Cahier des charges SIEM MedTech en Rust

1- Présentation de l'entreprise

MedTech *est une entreprise créée en janvier 2026 par 3 personnes qui en sont aussi les seuls acteurs pour le moment : Clément Languedoc, Mehdi Abadi, et Johanna Meguedad.*

Notre projet est de mettre en place et vendre un SIEM (Security Information and Event Manager) que l'on développera en Rust. Nous savons que la concurrence est rude (Splunk, ELK etc.). Pour faire face à cela, notre stratégie est de vendre ***un produit de même qualité mais à prix largement inférieur.*** *En effet, nous n'aurons pas de prix au Go/jour, mais un prix fixe, peu importe le volume de journaux que notre SIEM devra traiter. Cette tarification rendra votre comptabilité beaucoup plus facile à gérer, ce qui permettra un gain de temps et d'argent. Nos cibles sont les TPE, les projets étudiants, les associations et* ***toute personne n'ayant pas le budget de FMN à mettre dans la sécurité de leur infrastructure.***

Notre SIEM sera développé en Rust, un langage moderne et connu pour sa gestion des 'undefined behavior', le rendant plus sécurisé.

Problématique actuelle dans les CERT/CSIRT :
Les logs sont dispersés, non corrélés et difficilement exploitables en cas d'incident. La pertinence de la reconstitution des logs est un élément crucial dans les investigations. Notre rôle est donc d'aider les équipes CSIRT/CERT à analyser les logs et les interpréter plus rapidement/facilement.

2- Objectifs du Projet

Développer un SIEM performant, sécurisé et modulaire en Rust permettant :

- La collecte centralisée des logs
- L'analyse et la corrélation des événements
- La détection d'anomalies et comportements suspects
- L'alerte en temps réel
- La visualisation via une interface web, avec possibilité d'administration par les équipes SOC.

3- Choix du langage de programmation

- Sécurité mémoire (pas de segmentation fault)
- Performance élevée
- Concurrence sécurisée (thread-safe)
- Binaire unique déployable
- Adapté aux systèmes bas niveau
- Testing facile à mettre en place

Périmètre Fonctionnel

4- Collecte des logs

Nous collecterons :

- Logs systèmes (linux/windows)
- Logs applications (API MedTech)
- Logs base de données
- Logs réseau (pare-feu, IDS)

Méthodes de collecte :

- Agent Rust
- Syslog
- API REST
- Fichiers locaux

5- Normalisation des logs

- Transformation en format JSON structuré
- Ajout de métadonnées tels que les timestamp, host, type d'event, criticité.

6- Moteur de corrélation

Fonctionnalités :

- Règles prédéfinies (ex : 5 échecs de login en 2 minutes)
- Corrélation multi-sources
- Détection brute force
- Détection élévation de privilèges (trouver des règles correspondantes)
- Détection exfiltration de données
- Détection ransomware (ex : extension de fichiers)
- Nous devrons utiliser la base de données MITRE ATTACK pour détecter des comportements suspects.

7- Détection statistique simple

- Analyse de fréquence
- Score de risque par événement

8- Système d'alertes

- Alertes en temps réel
- Niveaux : faible / moyen / critique

- Notification : email

9- Interface Web

Fonctionnalités :

- Dashboard en temps réel
- Visualisation des logs
- Filtres avancés
- Historique des alertes
- Gestion des règles
- Gestion des utilisateurs (RBAC – role based)
- Barre de recherche

Technologies :

- Backend : Rust (Actix Web ?)
- Frontend : React
- Gestion de projet : Git
- Gestion de tickets : issues Git

Architecture Technique

10- Architecture Globale

Composants :

1. Agent de collecte
2. Serveur d'ingestion
3. Pipeline de traitement
4. Base de données
5. Moteur de corrélation
6. API backend
7. Interface Web

10.1- Stockage

Base de données : PostgreSQL

10.2-Sécurité

- TLS
- Authentification JWT (token)
- Chiffrement des données sensibles
- Séparation des rôles : Admin, Analyste CERT

11- Contraintes Non Fonctionnelles

- Traitement minimum : 5 000 événements/seconde
- Latence alerte < 5 secondes

Sécurité

- Protection contre injections
- Protection contre DoS
- Logs immuables

Conformité

- Respect RGPD
- Journalisation traçable

Organisation du Projet (Équipe de 3 + timeline) – Arbitraire

Johanna – Backend & Ingestion - 1

- Agent Rust
- Collecte logs
- Parsing
- Normalisation

Mehdi – Moteur SIEM - 2

- Corrélation
- Règles
- Détection d'anomalies
- Système d'alertes

Clément – Interface & API - 3

- API REST
- Authentification
- Dashboard
- Gestion utilisateurs

Planning Prévisionnel

Phases	**Durée (semaines)**
Analyse & conception	**2**
Développement Agent	**3**
Développement moteur	**4**
Développement interface	**2**
Tests & validation	**2**
Documentation	**1**

Durée totale : 16 semaines (2 semaines de marge)

Livrables

1- Code source Rust documenté
2- Documentation technique
3- Manuel utilisateur
4- Rapport de sécurité
5- Démo fonctionnelle **(vidéo de sauvegarde au cas où)**

Risques Identifiés

Manque de temps : priorité aux fonctionnalités d'un SIEM basique

Évolutions Futures

- Intégration IA/ML
- SOAR (automatisation réponse)
- Intégration avec outils tiers
- Version cloud SaaS

Conclusion

Le projet SIEM de **MedTech** vise à :

- Centraliser la sécurité
- Protéger les données
- Améliorer la détection d'incidents
- Renforcer la résilience cyber

Le choix de Rust garantit performance, robustesse et sécurité, éléments critiques pour le secteur médical.