

RAPPORT DE PROJET

TOOLBOX AUTOMATISEE



CYBERSECURITY



SUP DE VINCI ISI NANTES
Clément BOUILLAUD – Simon JACQUOT

Table des matières

1.	Introduction et Contexte du Projet	3
1.1.	Présentation de l'entreprise X.....	3
1.2.	Besoins en matière de tests d'intrusion	3
1.3.	Justification du projet.....	3
1.4.	Objectifs du projet	4
1.5.	Portée du projet.....	4
1.6.	Limites du projet.....	4
2.	Schéma d'architecture.....	5
3.	Détails des composants de la toolbox	7
3.1.	Les modules	7
3.1.1.	Script maître : app.py	7
3.1.2.	Module de collecte d'informations	7
3.1.3.	Module d'analyse des vulnérabilités.....	7
3.1.4.	Module d'exploitation des vulnérabilités	8
3.1.5.	Module d'attaque par brute force	8
3.2.	Les Outils	9
3.2.1.	Nmap	9
3.2.2.	WhatWeb	9
3.2.3.	Gobuster	9
3.2.4.	WPScan	10
3.2.5.	Wapiti	10
3.2.6.	Metasploit Framework.....	10
3.2.7.	Pymetasploit3 (Python Metasploit RPC Client)	11
3.2.8.	Flask	11
3.2.9.	Plotly.js	12
4.	Méthodologie de réalisation.....	13
5.	Le planning prévisionnel.....	17
6.	La liste des dépendances	18
6.1.	Dépendances Système	18
6.3.	Contraintes Techniques.....	19
6.5.	Mesures d'atténuation	20
6.6.	La liste, les versions et les licences des composants entrant dans la réalisation du système.....	20

7.	Mode d'emploi	21
7.1.	Durée de fonctionnement de la toolbox	21
7.2.	Prérequis techniques obligatoires.....	21
7.3.	Utilisation de la toolbox.....	22
7.4.	Compréhension du rapport.....	23
8.	Gestion des Risques	27
8.1.	Analyse des risques.....	27
8.1.1.	Risques techniques.....	27
8.1.2.	Risques opérationnels.....	28
8.2.	Stratégie de gestion des incidents et des problèmes	30
8.3.	Maintenance et Évolution : Assurer la pérennité et l'adaptabilité de la toolbox.....	31
8.3.1.	Maintenir la toolbox à jour : un processus continu	32
8.3.2.	Évolution future : anticiper les besoins de demain	33
9.	Conclusion.....	35
9.1.	Récapitulation des points clés.....	35
9.2.	Conformité au cahier des charges	35
9.3.	Perspectives.....	35

1. Introduction et Contexte du Projet

1.1. Présentation de l'entreprise X

L'entreprise X (que l'on va appeler Doli Industry) est une société spécialisée dans le domaine de l'industrie, opérant à l'échelle internationale. Dans un environnement numérique en constante évolution, où les menaces informatiques sont de plus en plus sophistiquées, l'entreprise Doli Industry reconnaît l'importance cruciale de garantir la sécurité de ses systèmes d'information et de ses données sensibles.

1.2. Besoins en matière de tests d'intrusion

Actuellement, l'entreprise Doli Industry réalise des tests d'intrusion de manière manuelle. Cependant, cette approche présente certaines limites :

- Temps de réalisation : Les tests manuels peuvent être longs à réaliser, en particulier pour les systèmes complexes.
- Couverture limitée : Il peut être difficile de couvrir tous les aspects de la sécurité avec des tests manuels.
- Difficulté d'analyse : L'analyse des résultats des tests manuels peut être complexe et chronophage.

1.3. Justification du projet

Pour répondre à ces défis, l'entreprise Doli Industry souhaite se doter d'une toolbox automatisée pour la réalisation de tests d'intrusion. Cette toolbox permettra de :

- Réduire les coûts : En automatisant une partie des tâches, la toolbox réduira le besoin de recourir à des experts en sécurité.
- Accélérer les tests : L'automatisation permettra de réaliser les tests plus rapidement, ce qui permettra de les effectuer plus fréquemment.
- Améliorer la couverture : La toolbox pourra être configurée pour couvrir un large éventail de scénarios de test.
- Faciliter l'analyse : La toolbox générera des rapports détaillés et des visualisations pour faciliter l'interprétation des résultats.

