Mini-Projet Python – Analyse de sécurité & scanner réseau

Contexte général

Dans le contexte d'un environnement sécurisé, il est crucial d'identifier et de comprendre les comportements suspects dans les systèmes d'information. Les tentatives de connexion non autorisée, les accès répétés aux ressources interdites, ou la détection de ports ouverts sur des machines cibles sont des exemples d'indicateurs précoces d'activité malveillante.

Ce mini-projet vous met dans la peau d'un analyste sécurité chargé :

- ✓ d'extraire et de traiter des fichiers logs système (authentification, réseau),
- ✓ de repérer les adresses IP suspectes,
- ✓ d'analyser et visualiser les résultats,
- ✓ et de vérifier si certaines adresses IP répondent sur le réseau (scan de ports).

Le projet se construit progressivement à partir du Jour 2 jusqu'au Jour 4, à travers les différentes thématiques abordées.

Objectifs pédagogiques

- ✓ Mettre en œuvre les connaissances Python autour de la manipulation de fichiers, de données, de la visualisation, et du réseau.
- ✓ Comprendre comment les outils de base permettent d'analyser des comportements suspects.
- ✓ Développer un outil simple d'aide à la détection d'activité malveillante.
- ✓ Apprendre à structurer un projet Python avec plusieurs modules.

Consignes de réalisation

Partie 1 – Analyse de logs

- ✓ Lire un fichier de logs Linux ou Windows (ex : auth.log ou export Security.evtx).
- ✓ Extraire les lignes pertinentes (échecs de connexion, 404, etc.).
- ✓ Isoler les adresses IP sources.
- ✓ Enregistrer les résultats sous forme de dictionnaire ou tableau (IP → nombre d'occurrences).

Partie 2 – Traitement & Visualisation

- ✓ Convertir les données en DataFrame avec pandas.
- ✓ Trier les IP les plus actives (Top 5).
- ✓ Générer un bar chart (ou autre graphique pertinent) avec matplotlib.pyplot.
- ✓ Bonus : détecter des bots ou scanners via le champ User-Agent ou le rythme des requêtes.

Partie 3 – Scan de ports réseau

- ✓ Pour les IP suspectes, utiliser socket pour tenter une connexion sur des ports connus (22, 80, 443…).

- ✓ Implémenter un mini scanner de ports (mono-thread, puis multithread).
- ✓ Identifier les ports ouverts (utiliser connect_ex()).
- ✓ Bonus : ajouter une option --verbose pour afficher les ports fermés.

Fonctionnalités à implémenter

Obligatoires :

- ✓ Script principal main.py ou notebook.ipynb
- ✓ Traitement de logs et extraction IP
- ✓ Analyse statistique (nombre d'occurrences par IP)
- ✓ Visualisation graphique des résultats
- ✓ Scanner simple de ports (avec socket)

Bonus :

- ✓ Version multithreadée du scanner
- ✓ Export des résultats au format CSV ou HTML
- ✓ Interface ligne de commande avec argparse
- ✓ Interface simple (menu texte ou CLI interactive)

Livrables attendus

1. Code source du projet (.py ou .ipynb), organisé par modules si nécessaire :
    - ✓ log_parser.py
    - ✓ data_analyzer.py
    - ✓ network_scanner.py
    - ✓ main.py ou scan_demo.ipynb
2. Jeu de données de test (fichier .log fourni ou généré)
3. Rapport ou présentation courte expliquant :
    - ✓ Les étapes techniques (découpage du projet)
    - ✓ Les choix techniques effectués
    - ✓ Des exemples de résultats (visualisation, IP suspectes, ports ouverts)
4. Fichier CSV ou graphique de visualisation généré (Top IPs, ports ouverts, etc.)