuscrits) - **CIFAR-10** (images multi-classes). Les attaques adversariales évaluées incluent : Carlini-Wagner (CW), Projected Gradient Descent (PGD), et Jacobian Saliency Map Attack (JSMA).
Méthodologie / Modèles	- Création d'un « watermark » secret appliqué à certaines caractéristiques du modèle. - Entraînement d'un réseau neuronal altéré via un processus de *self-poisoning* : une fraction des données est modifiée (poisonnée) pour forcer le modèle à intégrer le watermark dans sa structure interne. - À l'inférence, un détecteur (logistic regression) compare le watermark attendu à celui observé pour identifier les échantillons adversariaux. - Trois critères d'évaluation : **Viabilité** (maintien des performances), **Détection** (taux de détection élevé) et **Indistinguishabilité** (impossibilité pour un attaquant de repérer la marque).

Résultats principaux	- Détection jusqu'à 69,5 % des attaques adversariales, avec un taux de faux positifs de 14,1 %. - Maintien d'une précision élevée du modèle d'origine (perte d'environ 12 % seulement pour 90 % de données empoisonnées). - Les adversarial samples générés à partir de HoneyModels sont **indiscernables** de ceux produits par des modèles non protégés, empêchant l'attaquant d'identifier la présence du watermark.
Forces de l'étude	- Approche innovante qui détourne les techniques d'attaque pour améliorer la sécurité. - Faible coût computationnel comparé aux défenses classiques. - Compatible avec différents jeux de données et architectures. - Bon équilibre entre robustesse et précision.
Limites / Biais	- Méthode testée uniquement sur des modèles d'image simples (pas de NLP ou de contexte industriel). - Le taux de détection (70 %) reste insuffisant pour un déploiement opérationnel. - L'approche dépend d'un hyperparamétrage fin (taille et amplitude du watermark, taux de poison).
Lien avec mon projet	Cette étude démontre qu'un modèle de Machine Learning peut être « piégé » pour repérer des interactions malveillantes, exactement comme un honeypot réseau. Elle illustre la fusion entre le concept de honeypot et l'analyse ML, renforçant la pertinence d'exploiter le Machine Learning pour détecter automatiquement les menaces dans les données issues de honeypots.

Synthèse personnelle	L'article propose une idée originale : transposer la logique du honeypot dans l'univers du Machine Learning. Il offre une vision complémentaire à l'analyse de logs en cherchant à identifier les comportements adversariaux directement au niveau du modèle. Pour mon PRED, il constitue une référence clé sur la dimension défensive et proactive de l'apprentissage automatique appliqué à la cybersécurité.

Fiche de lecture bibliographique

Référence complète	I. M. M. Matin and B. Rahardjo, "The Use of Honeypot in Machine Learning Based on Malware Detection : A Review," 2020 8th International Conference on Cyber and IT Service Management (CITSM), Pangkal, Indonesia, 2020, pp. 1-6, doi : 10.1109/CITSM50537.2020.9268794.
Auteurs / Année	Iik Muhamad Malik Matin & Budi Rahardjo
Titre	The Use of Honeypot in Machine Learning Based on Malware Detection : A Review
Source	10.1109/CITSM50537.2020.9268794
Objectif de l'étude	Réaliser une revue systématique de la littérature (Systematic Literature Review - SLR) sur l'utilisation des honeypots dans les approches de détection de malwares basées sur le Machine Learning. L'objectif est d'identifier les tendances de recherche, les techniques employées, et les types de malwares concernés entre 2010 et 2020.

Contexte et motivation	L'augmentation massive des logiciels malveillants rend obsolètes les méthodes traditionnelles de détection (signature, heuristique, statique). Le Machine Learning s'impose comme une alternative plus efficace, mais dépend fortement de la qualité et de la quantité de données d'entraînement. Les honeypots peuvent combler ce manque en générant et collectant automatiquement des données d'attaque réelles, utiles pour entraîner des modèles ML. Cependant, l'absence de schémas normalisés d'intégration entre honeypots et ML rend leur exploitation encore floue.
Données utilisées	Les auteurs ne collectent pas directement de données, mais synthétisent 684 articles identifiés dans les bases **IEEE Xplore** et **ACM Digital Library**, dont 11 sont retenus selon des critères d'inclusion/exclusion stricts (2010–2020). Ces études utilisent soit des honeypots virtuels (VMware, UML, Virtual PC), soit des jeux de données publics (Nebula, IoTPOT).
Méthodologie / Modèles	Méthodologie de revue systématique en trois étapes : planification, exécution et documentation. Les critères PICOC (Population, Intervention, Comparison, Outcomes, Context) guident la sélection. Trois questions de recherche (RQ1–RQ3) structurent l'analyse : 1. Quelles sont les tendances de recherche ? 2. Quelles techniques de honeypot sont utilisées ? 3. Quels types de malwares sont détectés ? Les études sont classées selon leurs approches (honeypot virtuel vs. dataset open source) et les catégories de malwares traitées (IoT, Windows PE, flux réseau).

