Documentation Technique - Toolbox de Cybersécurité

Table des matières

Introduction

La Toolbox de Cybersécurité est une suite d'outils intégrés permettant d'effectuer diverses tâches liées à la sécurité informatique. Cette plateforme regroupe plusieurs outils populaires tels que Metasploit, Nmap, OWASP ZAP, WPScan, Gobuster, TCPdump, SQLmap, Hydra, Nikto, John the Ripper, TheHarvester et Subfinder dans une interface web unifiée.

Dépôt GitHub: https://github.com/Amineb-sio/Toolbox

Prérequis

- Système d'exploitation compatible (Kali Linux recommandé)
- Python 3.13 ou supérieur
- Poetry (gestionnaire de dépendances Python)
- Docker et Docker Compose
- Connexion Internet pour télécharger les dépendances et les images Docker

Installation

1. Cloner le dépôt

bash

```
git clone https://github.com/Amineb-sio/Toolbox.git
cd Toolbox
```

2. Installation des dépendances Python

La Toolbox utilise Poetry pour gérer ses dépendances. Pour installer toutes les dépendances nécessaires, exécutez la commande suivante dans le répertoire principal de la Toolbox:

```bash
poetry install
```

Cette commande va créer un environnement virtuel et installer toutes les dépendances définies dans le fichier `pyproject.toml`, notamment:

- Flask et ses extensions
- Requests
- SQLAlchemy
- Cryptographie
- Outils d'analyse réseau
- Bibliothèques de reporting
- Et d'autres dépendances spécifiques

L'installation peut prendre quelques minutes et vous verrez la progression de l'installation de chaque package comme illustré dans les logs.

3. Déploiement des services Docker

La Toolbox s'appuie sur plusieurs services Docker pour fonctionner correctement. Pour démarrer ces services, exécutez:

```bash
docker-compose up -d
```

Cette commande va télécharger et démarrer les conteneurs suivants:

- **Portainer**: Interface de gestion Docker (accessible sur le port 9000)
- **Keycloak**: Service d'authentification (accessible sur le port 8080)
- **PostgreSQL**: Base de données (port 5432)
- **pgAdmin**: Interface d'administration PostgreSQL (port 5050)

Le système téléchargera les images suivantes:

- portainer/portainer-ce
- quay.io/keycloak/keycloak:24.0.1
- postgres:15-alpine

- dpage/pgadmin4:latest

Configuration des services

Keycloak

Keycloak est configuré pour gérer l'authentification des utilisateurs. La Toolbox est préconfigurée pour se connecter à l'instance Keycloak sur http://localhost:8080.

Par défaut, vous pourrez vous connecter avec le compte administrateur configuré dans Keycloak.

Base de données PostgreSQL

La base de données PostgreSQL est utilisée pour stocker les données de l'application. Les identifiants de connexion par défaut sont définis dans le fichier docker-compose.yml.

Démarrage de l'application

Pour démarrer tous les services de la Toolbox, utilisez le script de démarrage fourni:

```bash
poetry run bash ./start_all.sh
```

Ce script va:

1. Démarrer tous les microservices Python en arrière-plan
2. Initialiser le système de journalisation
3. Démarrer l'application principale sur http://127.0.0.1:5000

Services démarrés par le script

- Metasploit Web Interface
- Nmap Scanner
- OWASP ZAP
- WPScan
- Gobuster
- TCPdump Analyzer
- SQLmap
- Hydra
- Nikto
- John the Ripper
- TheHarvester

- Subfinder
- Auto-Sécurité

Une fois démarré, vous verrez des logs similaires à ceux-ci:

```
Lancement de /home/kali/Desktop/Toolbox/Python metasploit Webmin/app.py en arrière-plan...
Lancement de /home/kali/Desktop/Toolbox/Python nmap/app.py en arrière-plan...
Lancement de /home/kali/Desktop/Toolbox/Python owasp/app.py en arrière-plan...
...
```

Architecture de la solution

La Toolbox suit une architecture microservices où chaque outil est encapsulé dans son propre service Python. L'application principale (main.py) sert de point d'entrée et de coordination entre ces différents services.

Stockage des données

- **Sessions**: /tmp/tmp[random]
- **Sauvegardes**: /[chemin_installation]/backups
- **Clés cryptographiques**: /[chemin_installation]/secure_keys
- **Logs**: /[chemin_installation]/logs
- **URL de base pour les redirections**: http://127.0.0.1
- **Configuration Keycloak**: http://localhost:8080

Composants et services

Services principaux

- **Application Web Flask**: Interface utilisateur accessible sur http://127.0.0.1:5000
- **Keycloak**: Gestion des identités et des accès sur http://localhost:8080
- **PostgreSQL**: Base de données persistante
- **Portainer**: Gestion des conteneurs Docker sur http://localhost:9000

Microservices d'outils de sécurité

Chaque outil est implémenté comme un microservice Python indépendant qui expose une API REST pour l'interaction avec l'application principale.

Dépannage

Logs

Les logs de l'application sont stockés dans le répertoire `/[chemin_installation]/logs` et suivent un format de nommage `toolbox_YYYY-MM-DD.log`.

Exemple de logs:

```
05-20 03:34:47 - I - [logging_system:166] - ========================================
05-20 03:34:47 - I - [logging_system:167] - Démarrage Toolbox: 2025-05-20 03:34:47
05-20 03:34:47 - I - [logging_system:168] - Log level: INFO
05-20 03:34:47 - I - [logging_system:169] - Log file:
/home/kali/Desktop/Toolbox/logs/toolbox_2025-05-20.log
```

Problèmes courants

- **Service non disponible**: Vérifiez que tous les conteneurs Docker sont en cours d'exécution avec `docker ps`
- **Erreur d'authentification**: Assurez-vous que Keycloak est correctement configuré et accessible
- **Microservice non démarré**: Vérifiez les logs spécifiques du service dans le répertoire des logs

Pour redémarrer tous les services:

bash

```bash
docker-compose down
docker-compose up -d
poetry run bash ./start_all.sh
```

Nettoyage des conteneurs Docker

Si vous avez besoin de nettoyer votre environnement Docker, vous pouvez utiliser les commandes suivantes pour arrêter et supprimer tous les conteneurs et images:

bash

```bash
# Arrêter tous les conteneurs
docker stop $(docker ps -aq)

# Supprimer tous les conteneurs
docker rm $(docker ps -aq)

# Supprimer toutes les images
docker rmi $(docker images -q)
```

Ou en une seule commande:

bash

```
docker stop $(docker ps -aq) && docker rm $(docker ps -aq) && docker rmi $(docker images -q)
```