1.4. Objectifs du projet

Les principaux objectifs de ce projet sont les suivants :

- Concevoir et développer une toolbox automatisée pour la réalisation de tests d'intrusion.
- Intégrer une variété d'outils et de techniques de test d'intrusion.
- Fournir une interface utilisateur conviviale pour faciliter l'utilisation de la toolbox.
- Générer des rapports détaillés et des visualisations pour faciliter l'analyse des résultats.
- Améliorer la sécurité globale de l'entreprise Doli Industry en identifiant et en corrigeant les vulnérabilités.

1.5. Portée du projet

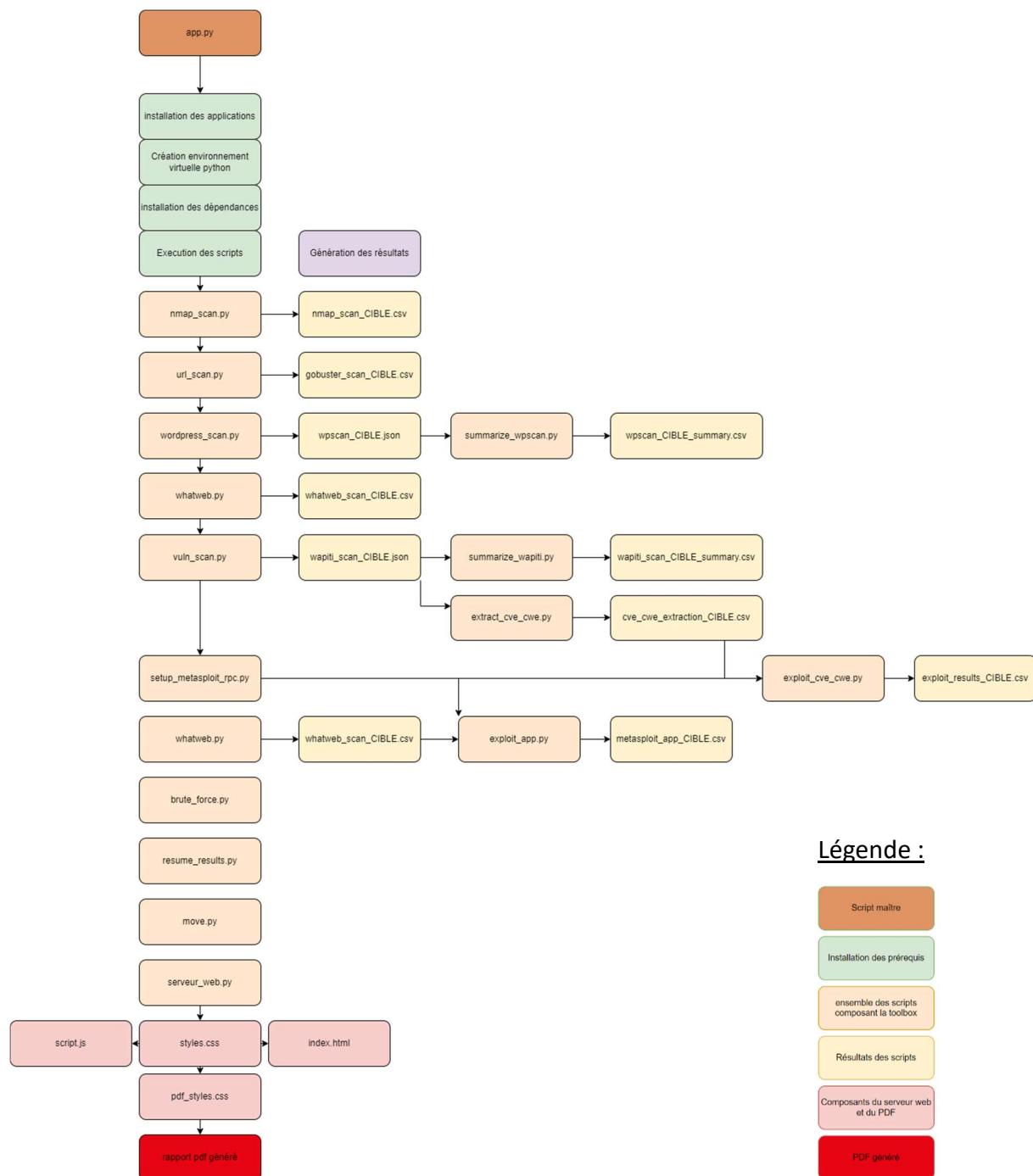
Le projet se concentrera sur le développement de la toolbox elle-même. Il n'inclura pas la mise en œuvre de mesures correctives pour les vulnérabilités identifiées lors des tests. La toolbox sera conçue pour être utilisée par le personnel de sécurité de l'entreprise Doli Industry, qui sera responsable de l'interprétation des résultats et de la prise de mesures correctives appropriées.

