Ce rapport détaille la conception et la réalisation de notre projet python ou l’on a mis en place des outils d'analyse de sécurité, en se concentrant sur les étapes techniques, le découpage du projet et les choix techniques sous-jacents. Il intègre des capacités de parsing de journaux, d'analyse de données et de scan de ports réseau. L'objectif principal est d'identifier des activités suspectes à partir de fichiers journaux système (comme auth.log pour Linux) et de permettre une investigation plus poussée via des scans réseau ciblés.

**Découpage du projet et étapes techniques**

Le projet a été découpé en plusieurs modules Python, chacun ayant une responsabilité spécifique. Cela permet de faciliter la maintenance, la testabilité et l'évolutivité.

**a.** Module **main.py**

* **Rôle :** C'est le point d'entrée de l'application, gérant l'interface utilisateur via un menu interactif et orchestrant les appels aux autres modules.
* **Étapes Techniques :**
  + Initialisation des répertoires : Au démarrage (if \_\_name\_\_ == '\_\_main\_\_':), il s'assure que les dossiers logs, graphs et exports existent, ce qui est crucial pour le stockage des fichiers d'entrée et de sortie.
  + Boucle de Menu : Une boucle while True affiche les options principales (analyser les logs, visualiser les données, scanner les ports, quitter) et gère les choix de l'utilisateur.
  + Gestion de l'État : Des variables comme all\_extracted\_ips, all\_pertinent\_log\_lines, et top\_n\_ips\_for\_scan sont maintenues pour passer des données entre les différentes étapes de l'analyse.
  + Appels Modulaires : Il importe et appelle les fonctions spécifiques des autres modules (parse\_linux\_auth\_log, analyze\_and\_visualize\_ips, simple\_port\_scan, multi\_threaded\_port\_scan, detect\_bots\_or\_scanners).

**b.** Module **log\_parser.py**

* **Rôle :** Extraire des informations pertinentes, notamment des adresses IP, à partir de fichiers journaux Linux, en se concentrant sur les événements d'authentification suspects.
* **Choix Techniques :**
  + re (RegEx) : L'utilisation du module re pour le parsing de logs à été utilisé, car les journaux sont des données textuelles non structurées. Des expressions régulières spécifiques (suspicious\_sshd\_patterns) sont définies pour cibler précisément les tentatives d'authentification échouées, les utilisateurs invalides, et les déconnexions suspectes (particulièrement dans la phase [preauth]). Ce choix offre une grande flexibilité pour adapter les patterns à différents types d'événements ou de formats de logs.
  + collections.defaultdict(int) **:** L'utilisation de defaultdict pour ip\_occurrences simplifie le comptage des occurrences d'IP, évitant les vérifications if ip not in dict: avant d'incrémenter.
  + Gestion des Erreurs : Des blocs try-except sont utilisés pour gérer les cas où le fichier journal n'est pas trouvé (FileNotFoundError) ou d'autres exceptions générales lors de la lecture, rendant le script plus robuste.

**c.** Module **data\_analyzer.py**

* **Rôle :** Traiter les adresses IP extraites des logs pour en déduire des statistiques (comme les IPs les plus actives) et les visualiser sous forme de graphiques. Il inclut également une fonction de détection de bots.
* **Choix Techniques :**
  + pandas **:** L'utilisation de pandas pour la manipulation des données (DataFrame) permet de trier, filtrer et agréger les données IP de manière efficace, facilitant l'identification des "Top N" IPs.
  + matplotlib.pyplot **:** Ce module est standard pour la création de graphiques statiques en Python. Nous l’utilisons pour générer un graphique à barres des IPs les plus actives, fournissant une visualisation claire des menaces potentielles.
  + Détection de Bots/Scanners : Une approche basée sur des signatures de User-Agent est implémentée. Elle permet de détecter les scanners courants et les robots qui s'identifient via leur chaîne User-Agent dans les logs.
  + Gestion des Fichiers : La sauvegarde des graphiques dans un répertoire dédié (graphs) et la vérification de son existence (os.makedirs) contribuent à l'organisation du projet.

**d.** Module **network\_scanner.py**

* **Rôle :** Réaliser des scans de ports sur des adresses IP ciblées pour identifier les ports ouverts. Il offre des options mono-thread et multi-thread.
* **Choix Techniques :**
  + socket **:** Le module socket de Python permet la connexion réseau. socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) crée un socket TCP/IP, et connect\_ex est utilisé pour des scans non bloquants, pour un scanner de ports.
  + threading **et** queue.Queue **:** Pour le scan multi-threadé, l'utilisation du module threading avec une Queue gère la liste des tâches (IP, port) à scanner. Cela améliore considérablement la vitesse de scan par rapport à une approche mono-thread, en particulier pour un grand nombre d'IPs et de ports.
  + COMMON\_PORTS **:** La définition d'une liste de ports communs prédéfinis permet un scan rapide et ciblé sur les services les plus fréquemment exposés.
  + timeout **:** L'ajout d'un timeout pour les connexions (sock.settimeout(timeout)) permet d’éviter que le scanner ne se bloque sur des hôtes non réactifs ou des ports filtrés.
  + **Verbosité :** L'option verbose permet à l'utilisateur de choisir s'il souhaite voir tous les ports scannés (ouverts et fermés) ou seulement les ports ouverts, offrant plus de flexibilité.
  + export\_scan\_results\_to\_csv **:** Cette fonction utilise pandas pour structurer les résultats du scan et les exporter facilement dans un format CSV, ce qui est pratique pour le reporting et l'intégration avec d'autres outils.

En conclusionl'application de concepts de programmation Python (modularité, gestion des threads, manipulation de données avec pandas, expressions régulières) est un cas d'usage concret en cybersécurité. Ce projet met en avant l’analyse de sécurité et des outils Python pour l’application dans le domaine des réseaux et systèmes.

**Captures d’écran des résultats**

Capture du menu principale :

**Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.**

Capture de l’exécution du choix 1 avec l’analyse des logs avec le fichier auth.log :

**Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.**

Capture de l’exécution du choix 2 avec traitement et visualisation des données, mise en forme dataframe, top des IP les plus actives, génération de graphique et détection de bot :

**Une image contenant texte, capture d’écran

Le contenu généré par l’IA peut être incorrect.**

**Une image contenant texte, capture d’écran, diagramme, Tracé

Le contenu généré par l’IA peut être incorrect.**

Capture de l’exécution du choix 3 avec scan des ports réseaux (ouvert ou fermés) mono et multithread et génération d’un fichier CSV des résultats :

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

**Avis sur le cours (de Farès et Adrien)**

Nous avons apprécié l'organisation de ce cours. La matinée dédiée à la théorie et aux TP nous préparait pour le projet de l'après-midi, où nous mettions en pratique les concepts appris. Les travaux pratiques et les projets étaient intéressants, car ils nous permettaient d'utiliser des bibliothèques (re, stocket, pandas, csv …) en lien direct avec les réseaux et les systèmes.