Document Technique : SAHEL GUARD

Une Architecture de Cybersécurité Décentralisée pour les Infrastructures Critiques sur AWS

Version : 1.1

Date : 23 Octobre 2025

Auteurs : L'équipe SAHEL GUARD

Résumé Exécutif (Abstract)

SAHEL GUARD est un système de détection et de réponse aux intrusions (IDPS) de nouvelle génération, conçu pour répondre aux défis de sécurité des infrastructures critiques dans des environnements où la confiance est une ressource rare. Le projet exploite la puissance du réseau Hedera pour créer une couche de confiance distribuée, garantissant l'immuabilité, la transparence et la traçabilité des événements de sécurité. En combinant une intelligence artificielle prédictive, un système de réputation pour les capteurs et des mécanismes d'incitation économique via le Hedera Token Service (HTS), SAHEL GUARD propose un modèle de cybersécurité participatif et vérifiable. Le Hedera Consensus Service (HCS) est utilisé comme un journal d'audit inviolable pour toutes les alertes et signatures de menaces. L'architecture est conçue pour une mise en production sur Amazon Web Services (AWS), assurant une haute disponibilité, une scalabilité et une sécurité de niveau entreprise.

1. Introduction

La sécurisation des infrastructures numériques critiques (énergie, télécommunications, services financiers) est un enjeu majeur. Les systèmes de sécurité traditionnels, souvent centralisés, souffrent de plusieurs lacunes :

- **Opacité :** Les journaux d'événements sont stockés dans des bases de données propriétaires, rendant l'audit externe complexe et sujet à caution.
- **Manque de Confiance :** Dans un contexte multi-acteurs (agences gouvernementales, opérateurs privés), il n'existe aucune garantie que les données n'ont pas été altérées ou supprimées.
- **Modèle Passif :** Les systèmes de surveillance manquent souvent de mécanismes pour inciter la qualité et la rapidité des détections, reposant sur une simple obligation de service.

SAHEL GUARD a été conçu pour répondre à ces défis en s'appuyant sur un postulat fondamental : la confiance ne doit pas être une hypothèse, mais une propriété architecturale du système.

Vision et Objectifs :

Notre vision est de créer un écosystème de cybersécurité où la transparence et la collaboration sont encouragées par la technologie. Les objectifs du projet sont :

- **Confiance par la Preuve :** Utiliser Hedera HCS pour créer un registre d'audit immuable et chronologiquement ordonné de toutes les menaces.
- **Intelligence Collective :** Mettre en place un registre public de signatures de menaces sur HCS pour un partage de renseignements décentralisé.
- **Incitation à l'Excellence :** Développer une économie de tokens avec Hedera HTS pour récompenser les capteurs les plus performants.
- **Défense Proactive :** Intégrer une IA capable non seulement de détecter, mais aussi de prédire les menaces en analysant les flux de données en temps réel.
- **Déploiement Robuste :** Concevoir une architecture prête pour la production, tirant parti des services cloud d'AWS pour la scalabilité et la résilience.

2. Architecture du Système

L'application est structurée autour d'une architecture client-serveur avec une intégration profonde des services Hedera.