1.6. Limites du projet

La toolbox ne pourra pas détecter toutes les vulnérabilités possibles. Elle sera limitée par les outils et les techniques qu'elle intègre. De plus, la toolbox ne pourra pas remplacer complètement les tests d'intrusion manuels, qui restent nécessaires pour certains types de vulnérabilités et pour les systèmes les plus critiques.

2. Schéma d'architecture

L'architecture du projet est organisée en plusieurs modules indépendants, chacun responsable d'une tâche spécifique. Les modules interagissent avec les fichiers de résultats générés lors de l'exécution de chaque étape. Le schéma d'architecture peut être représenté de la manière suivante :



Voici également un aperçu des répertoires de ce projet :

```
projet_sdv_toolbox/      # Répertoire racine du projet
├─ static/              # Dossier pour les fichiers statiques (CSS, JS, images)
│   ├── style.css        # Feuille de style pour la mise en forme de l'interface web
│   ├── pdf_style.css    # Feuille de style pour la mise en forme du pdf
│   └─ script.js         # Fichier JavaScript pour les fonctionnalités dynamiques du tableau de bord
├─ templates/           # Dossier pour les templates HTML du serveur web
│   └─ index.html        # Template principal pour afficher les résultats sur la page web
├─ results/             # Dossier contenant les résultats des scans et analyses
├─ app.py               # Script principal qui exécute tous les autres scripts
├─ bruteforce.py        # Script pour effectuer des attaques de brute force
├─ exploit_appli.py     # Exploite des vulnérabilités liées aux applications
├─ exploit_cve_cwe.py   # Exploite les CVE et CWE trouvées
├─ extract_cve_cwe.py   # Extraire les CVE et CWE des résultats de scan
├─ move.py              # Déplace les fichiers de résultats dans des dossiers spécifiques
├─ nmap_scan.py         # Effectue un scan Nmap
├─ README.md            # Fichier d'explication du projet (à compléter)
├─ resume_results.py    # Crée un résumé des résultats des différents scans
├─ serveur_web.py       # Lancement du serveur web Flask pour afficher les résultats
├─ setup_metasploit_rpc.py # Configure le service RPC de Metasploit
├─ summarize_wapiti.py  # Résume les résultats du scan Wapiti dans un fichier CSV
├─ summarize_wpscan.py  # Résume les résultats du scan WPScan dans un fichier CSV
├─ url_scan.py          # Effectue un scan de répertoires à l'aide de Gobuster
├─ vuln_scan.py         # Lance Wapiti pour analyser les vulnérabilités web
├─ whatweb.py           # Identifie les technologies et applications d'un site web
├─ wordpress_scan.py    # Lance WPScan sur un site WordPress
├─ results/             # Répertoire où sont déplacés tous les fichiers de résultats pour une cible donnée
│   └─ target_date/     # Sous-répertoire pour chaque cible (target) et date du scan
│       ├── nmap_scan_target.csv      # Résultats du scan Nmap
│       ├── gobuster_scan_target_80.csv # Résultats du scan Gobuster pour le port 80
│       ├── gobuster_scan_target_443.csv # Résultats du scan Gobuster pour le port 443
│       ├── wpscan_target.json        # Résultats du scan WPScan
│       ├── wpscan_target_summary.csv  # Résumé des résultats WPScan
│       ├── wapiti_scan_target.json   # Résultats du scan Wapiti
│       ├── wapiti_scan_target_summary.csv # Résumé des résultats Wapiti
│       ├── cve_cwe_extraction_target.csv # Extraction des CVE et CWE détectées
│       ├── exploit_results_target.csv # Résultats des exploits tentés
│       ├── brute_force_results_target.csv # Résultats des tentatives de brute force
│       ├── whatweb_results_target.csv # Résultats du scan WhatWeb
│       ├── metasploit_app_target.csv  # Résultats des exploits Metasploit pour les applications détectées
│       └─ summary_results_target.csv  # Résumé général des résultats
```

