

Ce rapport détaille la conception et la réalisation de notre projet python où 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 conclusion l'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 :

```
PS C:\Users\Fares\Desktop\M2\Projet PYTHON\Projet> & C:/Users/Fares/AppData/Local/Programs/Python/Python39-64/Scripts/python.exe C:/Users/Fares/Desktop/M2/Projet PYTHON/Projet/main.py

--- Menu Principal ---
1. Analyser les logs Linux (auth.log)
2. Traiter et visualiser les données (Analyse statistique et Graphique)
3. Scanner les ports réseau
4. Quitter
Choisissez une option:
```

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

```
Choisissez une option: 1

--- Analyse de Logs Linux (auth.log) ---
Parsing du fichier de log Linux: logs/auth.log
Parsing des logs Linux (événements SSHD suspects) terminé. 24 lignes pertinentes trouvées.
IPs extraites de logs/auth.log: {'198.51.100.1': 2, '198.51.100.2': 1, '203.0.113.45': 1, '203.0.113.46': 2, '198.51.100.3': 3, '203.0.113.50': 1, '198.51.100.4': 1, '203.0.113.55': 1, '198.51.100.6': 1, '203.0.113.60': 2, '198.51.100.7': 2, '203.0.113.70': 2, '103.208.220.101': 2, '218.92.0.31': 1, '198.51.100.80': 1, '198.51.100.81': 1}

Total des IPs collectées jusqu'à présent: 24 occurrences pour 16 IPs uniques.

--- Lignes de logs pertinentes trouvées ---
- Jun 3 08:17:05 webserver01 sshd[1025]: Failed password for useralpha from 198.51.100.1 port 11233 ssh2
- Jun 3 08:17:09 webserver01 sshd[1025]: Failed password for useralpha from 198.51.100.1 port 11235 ssh2
- Jun 3 08:17:15 webserver01 sshd[1028]: Failed password for invalid user nouser from 198.51.100.2 port 22345 ssh2
- Jun 3 08:18:01 db-primary sshd[1030]: Received disconnect from 203.0.113.45 port 33456:11: Bye Bye [preauth]
- Jun 3 08:18:02 db-primary sshd[1031]: Connection closed by 203.0.113.46 port 44567 [preauth]
- Jun 3 08:18:03 db-primary sshd[1032]: Connection closed by 203.0.113.46 [preauth]
- Jun 3 08:20:30 localhost sshd[1040]: Failed password for root from 198.51.100.3 port 12897 ssh2
- Jun 3 08:20:33 localhost sshd[1040]: Failed password for root from 198.51.100.3 port 12899 ssh2
- Jun 3 08:20:36 localhost sshd[1040]: Failed password for root from 198.51.100.3 port 12901 ssh2
- Jun 3 08:21:00 webserver01 sshd[1044]: Disconnected from authenticating user testuser 203.0.113.50 port 55678 [preauth]
- Jun 3 08:22:05 db-primary sshd[1048]: User restricted from 198.51.100.4 port 23901 ssh2: User restricted from 198.51.100.4 not allowed because not listed in AllowUsers
- Jun 3 08:24:45 webserver01 sshd[1058]: User olduser from 203.0.113.55 port 45923 ssh2: User olduser from 203.0.113.55 not allowed because password expired
- Jun 3 08:25:50 db-primary sshd[1062]: User locked from 198.51.100.6 port 56934 ssh2: User locked from 198.51.100.6 not allowed because account is locked
- Jun 3 08:26:10 localhost sshd[1065]: Invalid user hackerman from 203.0.113.60 port 12321 ssh2
- Jun 3 08:26:12 localhost sshd[1066]: Failed password for invalid user hackerman from 203.0.113.60 port 12323 ssh2
- Jun 3 08:27:00 webserver01 sshd[1070]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=198.51.100.7 user=attacker
- Jun 3 08:27:05 webserver01 sshd[1071]: Failed password for attacker from 198.51.100.7 port 23432 ssh2
- Jun 3 08:28:01 db-primary sshd[1076]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=203.0.113.70
- Jun 3 08:28:02 db-primary sshd[1077]: Failed password for invalid user serviceacc from 203.0.113.70 port 33445 ssh2
- Jun 3 10:00:00 mal-serv sshd[3001]: Failed password for root from 103.208.220.101 port 12345 ssh2
- Jun 3 10:00:05 mal-serv sshd[3002]: Failed password for admin from 218.92.0.31 port 22222 ssh2
- Jun 3 10:00:10 mal-serv sshd[3003]: Failed password for user from 103.208.220.101 port 34567 ssh2
- Jun 3 11:06:00 webserver01 sshd[1200]: error: Received disconnect from 198.51.100.80 port 50123:3: com.jcraft.jsch.JSChException: Auth fail [preauth]
- Jun 3 11:06:05 db-primary sshd[1201]: Connection closed by 198.51.100.81 port 50124 [preauth]
```