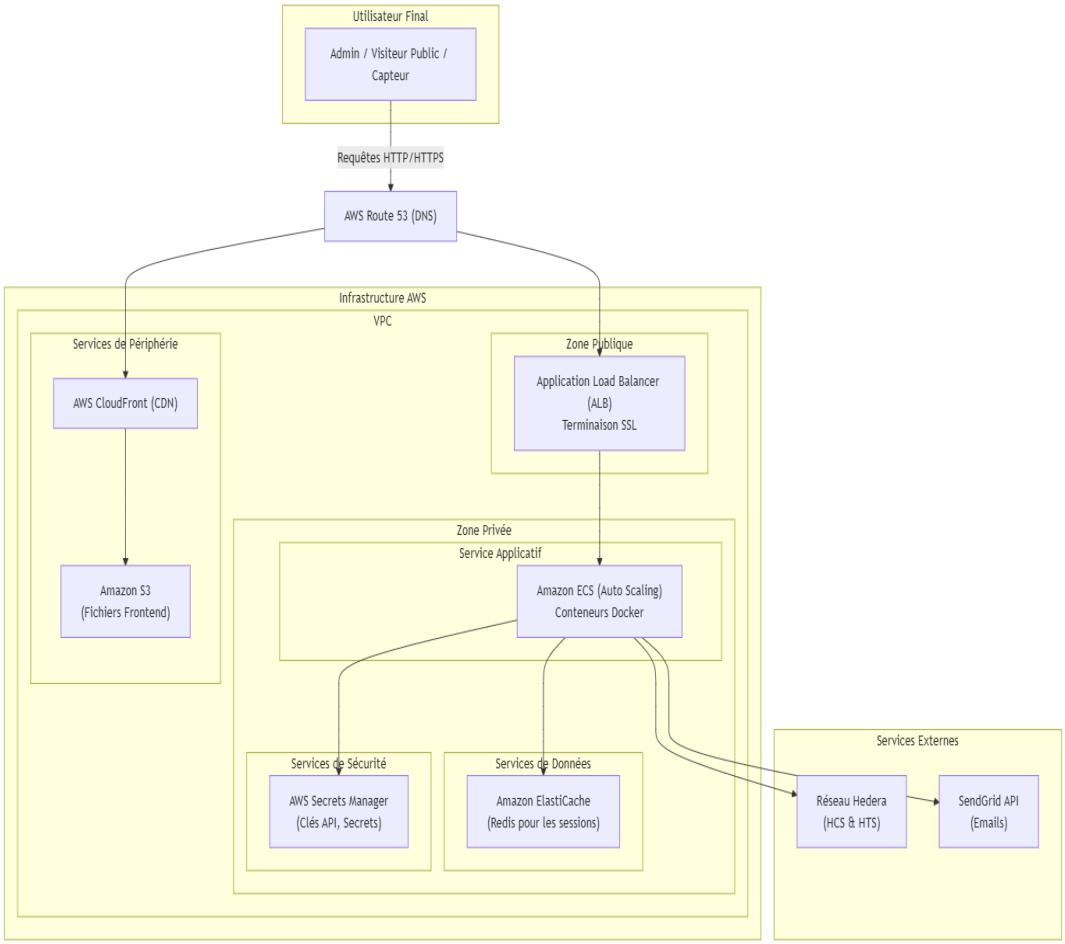
L'architecture de SAHEL GUARD est conçue pour être modulaire, sécurisée et hautement évolutive, en s'appuyant sur les services managés d'Amazon Web Services (AWS).

2.1. Architecture Applicative

- **Backend (Node.js / Express.js)** : Cœur logique de l'application, il orchestre les différents services :
    - **API et Serveur WebSocket (server.js) :** Point d'entrée pour les clients et orchestrateur principal. Il gère l'authentification, la configuration et la diffusion des événements en temps réel via Socket.IO.
    - **Moteur d'Analyse (analyzeTraffic) :** Reçoit les données réseau simulées et les soumet aux différents services d'analyse.
    - **Services Spécialisés :** Moteur d'IA Prédictive, Moteur de Règles Métier, Service d'Intégration Hedera, Service de Réputation, Service de Réponse Active, et Service de Tokens.
- **Frontend (HTML/CSS/JS)** :
    - **Tableau de Bord Administrateur (admin.html, admin.js) :** Interface riche pour la surveillance, la configuration et l'analyse détaillée.
    - **Page de Statut Public (user.html, user.js) :** Vue simplifiée et transparente destinée au grand public ou aux partenaires.

2.2. Architecture de Déploiement sur AWS

Le diagramme ci-dessous illustre l'architecture de déploiement cible sur AWS, montrant le flux des requêtes et l'interaction entre les différents services



- **Calcul et Réseau :**
  - **Amazon ECS :** Le backend Node.js est conteneurisé (Docker) et déployé sur **Amazon Elastic Container Service (ECS)**, permettant de gérer et de faire évoluer les conteneurs de manière automatique.
  - **Application Load Balancer (ALB) :** Placé devant les conteneurs ECS, il distribue le trafic, gère la terminaison SSL (HTTPS) et assure la haute disponibilité. C'est ce composant qui justifie la configuration app.set('trust proxy', 1).
  - **Auto Scaling Group :** Le service ECS est configuré pour faire évoluer automatiquement le nombre de conteneurs en fonction de la charge.
- **Stockage et Distribution de Contenu :**
  - **Amazon S3 & CloudFront :** Les fichiers statiques du frontend (HTML, CSS, JS) sont hébergés sur **Amazon S3** et distribués via **Amazon CloudFront**

(CDN), réduisant la charge sur le serveur Node.js et accélérant le temps de chargement.