Résultats principaux	- **Tendance** : augmentation notable des publications entre 2017 et 2019. - **Techniques** : la majorité des études utilisent des honeypots virtuels ou des datasets existants (Nebula, IoTPOT). - **Types de malwares** : prédominance des attaques **IoT botnet**, suivies des malwares Windows exécutables (PE) et des flux réseau malveillants. - Les honeypots permettent d'alimenter efficacement les modèles ML en données réelles et d'améliorer la détection de malwares variés.
Forces de l'étude	- Approche systématique et structurée (SLR) offrant une vision globale du domaine. - Identification claire des tendances et lacunes actuelles. - Mise en évidence de l'importance croissante des honeypots dans la détection ML. - Bon équilibre entre techniques pratiques et analyses conceptuelles.
Limites / Biais	- Aucune expérimentation directe (revue purement bibliographique). - Portée limitée aux bases IEEE et ACM (exclusion de Scopus, Springer...). - Peu de détails sur la qualité et la représentativité des données issues des honeypots. - Absence d'analyse quantitative approfondie (pas de méta-analyse).
Lien avec mon projet	Cette étude met en évidence le rôle clé des honeypots comme sources de données pour l'apprentissage automatique dans la détection de menaces. Elle justifie l'utilisation de logs et traces de honeypots pour développer un pipeline d'analyse basé sur le Machine Learning. Elle constitue un appui théorique solide pour la partie "état de l'art" de mon PRED, notamment pour situer mon travail dans la tendance post-2020 d'automatisation de la cyber threat intelligence.

Synthèse personnelle	Ce travail de revue offre une cartographie claire du domaine « Honeypot + Machine Learning ». Il confirme que les approches de détection basées sur des données issues de honeypots sont pertinentes, notamment dans les environnements IoT. Pour mon PRED, cette étude sert de référence de cadrage méthodologique et aide à positionner ma recherche entre collecte de logs et classification automatisée des menaces.

Fiche de lecture bibliographique

Référence complète	Alshahrani, A. (2023). Predication attacks based on intelligent honeypot technique. Inf. Sci. Lett, 12, 46.
Auteurs / Année	A. Alshahrani
Titre	Predication Attacks Based on Intelligent Honeypot Technique
Source	10.18576/isl/120347
Objectif de l'étude	Présenter un modèle prédictif d'attaques fondé sur une technique de honeypot intelligent intégrant le Machine Learning. L'objectif est d'améliorer la détection des intrusions et de réduire le taux de fausses alertes en combinant les capacités d'observation des honeypots avec la puissance de classification d'un algorithme d'apprentissage supervisé (Decision Tree).
Contexte et motivation	Les systèmes informatiques sont de plus en plus exposés à des attaques sophistiquées, et les outils de détection d'intrusion classiques (IDS/-NIDS) peinent à faire face. Les honeypots offrent un moyen de collecter des données d'attaque réelles mais sont souvent statiques et peu exploitables seuls. L'auteur propose un système hybride qui utilise les données issues des honeypots pour entraîner un modèle d'apprentissage capable de prédire et d'identifier les attaques en temps réel, tout en réduisant le taux de fausses alarmes.

Données utilisées	Les données utilisées proviennent du dataset CIRA-CIC-DoHBrw-2020, enrichi de logs collectés via des honeypots simulant différents services réseau. Ce jeu de données comprend **28 caractéristiques** représentant des flux réseau légitimes et malveillants, ensuite divisées en deux sous-ensembles : 60 % pour l'entraînement et 40 % pour les tests.
Méthodologie / Modèles	Le modèle repose sur une architecture combinant : - un honeypot intelligent pour la collecte et la génération de données d'attaque ; - un prétraitement des données (conversion de caractères et symboles en valeurs numériques) ; - un modèle d'apprentissage supervisé basé sur l'algorithme **Decision Tree (DT)** pour la détection et la classification des intrusions. Les performances sont évaluées selon trois métriques : taux de détection, taux de fausses alarmes et taux d'erreur global.
Résultats principaux	- **Single honeypot** : taux de détection = 95.52 %, fausses alarmes = 4.81 %, temps d'exécution = 13 s. - **Multi-honeypot** : taux de détection = 90.87 %, fausses alarmes = 12.3 %, temps = 43 s. - **Précision globale du modèle DT** : 93.19 %, supérieure à la moyenne observée dans les études antérieures (91 %). L'utilisation du DT améliore nettement la détection tout en réduisant les fausses alertes.
Forces de l'étude	- Intégration réussie entre honeypot et apprentissage automatique. - Amélioration mesurable du taux de détection par rapport à la littérature. - Méthodologie claire et reproductible (training/testing, métriques standardisées). - Application concrète à un jeu de données réel et reconnu (CIRA-CIC-DoHBrw).