Rapport Projet Python - IRMAL Farès et SPROCQ Adrien

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 :

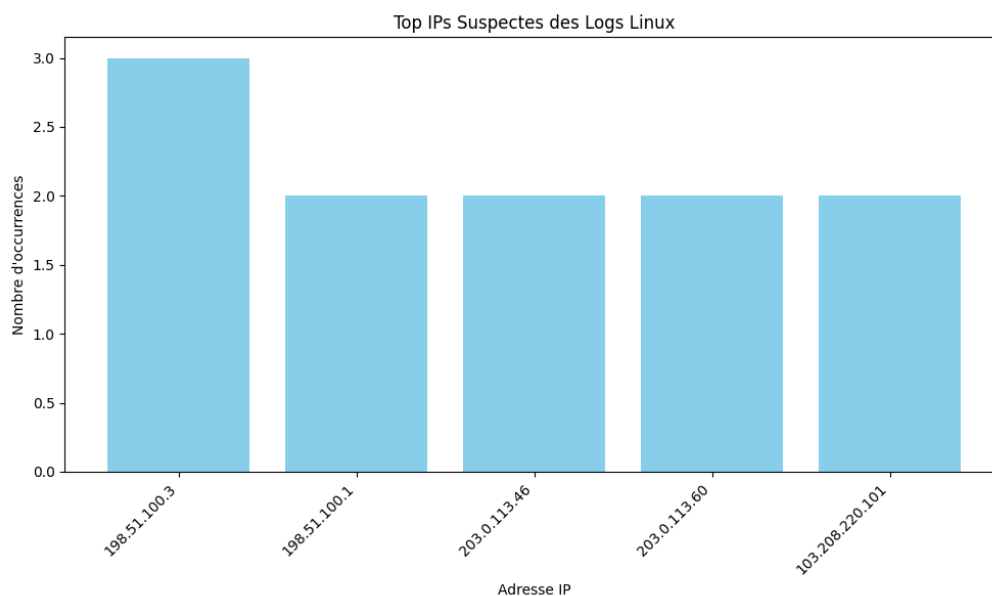
```
Choisissez une option: 2
--- Traitement & Visualisation (Analyse des données) ---
Analyse et visualisation des données IP...

DataFrame des IPs et occurrences:
  IP Address  Occurrences
4   198.51.100.3           3
0   198.51.100.1           2
3   203.0.113.46           2
9   203.0.113.60           2
12  103.208.220.101         2
11  203.0.113.70           2
10  198.51.100.7           2
1   198.51.100.2           1
7   203.0.113.55           1
6   198.51.100.4           1
5   203.0.113.50           1
2   203.0.113.45           1
8   198.51.100.6           1
13  218.92.0.31            1
14  198.51.100.80          1
15  198.51.100.81          1

Top 5 des IPs les plus actives:
  IP Address  Occurrences
4   198.51.100.3           3
0   198.51.100.1           2
3   203.0.113.46           2
9   203.0.113.60           2
12  103.208.220.101         2

Graphique enregistré sous: graphs\top_ips_chart.png

Tentative de détection de bots/scanners dans logs/auth.log (basé sur User-Agent)...
IPs suspectes détectées via User-Agent: ['103.208.220.101', '45.137.21.101', '45.137.21.100', '198.51.100.1']
```



