
CAHIER DES CHARGES DU MINI PROJET

Groupe	Classe
Etudiant 1 :Rym Ben Attia	4ING SSRI B
Etudiant 2 : Ahmed Ben Slimane	
Etudiant 3 :	

TITRE DE PROJET : Systemes and Network monitoring : Mini SIEM de détection d'intrusions

1. INTRODUCTION GENERALE ET CADRE DU MINI PROJET :

ce mini-projet consiste à développer un prototype fonctionnel d'un **SIEM simplifié** (Security Information and Event Management).

Le système doit être capable de :

- collecter des alertes générées par l'IDS Snort,
- les analyser et les enrichir avec des informations supplémentaires,
- détecter des comportements suspects en regroupant plusieurs alertes (corrélation),
- enregistrer toutes les données dans une base SQLite,
- et afficher les résultats sur une **interface web minimaliste** développée en Python (Flask).

2. OBJECTIFS :

Objectif général :

Développer un prototype d'outil de détection et d'analyse d'alertes réseau en utilisant **Snort** comme IDS et **Python** pour le traitement et l'affichage.

Objectifs spécifiques :

- Configurer Snort pour générer des alertes dans un fichier.
- Créer un agent Python capable de lire les alertes Snort en temps réel.
- Normaliser les alertes dans un format structuré.
- Enrichir les alertes (géolocalisation IP, ASN...).

- Mettre en place des règles simples de corrélation pour détecter des attaques.
- Enregistrer les alertes dans une base de données SQLite.
- Développer une interface web minimaliste pour visualiser les alertes.
- Tester et valider le système avec des scans réseau (Nmap, Scapy...).

3. Besoins fonctionnels :

3.1 Collecte des alertes

- Le système doit lire automatiquement les alertes écrites par Snort dans :
`/var/log/snort/alert_fast.log`
- Le système doit détecter l'arrivée de nouvelles lignes.

3.2 Normalisation

- Le système doit transformer chaque alerte Snort en dictionnaire Python standardisé : .csv

3.3 Enrichissement

- Le système doit récupérer des informations supplémentaires :
 - Pays
 - Ville
 - ASN
 - Organisation (Whois)
- Ces informations doivent être ajoutées dans la base de données.

3.4 Stockage

- Les alertes doivent être enregistrées dans la base SQLite.
- La table `alerts` doit contenir :
 - ID
 - Signature
 - IP Source
 - IP Destination
 - Gravité
 - Protocole

- Données d'enrichissement
- Timestamp automatique

3.5 Corrélation

- Le système doit analyser les alertes stockées pour détecter :
 - Plusieurs alertes depuis la même IP en moins de X minutes
 - Plusieurs signatures différentes depuis la même IP
- Le système doit afficher un message d'attaque dans la console (ou enregistrer un flag).

3.6 Interface Web

- L'interface doit permettre d'afficher les 50 dernières alertes.
- L'utilisateur doit voir :
 - alerte
 - IP source
 - pays
 - ASN
 - gravité
 - heure
- Les alertes doivent être colorées selon la gravité.

BESOINS NON FONCTIONNELS

Simplicité

- L'interface doit être minimaliste, légère et facile à comprendre.

Performance

- Le système doit ingérer au moins 50 alertes/min sans ralentissement visible.

Sécurité

- Le projet doit fonctionner uniquement en environnement de test contrôlé.
- Aucun test d'attaque ne doit être effectué sur un réseau public.

Maintenabilité

- Le code doit être clair, commenté, et organisé.

Portabilité

- Le système doit fonctionner sur n'importe quelle distribution Linux simple (Ubuntu/Debian).

3. Environnements de développement :

ENVIRONNEMENT MATERIEL

PC ou VM avec au minimum :

- CPU : Dual core
- RAM : 4GB
- Stockage : 10GB

ENVIRONNEMENT LOGICIEL

- **Linux Ubuntu/Debian**
- **Python 3.10+**
- **Snort 2.x ou 3.x**
- Bibliothèques Python :
 - Flask
 - geoip2
 - ipwhois
 - sqlite3
 - scapy

Diagramme de cas d'utilisation :