3. Détails des composants de la toolbox

La toolbox est conçue pour automatiser une grande partie du processus de test d'intrusion, de la collecte d'informations à l'exploitation des vulnérabilités. Chaque module remplit une fonction spécifique, contribuant à une analyse complète de la cible. L'orchestration de ces modules est assurée par le script maître `app.py`, qui gère la configuration et l'exécution séquentielle des différents scripts.

3.1. Les modules

3.1.1. Script maître : `app.py`

Ce module orchestre l'exécution des différents scripts, gère la configuration de l'environnement et exécute les scans et analyses dans un ordre défini.

3.1.2. Module de collecte d'informations

- `nmap_scan.py` : Scanne les ports ouverts sur la cible.
- `whatweb.py` : Identifie les technologies et applications Web utilisées sur la cible.
- `url_scan.py` : Scanne les répertoires accessibles via HTTP(S) avec Gobuster.
- `wordpress_scan.py` : Effectue un scan WPScan pour les sites WordPress détectés.

3.1.3. Module d'analyse des vulnérabilités

- `vuln_scan.py` : Utilise Wapiti pour détecter les vulnérabilités des applications Web.
- `extract_cve_cwe.py` : Extrait les CVE et CWE des résultats de Wapiti.

`summarize_wapiti.py` et `summarize_wpscan.py` : Résumement les résultats des scans Wapiti et WPScan en fichiers CSV pour une meilleure présentation dans le serveur web.

3.1.4. Module d'exploitation des vulnérabilités

- `exploit_cve_cwe.py` : Tente d'exploiter les vulnérabilités CVE/CWE détectées en utilisant Metasploit.
- `exploit_appli.py` : Exploite les applications identifiées par WhatWeb en utilisant Metasploit.

3.1.5. Module d'attaque par brute force

- `brute_force.py` : Effectue des attaques par brute force sur les ports Web ouverts détectés.

3.1.6. Module de génération des rapports

- `resume_results.py` : Génère un résumé des résultats sous forme de fichier CSV.
- `serveur_web.py` : Démarre un serveur Flask pour afficher les résultats sous forme de tableaux et de graphiques interactifs.
- `move.py` : Organise les fichiers de résultats dans des répertoires par cible et par date.

3.1.7. Python 3.11 et Environnement Virtuel (venv)

- Portabilité : Python est un langage multiplateforme qui peut être exécuté sur différents systèmes d'exploitation (Windows, Linux, macOS), ce qui le rend très portable.
- Bibliothèques riches : Python possède une vaste bibliothèque standard et un écosystème de bibliothèques tierces qui facilitent le développement rapide et l'automatisation des tâches complexes.
- Gestion des dépendances : L'utilisation de `venv` permet d'isoler les dépendances du projet, évitant ainsi les conflits entre différentes versions de bibliothèques installées sur le système.
- Facilité d'apprentissage : Python est réputé pour sa syntaxe simple et lisible, ce qui accélère le développement et la maintenance du code.

3.2. Les Outils

3.2.1. Nmap

- Popularité et fiabilité : Nmap est l'outil standard de l'industrie pour le scan des ports et la détection des services réseau. Il est largement reconnu pour sa fiabilité et ses résultats précis.
- Fonctionnalités avancées : Nmap offre une détection avancée des versions de services, la possibilité d'effectuer des scans SYN et la détection des systèmes d'exploitation, rendant l'outil extrêmement puissant pour l'analyse de la surface d'attaque.
- Extensibilité : Grâce à ses scripts NSE (Nmap Scripting Engine), Nmap peut être étendu pour effectuer des tâches supplémentaires comme l'analyse de vulnérabilités ou la détection de services spécifiques.