Limites / Biais	- Utilisation d'un seul algorithme de Machine Learning (Decision Tree). - Manque de diversité des attaques testées et absence d'évaluation sur des données réelles non simulées. - Pas d'analyse comparative avec des approches plus récentes (Random Forest, Deep Learning).
Lien avec mon projet	Ce travail illustre parfaitement la synergie entre honeypots et Machine Learning pour la détection automatique des menaces. Il constitue une base méthodologique pertinente pour mon PRED, notamment pour la mise en œuvre d'un pipeline de collecte de logs et de classification supervisée des attaques réseau. Les résultats de précision et de réduction des fausses alertes peuvent servir de référence pour évaluer mes propres modèles.
Synthèse personnelle	L'article d'Alshahrani (2023) démontre que l'association entre honeypots et Machine Learning renforce considérablement les capacités de détection proactive. Même si la solution reste expérimentale, elle prouve la faisabilité et la pertinence de l'analyse prédictive des menaces à partir de données de honeypots. C'est une référence directe pour les aspects de "prédiction et classification automatisée des attaques" dans mon projet.

Fiche de lecture bibliographique

Référence complète	Mémoire de Stage Recherche — Automatisation de la génération de rapports CTI via LLM appliqués aux données Honeypot, Orange Innovation, 2024.
Auteurs / Année	Étudiant Orange Innovation, 2024.
Titre	Automatisation de la génération de rapports CTI via LLM appliqués aux données Honeypot.
Source	Document interne (Mémoire de stage).

Objectif de l'étude	Développer une API complète permettant d'analyser automatiquement des données issues de honeypots (payloads, sessions, IoC) et de générer des rapports de Cyber Threat Intelligence (CTI) grâce à des modèles de Large Language Models (LLM). Objectifs : automatiser l'analyse malware, standardiser les rapports CTI, comparer plusieurs LLM et intégrer l'outil dans l'écosystème de sécurité d'Orange.
Contexte et motivation	Les analystes CTI doivent produire des rapports détaillés (IoC, fonctionnement malware, règles de détection, remédiation), un travail chronophage et répétitif. Les LLM permettent d'automatiser ces tâches, mais nécessitent une architecture fiable, contrôlée et sécurisée, surtout lorsqu'ils manipulent des données issues de honeypots (souvent bruitées, hétérogènes et complexes). Le projet vise donc une automatisation robuste de bout-en-bout pour accélérer et fiabiliser les rapports CTI internes.
Données utilisées	Logs et artefacts provenant de plusieurs honeypots du cluster T-Pot utilisés par Orange : - Cowrie (SSH), - Dionaea (malwares binaires), - Honeytrap, Heralding, Elastichoney, etc. Types de données traitées : sessions SSH, commandes malveillantes, fichiers capturés, IoC, URLs, IP, traces d'exécution, payloads réseaux.
Méthodologie / Modèles	Pipeline modulaire conçu autour d'une API : - extraction et nettoyage des données honeypot ; - prompts spécialisés pour analyser le malware, extraire les IoC, générer des règles Snort, et proposer une remédiation ; - génération automatique d'un rapport JSON + HTML final ; - sécurité API : authentification, gestion des clés, isolation des modèles ; - comparaison automatisée de plusieurs LLM (performances, cohérence, hallucinations). Le pipeline supporte différents modèles LLM interchangeables.

Référence complète	Abdallah, A., Alkaabi, A., Alameri, G., Rafique, S. H., Musa, N. S., & Murugan, T. (2024). *Cloud Network Anomaly Detection Using Machine and Deep Learning Techniques – Recent Research Advancements*. IEEE Access. DOI : 10.1109/ACCESS.2024.3390844.
Auteurs / Année	Amira Abdallah, Aysha Alkaabi, Ghaya Alameri, Saida Hafsa Rafique, Nura Shifa Musa, Thangavel Murugan — 2024.
Titre	Cloud Network Anomaly Detection Using Machine and Deep Learning Techniques – Recent Research Advancements.
Source	DOI : 10.1109/ACCESS.2024.3390844.
Objectif de l'étude	Présenter une revue récente et complète des techniques de Machine Learning (ML) et Deep Learning (DL) appliquées à la détection d'anomalies dans les réseaux cloud. L'objectif est de regrouper les méthodes existantes, les datasets utilisés, leurs avantages/limites, ainsi que les perspectives de recherche dans le domaine.
Contexte et motivation	Les environnements cloud sont largement utilisés et exposés à des attaques toujours plus complexes, notamment les attaques DDoS. Les systèmes de détection classiques ne sont pas adaptés aux volumes massifs et à la nature dynamique du trafic cloud. Les approches ML/DL offrent des solutions plus efficaces pour apprendre des comportements normaux et identifier les anomalies.
Données utilisées	La revue recense les principaux datasets utilisés dans la recherche ML/DL pour la détection d'anomalies cloud : - NSL-KDD, CICIDS 2017, CSE-CIC-IDS2018 ; - CICDDoS2019, CAIDA DDoS 2007 ; - BoT-IoT, Kyoto ; - divers logs IoT, DNS, MQTT, et flux réseau. Le papier fournit un tableau détaillé des caractéristiques (nombre de features, types d'attaques, nombre d'échantillons…).