Rapport Projet Python - IRMAL Farès et SPROCQ Adrien

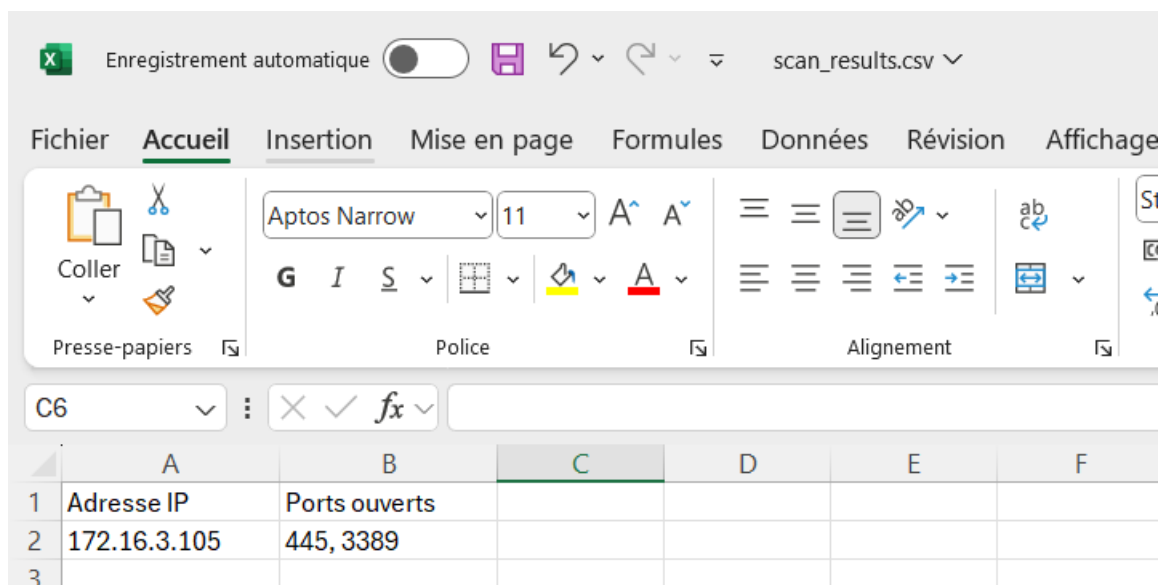
Capture de l'exécution du choix 3 avec scan des ports réseaux (ouvert ou fermés) mono et multithread et génération d'un fichier CSV des résultats :

```
Choisissez une option: 3

--- Scan de Ports Réseau ---
IPs suspectes détectées (pour le scan): ['103.208.220.101', '45.137.21.101', '45.137.21.100', '198.51.100.1']
Voulez-vous scanner ces IPs suspectes? (oui/non): non
Entrez des IPs à scanner (séparées par des virgules, ex: 127.0.0.1,8.8.8.8) ou laissez vide pour annuler: 172.16.3.105
Quel type de scan voulez-vous ? (mono-thread / multi-thread): multi-thread
Voulez-vous afficher les ports fermés (--verbose) ? (oui/non): oui
Entrez le nombre de threads (par défaut: 10):

Scan de ports multi-thread (avec 10 threads) pour les IPs: ['172.16.3.105']...
Port 139 est OUVERT sur 172.16.3.105
Port 135 est OUVERT sur 172.16.3.105
Port 445 est OUVERT sur 172.16.3.105
Port 3389 est OUVERT sur 172.16.3.105
Port 23 est FERMÉ sur 172.16.3.105
Port 21 est FERMÉ sur 172.16.3.105
Port 22 est FERMÉ sur 172.16.3.105
Port 25 est FERMÉ sur 172.16.3.105
Port 80 est FERMÉ sur 172.16.3.105
Port 443 est FERMÉ sur 172.16.3.105
Port 110 est FERMÉ sur 172.16.3.105
Port 53 est FERMÉ sur 172.16.3.105
Port 8080 est FERMÉ sur 172.16.3.105
Scan multi-thread terminé en 1.03 secondes.

--- Résultat du Scan de Ports ---
IP: 172.16.3.105 -> Ports ouverts: [135, 139, 445, 3389]
Voulez-vous exporter les résultats du scan au format CSV ? (oui/non): oui
Résultats exportés en CSV vers exports\scan_results.csv
```



The screenshot shows the Microsoft Excel interface with the 'Accueil' (Home) ribbon selected. The file name is 'scan_results.csv'. The table below contains the scan results:

	A	B	C	D	E	F
1	Adresse IP	Ports ouverts				
2	172.16.3.105	445, 3389				
3						

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, socket, pandas, csv ...) en lien direct avec les réseaux et les systèmes.