- **Gestion des Données et des Sessions :**
  - **Amazon ElastiCache for Redis :** Le système de session basé sur des fichiers (session-file-store) est remplacé par **Amazon ElastiCache** (Redis) pour permettre le partage de l'état des sessions entre plusieurs instances de l'application.
  - **AWS Secrets Manager :** Les informations sensibles (clés API Hedera, SendGrid, secret de session) sont gérées de manière sécurisée par **AWS Secrets Manager**, qui fournit une gestion centralisée des secrets avec rotation automatique et politiques d'accès granulaires.

3. Analyse Approfondie des Composants Clés

3.1. Le Moteur de Détection d'IA (ai-detection-advanced.js)

Notre approche de l'IA est hybride, combinant une analyse comportementale avec une logique prédictive basée sur l'historique des flux réseau.

- **Analyse Comportementale "Stateful" :** Le moteur maintient un état des communications en cours dans une Map (activeFlows). Pour chaque flux (défini par source_IP > destination_IP : port), il conserve un historique des paquets, un score de menace et un timestamp. Cette approche "stateful" est cruciale pour détecter des menaces complexes qui se déroulent sur la durée.
- **Logiques de Prédiction Spécifiques :** Le moteur implémente des heuristiques pour prédire des types d'attaques spécifiques :
  1. **Prédiction de Beaconing (Malware C2) :** Le système calcule l'intervalle de temps moyen et l'écart-type entre les paquets d'un même flux. Un faible écart-type couplé à un intervalle régulier est un indicateur fort d'une communication automatisée de type "Command & Control".
  2. **Prédiction d'Attaque par Force Brute :** Le moteur détecte une succession rapide de paquets de petite taille vers un port de service commun (21, 22, 3389).
  3. **Prédiction d'Exfiltration de Données :** En calculant le volume total de données sur un flux sortant, le système peut identifier des transferts anormalement élevés.

3.2. Le Service de Réponse Active (active-response-service.js)

Ce service automatise les contre-mesures pour neutraliser les menaces confirmées.

- **Logique de Décision :** Une action n'est déclenchée que pour les menaces de sévérité high ou critical.
- **Validation par la Réputation :** Pour éviter les faux positifs coûteux, le système vérifie la réputation du capteur (sensor-reputation.js). Si une menace est détectée par un capteur de faible réputation (ex : "Bronze"), l'action automatique est suspendue en attente d'une validation manuelle.
- **Action (Simulation) :** L'action par défaut est le blocage de l'adresse IP source. L'action est enregistrée et diffusée sur le tableau de bord pour une transparence totale.

4. Intégration Stratégique des Services Hedera

4.1. Hedera Consensus Service (HCS) : La Source de Vérité Immuable

L'utilisation d'Hedera n'est pas un ajout, mais le fondement de la confiance dans SAHEL GUARD.

- **Implémentation (hedera-config.js) :**
    1. **Création de Topics Dédiés :** Au démarrage, le système crée deux topics HCS via TopicCreateTransaction:
        - **Topic des Alertes :** Chaque alerte de sécurité validée est formatée en JSON et soumise à ce topic via une TopicMessageSubmitTransaction.
        - **Topic des Signatures de Menaces :** Pour chaque menace, une "signature" (ex : sourcePattern : "154.16.*. *") est extraite et soumise à ce second topic, créant une base de renseignements sur les menaces (Threat Intelligence) partagée et immuable.
- **Avantages Techniques :**
    - **Immuabilité et Ordonnancement :** HCS garantit que chaque message est enregistré de manière permanente et reçoit un timestamp consensuel, empêchant toute altération.
    - **Haute Performance et Faible Coût :** La capacité de HCS à traiter des milliers de transactions par seconde à un coût prédictible et très bas le rend idéal pour une application de journalisation à haut volume.