Méthodologie / Modèles	Revue structurée des approches suivantes : - ML : SVM, Random Forest, K-Means, modèles bayésiens ; - DL : CNN, LSTM, Autoencoders, GAN, modèles séquentiels ; - Modèles hybrides (CNN+optimisation, AE+SVM, Deep Belief Networks…). L'étude couvre principalement la littérature publiée entre 2020 et 2024.
Résultats principaux	- Les modèles DL surpassent les approches ML classiques sur les anomalies complexes. - Les modèles hybrides atteignent souvent des précisions supérieures à 98 %. - Les tendances récentes incluent l'usage de GNN (Graph Neural Networks) et de modèles seq2seq pour analyser le trafic réseau. - Les limites des travaux existants : manque de datasets réalistes, forte variabilité selon les jeux de données, peu de travaux sur les attaques d'évasion.
Forces de l'étude	- Revue exhaustive et bien structurée. - Très bonne synthèse des approches ML et DL modernes. - Vision claire des tendances 2020–2024. - Analyse détaillée des datasets et de leurs caractéristiques.
Limites / Biais	- Aucune expérimentation propre : étude purement bibliographique. - Plusieurs datasets utilisés dans la littérature ne reflètent plus les environnements cloud réels. - Peu de discussion sur les coûts computationnels des approches DL. - Absence de réflexion sur l'industrialisation ou le déploiement en production.

Fiches de lecture

Résultats principaux	- API fonctionnelle, sécurisée, modulaire et documentée ; - Production automatique de rapports CTI complets ; - Amélioration notable du temps d'analyse d'un malware ; - Standardisation des rapports internes ; - Comparaison multi-LLM montrant : - forte variabilité selon les tâches (STIX, règles Snort, explicabilité), - hallucinations partiellement contrôlées.
Forces de l'étude	- Projet complet mêlant cybersécurité, NLP, prompts, API, DevSecOps, - Très forte applicabilité opérationnelle pour les équipes CTI ; - Architecture propre, évolutive et sécurisée ; - Intégration facile dans un pipeline existant d'Orange.
Limites / Biais	- Hallucinations encore présentes selon les modèles ; - Score automatique encore limité (nécessite un meilleur jeu de référence) ; - Dépendance aux LLM externes (coût, confidentialité) ; - Pas encore d'intégration poussée de modèles locaux fine-tunés.
Lien avec mon projet	Cette étude montre l'importance d'automatiser l'analyse des données issues des honeypots, ce qui est directement en lien avec mon PRED. Elle confirme la pertinence d'utiliser des pipelines ML/NLP pour traiter automatiquement les logs, extraire des menaces, et standardiser la production d'information. C'est un bon exemple d'automatisation avancée appliquée à des données honeypot complexes.
Synthèse personnelle	Ce mémoire illustre la transition vers une cybersécurité augmentée par l'IA. L'intégration de LLM pour analyser et transformer des données honeypot brutes en rapports CTI exploitables représente une avancée majeure. Pour mon projet, il offre un cadre réaliste et opérationnel sur

Lien avec mon projet	Cette revue est particulièrement utile pour l'approches Deep Learning appliquées à l'Logs réseau. Elle montre comment les (LSTM, Autoencoders) peuvent être pertine à des logs provenant de honeypots. Elle IDS pour traiter des logs complexes et vo
Synthèse personnelle	Une étude très riche pour comprendre les a d'anomalies réseau. Elle clarifie les tenda des approches utilisées en recherche. Dans permet de justifier l'intégration du Deep honeypot et de positionner mon travail techniques de détection.

Fiche de lecture bibliographique

Référence complète	Shan Ali, Chaima Boufaied, Domenico Bianculli, Paula Branco, and Lionel Briand, "A comprehensive study of machine learning techniques for log-based anomaly detection," Empirical Software Engineering, vol. 30, no. 5, p. 129, 2025.
Auteurs / Année	Shan Ali et al. (2025)
Titre	A comprehensive study of machine learning techniques for log-based anomaly detection
Source	DOI : 10.1007/s10664-025-10669-3
Objectif de l'étude	Mener une étude empirique massive pour comparer objectivement les performances des techniques de Machine Learning (ML) classique et de Deep Learning (DL) pour la détection d'anomalies dans les logs. L'objectif est de vérifier si la complexité et le coût du Deep Learning sont justifiés par un gain de performance réel par rapport aux méthodes supervisées classiques.
Contexte et motivation	La littérature académique récente tend à privilégier quasi-systématiquement le Deep Learning (LSTM, Transformers) pour l'analyse de logs, souvent sans comparaison rigoureuse avec des baselines solides. Les auteurs remettent en cause ce dogme en introduisant des critères d'évaluation souvent ignorés : le coût computationnel (temps d'entraînement/inférence) et la stabilité face aux hyperparamètres.