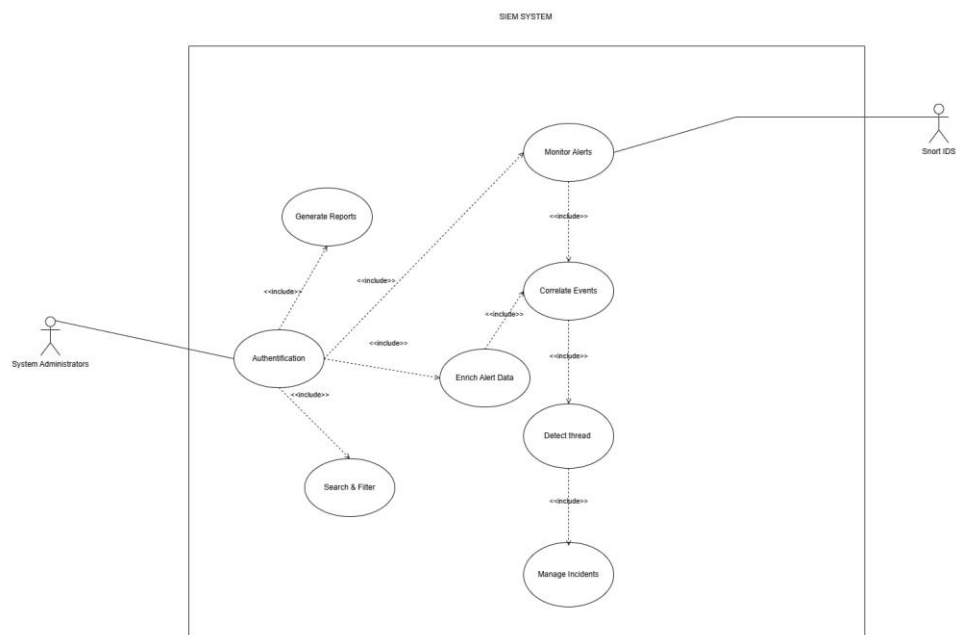


Diagramme de classe:

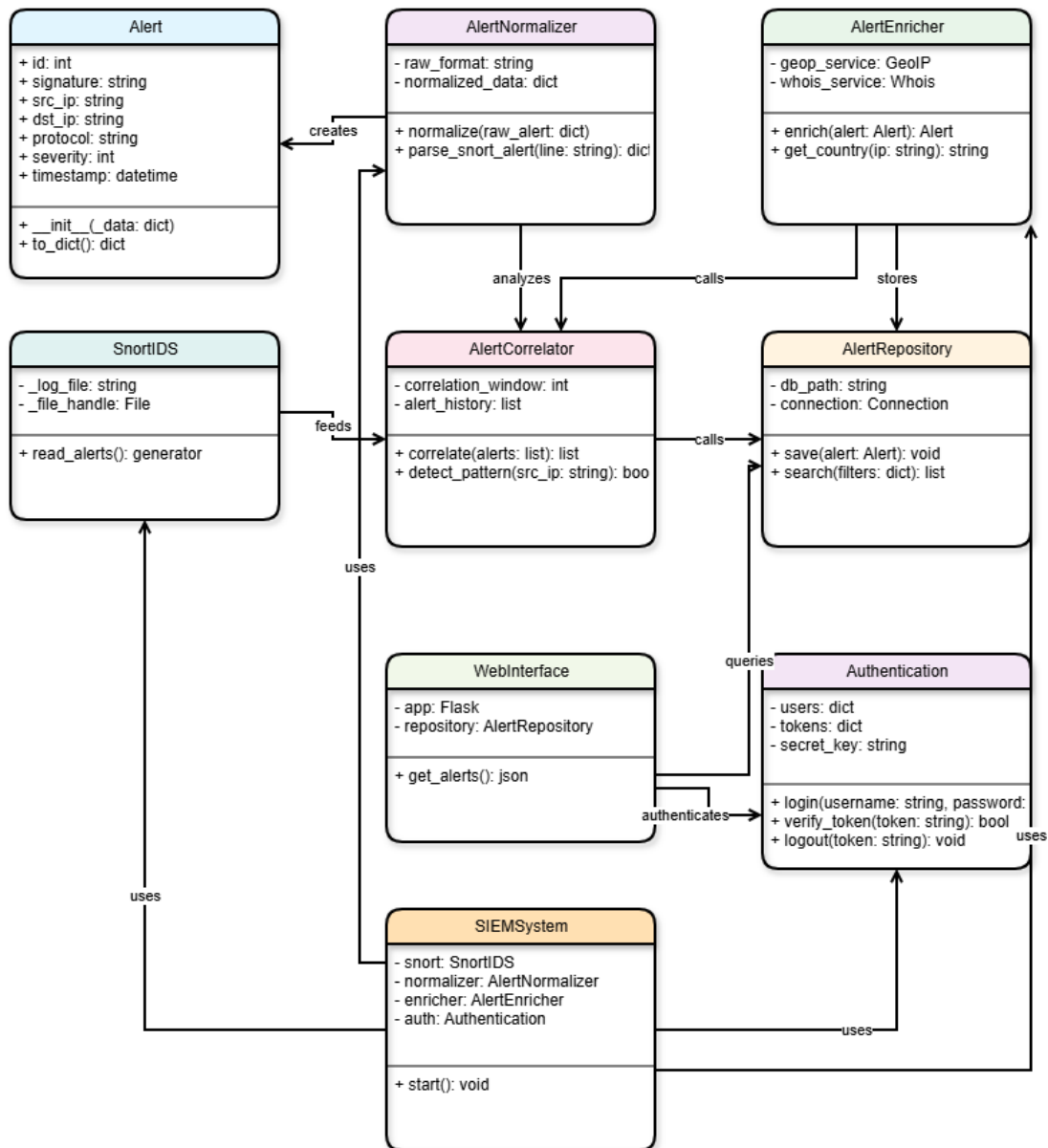
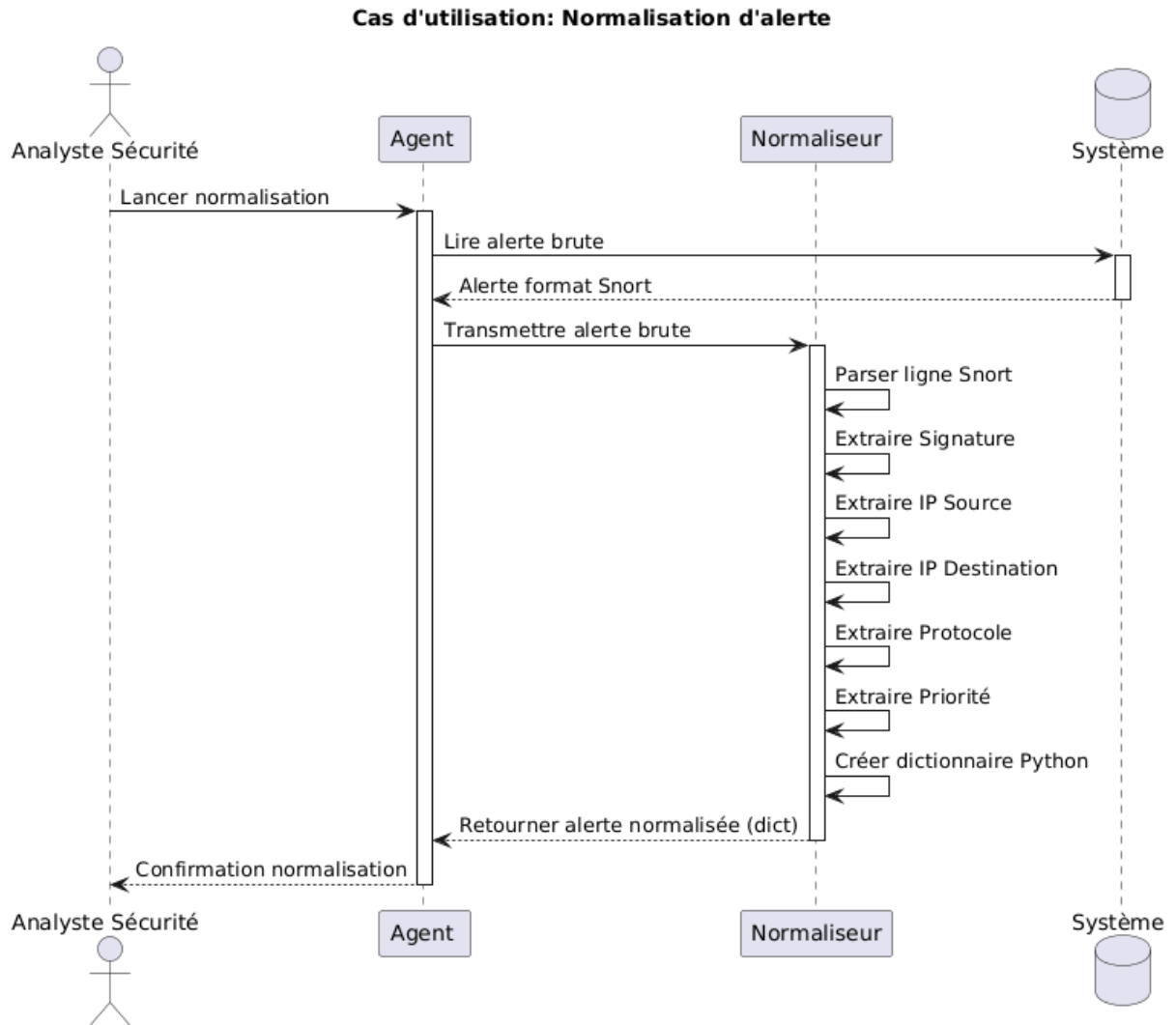


Diagramme de sequence :



Cas d'utilisation: Corrélation d'alertes