Bien que d'autres outils de scan de ports existent (comme Masscan ou Unicornscan), Nmap a été choisi pour sa combinaison de fiabilité, de fonctionnalités avancées et d'extensibilité, qui répondent parfaitement aux exigences de notre projet.

3.2.2. WhatWeb

- Détection de technologies : WhatWeb permet de détecter rapidement les technologies et frameworks web utilisés sur une cible, ce qui est essentiel pour orienter les étapes suivantes de l'audit de sécurité.
- Personnalisable : L'outil permet d'ajouter des plugins ou d'étendre ses capacités pour identifier des technologies spécifiques, ce qui le rend très flexible.
- Rapidité : WhatWeb est optimisé pour scanner rapidement les sites web, ce qui en fait un excellent choix pour une première reconnaissance des technologies utilisées par une cible.

3.2.3. Gobuster

- Brute-force de répertoires et fichiers : Gobuster est un outil de choix pour découvrir des répertoires et fichiers cachés sur un serveur web en utilisant des attaques par dictionnaire. Cela aide à identifier des points d'accès potentiels non documentés ou non sécurisés.
- Vitesse : Gobuster est écrit en Go, ce qui le rend extrêmement rapide par rapport à d'autres outils écrits en Python ou en Bash.

- Fiabilité : L'outil est simple, efficace, et largement utilisé dans les communautés de sécurité pour des tests rapides et fiables.

3.2.4. WPScan

- Spécialisation WordPress : WPScan est spécifiquement conçu pour détecter les vulnérabilités de WordPress, ce qui en fait l'outil idéal pour auditer cette plateforme, qui est l'une des plus répandues sur le web.
- Base de données des vulnérabilités : WPScan utilise une base de données de vulnérabilités de WordPress régulièrement mise à jour, assurant ainsi que l'analyse est basée sur les informations de sécurité les plus récentes.
- Automatisation : WPScan permet d'automatiser la détection des failles courantes dans WordPress, ce qui est crucial pour gagner du temps dans un audit de sécurité.

3.2.5. Wapiti

- Test de vulnérabilités web : Wapiti est utilisé pour effectuer des tests de sécurité web, notamment en cherchant des vulnérabilités telles que les injections SQL, XSS, et les failles de fichiers inclus. C'est un outil puissant pour automatiser l'évaluation de la sécurité des applications web.
- Analyse hors-ligne : Wapiti peut fonctionner sans se connecter au site cible en utilisant un sitemap ou un fichier d'entrée, ce qui est utile pour les tests dans des environnements contrôlés.
- Compatibilité : Wapiti génère des rapports au format JSON qui peuvent facilement être intégrés ou analysés par d'autres outils de la chaîne.

3.2.6. Metasploit Framework

- Cadre d'exploitation complet : Metasploit est l'outil par excellence pour les tests d'exploitation, offrant un large éventail d'exploits, de payloads, et de modules auxiliaires pour compromettre des systèmes après avoir détecté des vulnérabilités.
- Communauté active : Le framework est soutenu par une communauté active et une base de données régulièrement mise à jour, assurant l'accès aux derniers exploits et techniques d'attaque.

- Extensibilité : Metasploit permet de développer et d'intégrer facilement de nouveaux modules, rendant l'outil très adaptable aux besoins spécifiques des tests de sécurité.

Metasploit se distingue par son approche modulaire et sa vaste bibliothèque d'exploits, de payloads et de modules auxiliaires, permettant de simuler une grande variété d'attaques. Cette flexibilité est cruciale pour tester la résistance de la cible face à différents scénarios d'exploitation.

3.2.7. pymetasploit3 (Python Metasploit RPC Client)

- Automatisation des exploits : En utilisant pymetasploit3, il est possible d'automatiser l'interaction avec le framework Metasploit, permettant de lancer des attaques, de gérer les sessions et de récupérer les résultats de manière programmée.
- Intégration avec Python : L'intégration avec Python permet de combiner les capacités de Metasploit avec d'autres outils Python pour des analyses et des attaques plus complexes.
- Efficacité : Ce module facilite le contrôle de Metasploit via des scripts, rendant les tests d'exploitation plus rapides et moins sujets à des erreurs humaines.