Données utilisées	7 datasets de référence (Benchmarks) couvrant des millions de logs, divisés en deux catégories : - **Session-based (Structurés)** : HDFS (11M logs), Hadoop, F-dataset. - **Log message-based (Non-structurés)** : BGL (4.7M logs), Thunderbird, Spirit, Hades. Cela permet de tester la robustesse des modèles face à des données de natures très différentes.
Méthodologie / Modèles	Comparaison systématique de 9 algorithmes sur les 7 datasets : - **Supervisés Classiques** : Random Forest (RF), SVM. - **Deep Learning** : LSTM, LogRobust (Attention-based), NeuralLog (Transformer). - **Semi-supervisés** : DeepLog, OC-SVM, Logs2Graphs (GNN). Protocole rigoureux mesurant : F1-Score, Temps d'entraînement (sur GPU A100), Temps d'inférence, et analyse de sensibilité aux hyperparamètres.
Résultats principaux	- **Performance** : Sur les logs structurés (HDFS), le Random Forest (**99.8%**) égale les modèles DL les plus complexes (NeuralLog : 99.8%). Le DL ne surpasse le ML classique que sur des logs très bruités (BGL). - **Coût** : Le temps d'entraînement du Random Forest (96s) est jusqu'à **600 fois inférieur** à celui du NeuralLog (57 000s). - **Latence** : Le Random Forest permet une inférence quasi-instantanée (0.89s pour tout le dataset HDFS) contre une latence élevée pour les Transformers (70s).
Forces de l'étude	- Rigueur expérimentale exceptionnelle (comparaison exhaustive). - Prise en compte des coûts réels (temps de calcul), cruciaux pour l'industrie. - Démystification de la supériorité systématique du Deep Learning.

Fiche de lecture bibliographique

Référence complète	Lanka, Phani, Khushi Gupta, and Cihan Varol. "Intelligent Threat Detection—AI-Driven Analysis of Honeypot Data to Counter Cyber Threats." Electronics 13.13 (2024) : 2465.
Auteurs / Année	Lanka, Gupta & Varol (2024)
Titre	Intelligent Threat Detection—AI-Driven Analysis of Honeypot Data to Counter Cyber Threats
Source	DOI : 10.3390/electronics13132465
Objectif de l'étude	Proposer une méthodologie nouvelle utilisant les Grands Modèles de Langage (LLM) et le RAG (Retrieval-Augmented Generation) pour analyser **sémantiquement** les commandes shell offusquées capturées par des honeypots, là où les signatures classiques et le ML statistique échouent.
Contexte et motivation	Les attaquants utilisent des techniques d'offuscation (variables aléatoires, encodage base64, alias) pour masquer leurs commandes. Les systèmes de détection basés sur des mots-clés (Regex) sont aveugles à ces variations syntaxiques. L'étude vise à exploiter la capacité de compréhension du langage des LLM pour détecter l'*intention* malveillante derrière la syntaxe.
Données utilisées	Logs collectés sur une infrastructure de honeypots SSH haute interaction déployés sur le cloud AWS (USA, Europe, Asie) pendant une fenêtre de 7 jours. Total : **47 764 commandes brutes**, provenant de 1 154 adresses IP uniques.

Méthodologie / Modèles	Pipeline en 3 étapes (Architecture RAG) : - **Normalisation** : Utilisation de la librairie *Bashlex* pour parser les commandes et abstraire les variables (IPs, fichiers) en tokens génériques. - **Vectorisation** : Transformation des commandes normalisées en vecteurs (Embeddings) via le modèle *Salesforce/SFR-Embedding-Mistral*. - **Analyse Sémantique** : Recherche de similarité vectorielle (Cosine Similarity > 0.75) et qualification par un LLM (GPT-4-turbo) pour mapper l'attaque sur le framework MITRE ATT&CK.
Résultats principaux	- Sur 47 764 commandes, l'approche a permis de réduire le dataset à seulement **165 modèles d'attaques uniques** (clusters sémantiques). - Le système détecte des variantes d'attaques inconnues (Zero-Day) par simple proximité sémantique avec des attaques connues, sans signature préalable. - Le LLM fournit une **explication textuelle** de l'attaque, facilitant le travail de l'analyste SOC.
Forces de l'étude	- Approche très innovante (première application documentée du RAG aux logs SSH). - Résout le problème de l'explicabilité ("Pourquoi c'est une attaque ?"). - Testé sur des données réelles du Cloud, reflétant des menaces actives.
Limites / Biais	- Coût d'inférence élevé (appel API GPT-4 pour chaque analyse approfondie). - Latence non compatible avec le temps réel strict (plusieurs secondes par analyse), nécessitant une exécution asynchrone.
Lien avec mon projet	C'est la source d'inspiration technique pour la partie **"Exploration / Innovation"** de mon projet. Elle fournit le modèle technique (Bashlex + Embeddings) que je compte implémenter pour analyser les commandes que le Random Forest n'arrive pas à classifier avec certitude.

Limites / Biais	- L'étude se concentre sur la détection binaire (Anomalie/Normal) et moins sur la classification multi-classes d'attaques spécifiques. - Les datasets utilisés sont des standards académiques parfois anciens (HDFS date de 2009).
Lien avec mon projet	C'est la **référence critique centrale** de mon état de l'art. Elle fournit la justification scientifique pour privilégier le Random Forest comme "socle technique" (pour son efficacité prouvée) et pour ne réserver le Deep Learning/LLM qu'aux cas complexes où le ML classique échoue (comme démontré sur le dataset BGL).
Synthèse personnelle	Cette étude est fondamentale car elle valide l'approche pragmatique de mon projet : ne pas utiliser une technologie complexe par simple "hype". Elle prouve que pour des logs structurés, l'algorithme Random Forest est l'état de l'art réel en termes d'efficience.

