Mini-Projet Python – "Security Orchestrator"

Contexte général

Dans un environnement de sécurité moderne, les analystes sont submergés par le volume d'alertes générées par les systèmes (échecs de connexion SSH, erreurs HTTP répétées, etc.). Le temps de réponse est critique.

Ce mini-projet vous met dans la peau d'une équipe de développement chargée de créer un outil léger d'Orchestration et de Réponse à Incident (SOAR Light). Votre outil doit non seulement analyser les logs de manière exhaustive, mais surtout prendre une décision automatisée (déclencher une investigation réseau ciblée) basée sur un score de suspicion, transformant l'analyse passive en une action proactive.

Objectifs pédagogiques

Ce projet vise à vous faire monter en compétence sur :

Axe de Compétence	Objectif Spécifique
Intégration & Architecture	Structurer un projet Python en **modules interdépendants** (`log_parser → data_analyzer → network_scanner`).
Décision Automatisée	Mettre en place une **logique de seuil** pour qualifier une IP de "suspecte" et **déclencher une action**.
Performance	Implémenter le **multithreading** pour accélérer le scan réseau.
Interface Utilisateur	Utiliser le module `argparse` pour gérer l'ensemble du workflow via une **ligne de commande unique**
Analyse Multilog	Fusionner et corréler des données provenant de **plusieurs types de logs** (HTTP et SSH).

Consignes de réalisation

Organisation du Projet

- ✓ **Équipes** : Le projet sera réalisé en groupes de 3 apprentis.
- ✓ **Données** : Des exemples de fichiers `access.log` (Apache) et `auth.log` (SSH) vous seront fournis.

Fonctionnalités à Implémenter (Le Workflow)

Le script principal (`main.py`) doit orchestrer le workflow en trois étapes :

Étape 1 : Log Aggregation & Scoring (**Modules** log_parser.py **et** data_analyzer.py)

1. **Parsing Multi-Log :**
   - Extraire les adresses IP responsables d'

     erreurs 404 (HTTP).
   - Extraire les adresses IP responsables d'

     échecs de connexion (SSH, "Failed password").
2. **Détection Avancée :** Identifier les IPs dont le `user_agent` dans les logs HTTP contient des mots-clés de bots (bot, crawler, spider).
3. **Calcul du Score de Suspicion** :
   - Créer un tableau de bord unique des IPs en cumulant les occurrences de chaque comportement (404, échec SSH, bot).
   - Définir et appliquer un seuil de détection (ex : IP avec plus de 50 échecs OU plus de 10 requêtes bot ET 404) pour générer la liste des IPs Hautement Suspectes qui passeront à l'Étape 2.

Étape 2 : Automated Network Scan (**Module** `network_scanner.py`)

Cette étape doit être **automatiquement déclenchée** pour **uniquement** les IPs listées comme "Hautement Suspectes" à l'Étape 1.

1. **Mini-Scanner de Ports :** Utiliser le module socket pour tenter une connexion sur une plage de ports critiques (ex: 20-23, 80, 443, 3389).
2. **Performance :** Implémenter une **version multithreadée** pour accélérer la vérification des ports pour chaque IP.
3. **Résultats Ciblés :** Identifier et enregistrer uniquement les **ports ouverts**.

Étape 3 : Reporting et Visualisation (**Modules** `data_analyzer.py` **et** `main.py`)

1. **Visualisation :** Générer un **histogramme** (bar chart) avec `matplotlib` des **Top 5 des IPs** basé sur leur score de suspicion combiné (Étape 1).
2. **Rapport Final :** Exporter un fichier de synthèse (`rapport_securite.csv`) incluant :
   - L'**IP**
   - Le **Score de Suspicion** (total des incidents)
   - La **Raison Principale** (SSH Failed/404/Bot)
   - La liste des **Ports Ouverts** trouvés lors du scan.

Livrables Attendus

Les livrables suivants devront être soumis :

1. **Code Source Complet** : L'ensemble des scripts Python organisés en modules (`main.py, log_parser.py, data_analyzer.py, network_scanner.py`) .
2. **Jeu de Données** : Le fichier CSV ou texte du Rapport Final généré (`rapport_securite.csv` ou similaire).
3. **Graphique** : Le fichier image de la visualisation (PNG ou JPG) du Top 5 des IPs