3.2.8. Flask

- Framework web léger : Flask est un micro-framework pour Python, idéal pour développer des applications web légères comme une interface pour visualiser les résultats des scans de sécurité.
- Modularité : Flask est extrêmement modulaire, permettant d'ajouter uniquement les extensions nécessaires, ce qui le rend performant pour des applications spécifiques comme celle utilisée dans ce projet.
- Facilité de développement : Flask est simple à utiliser et à configurer, avec une courbe d'apprentissage douce, ce qui accélère le développement de l'interface web.

Nous aurions pu utiliser Django ou FastAPI mais ce sont des outils "lourd" et complexe.

3.2.9. Plotly.js

- Visualisation interactive : Plotly.js est une bibliothèque JavaScript qui permet de créer des graphiques interactifs et dynamiques, essentiels pour visualiser les résultats des analyses de sécurité de manière claire et intuitive.
- Support pour de nombreux types de graphiques : Plotly.js offre un large éventail de types de graphiques, y compris les camemberts, les graphiques en barres, et bien plus, ce qui est utile pour représenter divers types de données.
- Intégration avec Flask : Plotly.js s'intègre bien avec Flask pour générer des visualisations côté client, permettant une expérience utilisateur enrichie sur l'interface web.

Les composants de la toolbox, organisés en modules distincts, permettent une approche structurée et efficace des tests d'intrusion. De la collecte d'informations à l'exploitation des vulnérabilités, chaque module joue un rôle clé dans l'analyse de la sécurité de la cible. La génération de rapports détaillés et la visualisation interactive des résultats facilitent l'interprétation des données et la prise de décision éclairée pour renforcer la sécurité du système.

4. Méthodologie de réalisation

La réalisation de ce projet a été structurée autour d'une approche en phases, chaque phase étant dédiée à un aspect spécifique du processus de pentesting automatisé. Cette approche garantit une progression logique du projet, permettant de valider chaque étape avant de passer à la suivante.

4.1. Phase de Préparation

- Objectif : Configurer l'environnement de travail et préparer les outils nécessaires pour le projet.
- Actions :
 - Création d'un environnement virtuel Python pour isoler les dépendances.
 - Installation des outils nécessaires via apt-get et pip.
 - Vérification de la disponibilité des outils et de leur bon fonctionnement.
- Critères de succès :
 - L'environnement virtuel est créé et activé avec succès
 - Tous les outils requis sont installés et fonctionnent correctement
- Livrable : Environnement de travail fonctionnel avec tous les outils prêts à être utilisés.

4.2. Phase de Scan Initial

- Objectif : Réaliser un scan initial de la cible pour identifier les services et technologies en place.
- Actions :
 - Exécution des scripts `nmap_scan.py`, `whatweb.py`, `url_scan.py`, et `wordpress_scan.py`.
 - Collecte des résultats dans des fichiers CSV.
- Critères de succès :
 - Tous les scans ont été exécutés sans erreur
 - Les résultats ont été enregistrés dans les fichiers CSV correspondants

- Gestion des erreurs :
 - Si un scan échoue, une notification est envoyée à l'utilisateur avec les détails de l'erreur
 - Les résultats partiels sont toujours enregistrés, même en cas d'erreur
- Livrable : Résultats des scans initiaux sous forme de fichiers CSV.

4.3. Phase d'Analyse et Résumé

- Objectif : Analyser les résultats des scans pour extraire les vulnérabilités potentielles et les résumer.
- Actions :
 - Exécution des scripts `summarize_wpscan.py`, `summarize_wapiti.py`, `extract_cve_cwe.py`.
 - Compilation des résultats dans des fichiers CSV.
- Critères de succès :
 - Les scripts d'analyse ont été exécutés sans erreur
 - Les vulnérabilités potentielles ont été extraites et classées par ordre de gravité
 - Les résumés des résultats sont clairs et concis
- Gestion des erreurs :
 - Si un script d'analyse échoue, une notification est envoyée à l'utilisateur avec les détails de l'erreur
 - Les résultats partiels sont toujours enregistrés, même en cas d'erreur
- Livrable :
 - Résumés des résultats de scan et fichiers CSV d'analyse des vulnérabilités.