Synthèse personnelle (Aurélie/Noémie) : Cet article marque une rupture technologiq statistique à la compréhension sémantique. J apporte la brique "Intelligente" qu du Random Forest dans mon architecture h

Fiche de lecture bibliographique

Référence complète	Lee, S., Abdullah, A., Jhanjhi, N., & Kok, S. (2021). Classification of botnet attacks in IoT smart factory using honeypot combined with machine learning. PeerJ Computer Science, 7, e350.
Auteurs / Année	Lee et al. (2021)
Titre	Classification of botnet attacks in IoT smart factory using honeypot combined with machine learning
Source	DOI : 10.7717/peerj-cs.350
Objectif de l'étude	Développer et valider un système de détection d'intrusions léger et rapide pour protéger les usines connectées (Smart Factories) contre les botnets IoT, en utilisant une architecture de honeypots distribués couplée à du Machine Learning.
Contexte et motivation	Les environnements industriels (IoT/SCADA) ont des ressources limitées (peu de CPU/RAM) et exigent une latence minimale pour ne pas perturber la production. Les solutions de sécurité lourdes (Deep Learning complexe) ne sont pas applicables. L'objectif est de prouver qu'un modèle classique peut être à la fois précis et ultra-rapide sur ce type de matériel.
Données utilisées	Données générées par un banc d'essai physique simulant une usine intelligente (capteurs, contrôleurs, RFID) attaquée par des botnets réels (notamment Mirai et Gafgyt). La collecte est effectuée via des honeypots T-Pot installés sur des **Raspberry Pi**.
Méthodologie / Modèles	- Déploiement de honeypots sur matériel contraint (Raspberry Pi). - Entraînement de modèles via la plateforme Weka. - Comparaison de plusieurs algorithmes : Random Forest, J48 (Decision Tree), Naive Bayes. - Mesure critique du **temps de détection** (latence) et du taux de faux positifs (FPR).

Résultats principaux	- Le modèle **Random Forest** atteint la meilleure précision globale de **96,66%**. - Le temps de détection est extrêmement faible : **0,1 milliseconde** par paquet. - Le taux de faux positifs est maintenu très bas (0,24%), ce qui est crucial pour éviter les arrêts de production injustifiés.
Forces de l'étude	- Validation en environnement contraint réel (Preuve de concept IoT). - Mesure précise de la latence, un critère souvent oublié dans les papiers théoriques. - Confirmation de l'efficacité du Random Forest sur du matériel modeste.
Limites / Biais	- Environnement simulé en laboratoire, potentiellement moins bruité qu'une vraie usine en production. - L'étude se concentre uniquement sur les signatures de botnets connus.
Lien avec mon projet	Cette étude justifie le choix du Random Forest pour la partie "Performance" de mon projet. Elle prouve que ce modèle est capable de traiter des flux en temps réel avec une latence négligeable, ce qui est un argument fort pour mon architecture en "entonnoir" (filtrage rapide avant analyse lente).
Synthèse personnelle	Un papier très pragmatique qui rappelle que la complexité n'est pas toujours la solution. Pour des systèmes embarqués ou temps réel, l'efficacité du ML classique reste imbattable face aux réseaux de neurones profonds.

B

Planning

Nous avons constaté, lors de cette première partie du projet, que l'estimation de la durée des tâches est un exercice complexe, particulièrement dans un contexte de recherche exigeant une phase bibliographique initiale chronophage. En effet, l'évaluation difficile de notre temps disponible durant les premières semaines a engendré un retard sur la réalisation de l'état de l'art. Un enseignement majeur pour l'établissement d'un planning prévisionnel est donc la nécessité d'intégrer avec plus de précision la disponibilité effective allouée au projet par les différents participants.