- **Transparence Auditable :** Des tiers (régulateurs, auditeurs) peuvent s'abonner au topic HCS via un "mirror node" pour vérifier de manière indépendante et en temps réel le flux d'alertes.

4.2. Hedera Token Service (HTS) : L'Économie de la Participation

- **Implémentation (token-service-simple.js) :**
    1. **Système de Réputation (sensor-reputation.js) :** Chaque capteur gagne des points d'expérience (XP) pour chaque alerte valide. L'XP permet de monter en niveau (Bronze, Silver, Gold...), chaque niveau étant associé à un multiplicateur de récompense.
    2. **Calcul Dynamique des Récompenses :** Le service de token calcule une récompense de base (liée à la sévérité de l'alerte) et la multiplie par le bonus de réputation du capteur.
    3. **Distribution (Simulation) :** La fonction distributeReward simule la distribution de HBAR. Le fichier token-service-backup.js contient une implémentation prête à l'emploi avec de vraies transactions HTS (TokenCreateTransaction, TransferTransaction), démontrant la faisabilité technique en production.
- **Avantages Stratégiques :**
    - **Création d'un Cercle Vertueux :** Ce modèle incite les opérateurs de capteurs à maintenir leur équipement et à améliorer leurs algorithmes de détection pour maximiser leurs récompenses.
    - **Gouvernance Future :** La possession de tokens peut servir de base à un système de gouvernance décentralisée (governance-service.js).

Conclusion :

SAHEL GUARD est bien plus qu'un simple prototype de hackathon ; il s'agit d'une démonstration fonctionnelle et d'une vision architecturale pour la prochaine génération de systèmes de cybersécurité. En relevant les défis fondamentaux d'opacité et de manque de confiance inhérents aux systèmes centralisés, notre projet ne se contente pas de proposer une solution technique, mais un nouveau paradigme pour la sécurité des infrastructures critiques.

L'intégration stratégique des services Hedera est au cœur de cette innovation. Le **Hedera Consensus Service (HCS)** n'est pas utilisé comme une simple base de données, mais comme une véritable "couche de confiance" applicative. Il transforme la journalisation des événements de sécurité, passant d'une archive interne et modifiable à un registre public, immuable et chronologiquement incontestable. Cette transparence vérifiable est essentielle pour les environnements multi-acteurs (gouvernements, opérateurs privés, auditeurs) où la confiance ne peut être présumée. Parallèlement, le **Hedera Token Service (HTS)** fournit le moteur d'une économie participative durable. En liant les récompenses à la performance et à la réputation des capteurs, nous créons un cercle vertueux qui incite à l'excellence et à la participation active, le tout géré de manière efficace et à faible coût grâce à Hedera.

La conception de l'architecture pour un déploiement sur **Amazon Web Services (AWS)** ancre cette vision dans une réalité de production. En prévoyant l'utilisation de services comme Amazon ECS pour la scalabilité, AWS Secrets Manager pour la sécurité des clés, et ElastiCache pour la gestion des sessions distribuées, SAHEL GUARD est pensé non seulement pour fonctionner, mais pour fonctionner de manière résiliente, sécurisée et à grande échelle. Cela démontre une feuille de route claire, allant du prototype à un service de niveau entreprise, prêt à être déployé dans des environnements exigeants.

En regardant vers l'avenir, le potentiel de SAHEL GUARD est immense. Les fondations sont posées pour évoluer vers une organisation autonome décentralisée (DAO), où les détenteurs de tokens pourraient gouverner le protocole en votant sur des propositions cruciales, telles que l'ajustement des seuils de l'IA ou l'allocation des fonds de la trésorerie. L'intégration de modèles d'apprentissage automatique plus sophistiqués et l'expansion du réseau de capteurs sont les prochaines étapes logiques de notre développement.

En synthèse, SAHEL GUARD se présente comme un témoignage puissant de la manière dont la technologie de registre public d'Hedera, associée à une infrastructure cloud robuste, peut être

exploitée pour résoudre des problèmes critiques du monde réel. Nous n'avons pas seulement construit une application de sécurité ; nous avons conçu un modèle pour l'avenir de la confiance et de la collaboration dans le domaine vital de la cybersécurité.