4.4. Phase d'Exploitation

- Objectif : Exploiter les vulnérabilités identifiées pour confirmer leur présence et leur impact.
- Actions :
 - Utilisation de Metasploit via les scripts `exploit_cve_cwe.py` et `exploit_appli.py` pour exploiter les vulnérabilités.
 - Enregistrement des résultats dans des fichiers CSV.
- Critères de succès :
 - Les scripts d'exploitation ont été exécutés sans erreur
 - Les vulnérabilités ont été exploitées avec succès et les preuves ont été collectées
 - Aucun dommage n'a été causé à la cible lors de l'exploitation
- Gestion des erreurs :
 - Si un script d'exploitation échoue, une notification est envoyée à l'utilisateur avec les détails de l'erreur
 - Les tentatives d'exploitation infructueuses sont enregistrées pour analyse ultérieure
- Livrable : Rapports d'exploitation documentant les vulnérabilités exploitées avec succès.

4.5. Phase de Consolidation

- Objectif : Consolider tous les résultats et les déplacer dans une structure de répertoires appropriée.
- Actions :
 - Exécution du script `move.py` pour organiser les fichiers de résultats.
 - Génération d'un résumé global des résultats via `resume_results.py`.
- Critères de succès :
 - Tous les fichiers de résultats ont été déplacés dans la structure de répertoires correcte
 - Le résumé global a été généré avec succès.

- Gestion des erreurs :
 - Si une erreur se produit lors du déplacement ou de la génération du résumé, une notification est envoyée à l'utilisateur.
- Livrable : Dossier structuré contenant tous les résultats et résumés.

4.6. Phase de Présentation

- Objectif : Présenter les résultats de manière claire et exploitable via une interface web.
- Actions :
 - Déploiement de l'interface web en utilisant Flask via le script `serveur_web.py`.
- Critères de succès :
 - L'interface web est accessible et fonctionnelle
 - Les résultats sont affichés de manière claire et interactive
- Gestion des erreurs :
 - Si l'interface web ne peut pas être démarrée, une notification est envoyée à l'utilisateur avec les détails de l'erreur
- Livrable : Interface web fonctionnelle pour consulter les résultats du pentest
- Contrôle de version
 - Le projet est géré avec GitHub pour suivre les modifications du code et des données.
 - Les commits réguliers permettent de revenir à des versions antérieures si nécessaire.

4.7. Amélioration continue

- La méthodologie est itérative et permet des ajustements en fonction des résultats de chaque phase.
- De nouvelles fonctionnalités ou améliorations peuvent être ajoutées dans les versions futures de la toolbox en fonction des besoins et des retours d'expérience.

5. Le planning prévisionnel

Phase	Date de Début	Date de Fin	Activité	Livrable	Responsable
Préparation	20/06/2024	26/06/2024	Installation des outils nécessaires, configuration de l'environnement virtuel.	Environnement virtuel configuré, outils installés	Simon Jacquot
Scan Initial	27/06/2024	05/07/2024	Exécution des scripts de scan (nmap_scan.py, whatweb.py, url_scan.py, wordpress_scan.py).	Fichiers CSV des résultats de scan	Clément Bouillaud
Analyse des Résultats	08/07/2024	15/07/2024	Analyse des résultats des scans (summarize_wpscan.py, summarize_wapiti.py).	Résumés des résultats dans des fichiers CSV	Simon Jacquot
Extraction des Vulnérabilités	16/07/2024	22/07/2024	Extraction des vulnérabilités à partir des résultats (extract_cve_cwe.py).	Fichiers CSV des vulnérabilités (CVE/CWE)	Clément Bouillaud
Exploitation des Vulnérabilités	23/07/2024	02/08/2024	Exploitation des vulnérabilités via Metasploit (exploit_cve_cwe.py, exploit_appli.py).	Rapports d'exploitation	Simon Jacquot
Consolidation des Résultats	05/08/2024	09/08/2024	Consolidation des résultats dans une structure organisée (move.py).	Dossier structuré avec tous les résultats	Clément Bouillaud
Génération du Résumé	12/08/2024	16/08/2024	Génération d'un résumé global des résultats (resume_results.py).	Résumé global des résultats	Simon Jacquot
Déploiement de l'Interface Web	19/08/2024	23/08/2024	Déploiement de l'interface web pour la présentation des résultats (serveur_web.py).	Interface web opérationnelle	Simon Jacquot
Revue et Validation	26/08/2024	30/08/2024	Validation des résultats, revue des livrables.	Validation des résultats et des livrables	Simon Jacquot, Clément Bouillaud
Rédaction de la Documentation	02/09/2024	03/09/2024	Rédaction de la documentation du projet et des résultats.	Documentation complète du projet	Simon Jacquot, Clément Bouillaud
Présentation Finale	03/09/2024	03/09/2024	Préparation et présentation des résultats finaux aux parties prenantes.	Présentation finale	Simon Jacquot, Clément Bouillaud