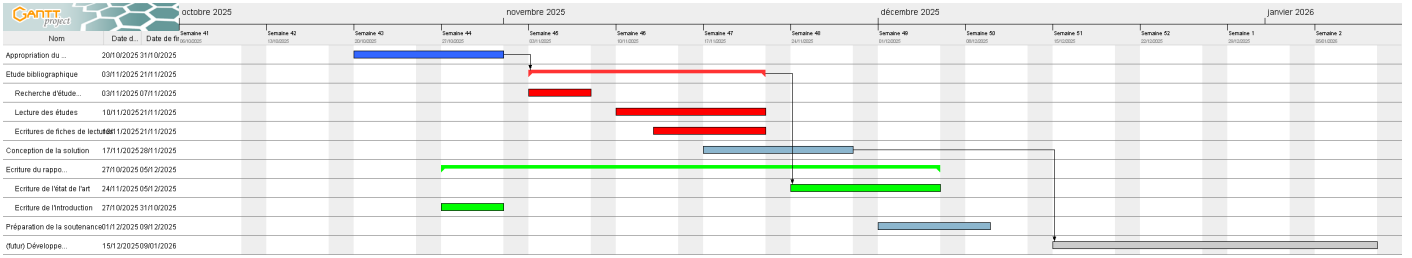


FIGURE B.1 – Gantt prévisionnel sur le travail à réaliser

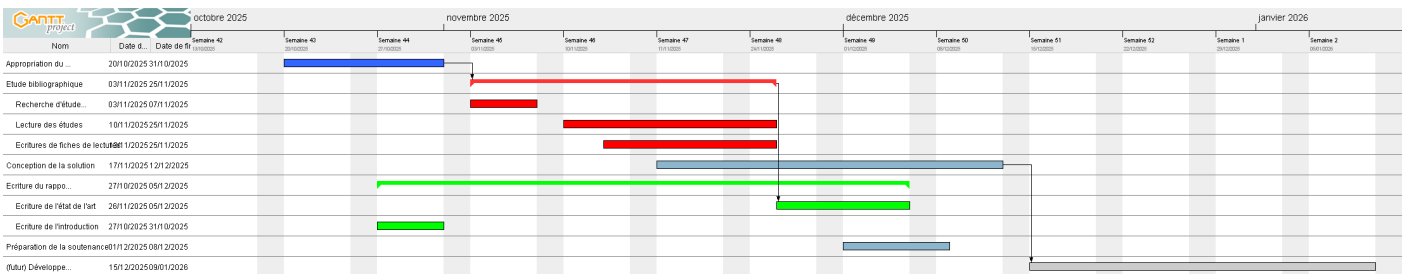


FIGURE B.2 – Gantt réajusté avec ce qui a réellement été fait

C

Rapport Hebdomadaires

Fiche de suivi de la semaine 1
du 20 octobre 2025 au 25 octobre 2025

Temps de travail de Luka FORTUN: 3 h 30 m

Travail effectué.

- Initialisation du rapport : premier jet du contexte du sujet et de la problématique ;
- Documentation sur l'état de l'art (lecture et étude de la bibliographie proposée dans le sujet) ;

Travail non effectué.

Échanges avec le commanditaire.

Planification pour la semaine prochaine.

- Continuer les recherches bibliographiques sur l'état de l'art ;
- Réfléchir aux différentes solutions vers lesquels se tourner pour répondre au problème du sujet ;
- Discuter avec l'encadrant sur les détails du sujet ;

Fiche de suivi de la semaine 2
du 2 novembre 2025 au 9 novembre 2025

Temps de travail de Luka FORTUN: 10 h 30 m

Travail effectué.

- Rajout d'une partie sur les objectifs du projet au rapport
- Documentation sur l'état de l'art (lecture et étude de la bibliographie proposée dans le sujet ainsi que des recherches annexes d'autres études) ;
- Début de réalisation de fiches de lectures résumant chaque article lu dans le cadre de l'état de l'art.

Travail non effectué.

- Réfléchir aux différentes solutions vers lesquels se tourner pour répondre au problème du sujet ; Trop tôt dans le projet pour pouvoir faire décider de ce genre de choses, besoin de beaucoup plus faire l'état de l'art et la bibliographie avant de faire ceci.

Échanges avec le commanditaire.

Planification pour la semaine prochaine.

— Commencer à écrire un plan de l'état de l'art ;
— Continuer l'approfondissement de l'état de l'art sur le sujet ;
— Approfondir l'état de l'art du côté du Deeplearning ;

Fiche de suivi de la semaine 3
du 9 novembre 2025 au 16 novembre 2025

Temps de travail de Luka FORTUN: 3 h 30 m

Travail effectué.

— Relecture et structuration des fiches de lecture déjà produites ;
— Intégration de nouvelles études orientées Deep Learning (LSTM, Autoencoders, CNN) pour élargir l'état de l'art ;
— Début d'identification des grandes catégories de travaux (honeypots, ML classique, DL pour l'analyse de logs).

Travail non effectué.

— Rédaction d'un plan structuré de l'état de l'art : semaine chargée en parallèle par d'autres projets, ce qui a limité le temps alloué ;
— Harmonisation complète des fiches de lecture : reportée à la semaine suivante.

Échanges avec le commanditaire.

Aucun échange formel cette semaine : travail principalement centré sur la collecte d'articles supplémentaires et la compréhension de leurs apports.

Planification pour la semaine prochaine.

— Finaliser un premier plan de l'état de l'art (structuration en parties et sous-parties) ;
— Continuer la recherche d'articles sur le Deep Learning appliqué aux logs de sécurité ;
— Rédiger les premières sections de l'état de l'art à partir des fiches de lecture.

Fiche de suivi de la semaine 4
du 17 novembre 2025 au 23 novembre 2025

Temps de travail de Luka FORTUN: 3 h 00 m

Travail effectué.

— Création d'un premier plan d'état de l'art (à finir dans la semaine prochaine)
— Rédiger les premières sections de l'état de l'art à partir des fiches de lecture.

Travail non effectué.

— Continuer la recherche d'articles sur le Deep Learning appliqué aux logs de sécurité ; Manque de temps causé par la préparation de la Nantarena (projet d'envergure à Polytech)

Échanges avec le commanditaire.

Aucun échange formel cette semaine : travail principalement centré sur la collecte d'articles supplémentaires et la compréhension de leurs apports.

Planification pour la semaine prochaine.

- Continuer la recherche d'articles sur le Deep Learning appliqué aux logs de sécurité ;
- Finir la rédaction de l'état de l'art
- Réfléchir à une premier de solution

Fiche de suivi de la semaine 5
du 24 novembre 2025 au 30 novembre 2025

Temps de travail de Luka FORTUN: 13 h 00 m

Travail effectué.

- Finalisation d'une première version de l'état de l'art pour le rapport intermédiaire ;
- Reflexions et recherches sur la solution proposée ;
- Recherches supplémentaires plus techniques sur les Honeypots ;

Travail non effectué.

Échanges avec le commanditaire.

Planification pour la semaine prochaine.

- Corriger et paufiner l'état de l'art ;
- Améliorer et finaliser une première solution technique et la soumettre au client ;
- Finaliser le rapport intermédiaire ;

Fiche de suivi de la semaine 6
du 1 décembre au 7 décembre 2025

Temps de travail de Luka FORTUN: 28 h 00 m

Travail effectué.

- Finaliser le rapport intermédiaire ;
- Corriger et paufiner l'état de l'art ;
- Améliorer et finaliser une première solution technique ;
- Préparation de la soutenance ;

Travail non effectué.

Échanges avec le commanditaire.

Planification pour la semaine prochaine.

- Commencer a explorer la mise en place concrete de la solution technique ;
- Définir clairement le planning du développement de la solution ;

Le tableau C.1 récapitule le taux d'avancement du projet. Rappelons que le temps de travail théorique *mini-*

	Temps prévu		Luka FORTUN					
	bas	haut	hebdo.	Σ	%	hebdo.	Σ	%
Semaine	h : m	h : m	h : m	h : m		h : m	h : m	
1	10 : 00	12 : 30	3 : 30	3 : 30	35 (28)	:	:	
2	20 : 00	25 : 00	10 : 30	14 : 00	70 (56)	:	:	
3	30 : 00	37 : 30	3 : 30	17 : 30	58 (46)	:	:	
4	40 : 00	50 : 00	3 : 00	20 : 30	51 (41)	:	:	
5	50 : 00	62 : 30	13 : 00	33 : 30	67 (53)	:	:	
6	60 : 00	75 : 00	28 : 00	61 : 30	102 (82)	:	:	

TABLE C.1 – Avancement du projet par rapport au temps de travail théorique minimal (respectivement haut)

mal correspond au temps indiqué sur la maquette pédagogique auquel on ajoute un strict minimum de 20 % correspondant au travail personnel hors emploi du temps. La partie « haute » de la fourchette correspond à 50 % de temps supplémentaire au titre du travail personnel.

Auto-évaluation

Rapport	Organisation	Plan	Equilibre			
			Cohérence		1	
		Fluidité	Introductions (partielles)			1
			Transitions		1	
			Conclusions (partielles)			1
		Tableaux, figures	Numérotés		1	
			Légendés		1	
	Rédaction	Orthographe	Références (non "en ligne")		1	
			Coquilles		1	
			Fautes évitables		1	
			Franglais, jargon			1
		Rédaction	Aisée		1	
			Absence de plagiat !		1	
		Bibliographie	Références	Suffisantes (nombre, intérêt)		1
				Pérennes		1
				Complètes (auteurs, pages..)	1	
			Références dans le texte	Conséquentes (volume)	1	
					1	

Proposition de note haute	17,03
Proposition de note basse	12,03

Proposition de note du jury

Projection	Organisation	Plan			1		
		Liaisons			1		
	Contenu	Numérotation			1		
		Informatif				1	
		Concis			1		
		Clair				1	
		Orthographe			1		
		Illustrations				1	
Oral	Présentation	Aisance			1		
		Tenue				1	
		Articulation, compréhension			1		
	Durée	Respect				1	
		Temps de parole équilibré			1		
	Réponses	Pertinence			1		
		Argumentation			1		

Proposition de note haute	17,78
Proposition de note basse	12,96

Proposition de note du jury

Travail	Etude	Bibliographie	Adéquate		1		
			Suffisante		1		
		Cahier des charges	Clair			1	
			Formalisé			1	
		Hypothèses envisagées	Nombre			1	
			Pertinence			1	
		Validation	Analyse *a priori*			1	
			Tableau comparatif		1		
			Choix argumenté(s)		1		
			Faisabilité		1		
	Complexité	Temps consacré				1	
		Résultats obtenus				1	
		Difficulté	Intrinsèque		1		
			Vis-à-vis du binôme		1		
	Annexes	Fiches d'avancement	Régularité			1	
			Détaillées				1
		Gantt	Prévisionnel et justifications				1
			Effectifs, erreurs, corrections				1

Proposition note haute	16,36

FIGURE D.1 – Auto Evaluation de la phase 1