6. La liste des dépendances

6.1. Dépendances Système

- Python 3.11
- pip (Gestionnaire de paquets Python)
- Virtualenv (venv) (pour l'isolation des environnements Python)
- Gobuster (outil de brute-force pour les répertoires)
- WPScan (analyseur de vulnérabilités WordPress)
- Nmap (outil de scan de ports)
- Metasploit Framework (outil d'exploitation des vulnérabilités)

6.2. Dépendances Python du Fichier 'requirements.txt'

```
requirements.txt
1  # Pour les requêtes HTTP
2  requests
3
4  # Pour exécuter des commandes shell externes
5  subprocess-run
6
7  # Pour l'analyse des fichiers CSV
8  pandas
9
10 # Pour le scan Wapiti
11 wapiti3
12
13 # Pour gérer les chemins de fichiers et autres utilitaires
14 pathlib
15
16 # Pour l'exécution des scans Nmap en Python
17 python-nmap
18
19 # Pour la gestion des mots de passe (utilisé pour corriger l'erreur bcrypt)
20 bcrypt==3.2.0
21 passlib
22
23 # Pour la mise à jour de setuptools et pip (si nécessaire)
24 setuptools
25
26 # Pour le serveur web
27 flask
28 pandas
29 matplotlib
30 plotly
31 WeasyPrint
32 pdfkit
33
34 # Pour metasploit
35 pymetasploit3
```

6.3. Contraintes Techniques

6.3.1. Configuration des environnements

- Description : La configuration de l'environnement Python, des outils, et des bibliothèques peut varier selon le système d'exploitation.
- Impact : Un échec dans la configuration pourrait retarder le projet ou entraîner des dysfonctionnements dans l'exécution des scripts.

6.3.2. Dépendances externes

- Description : Le projet dépend de plusieurs outils tiers (WPScan, Nmap, Metasploit) dont la disponibilité et la compatibilité peuvent varier.
- Impact : Des mises à jour ou des incompatibilités peuvent survenir, rendant certains outils inutilisables sans modifications ou ajustements.

6.4. Risques Opérationnels

6.4.1. Fausses vulnérabilités (faux positifs)

- Description : Certains outils peuvent signaler des vulnérabilités qui n'existent pas réellement.
- Impact : Cela peut conduire à des pertes de temps lors de l'analyse ou à des rapports erronés.

6.4.2. Exploitation non fiable :

- Description : L'automatisation des exploits via Metasploit peut ne pas fonctionner dans certains environnements ou configurations cibles spécifiques.
- Impact : Certaines vulnérabilités pourraient ne pas être exploitées, réduisant la portée du test.

6.5. Mesures d'atténuation

- Documentation : Maintenir une documentation précise des versions et configurations utilisées pour faciliter la reproduction des environnements.
- Validation Manuelle : Effectuer une validation manuelle des résultats critiques pour réduire les faux positifs.
- Tests dans un Environnement Contrôlé : Tester les outils et scripts dans un environnement contrôlé avant de les appliquer à des systèmes de production.

6.6. La liste, les versions et les licences des composants entrant dans la réalisation du système

Composant	Version	Licence	Preuve des Droits Acquis
Python	3.11	PSF License (Python Software Foundation License)	Open-source, installation via apt-get
Flask	2.0.1	BSD-3-Clause	Installé via pip, open-source
pandas	1.5.3	BSD-3-Clause	Installé via pip, open-source
Plotly	5.14.1	MIT License	Installé via pip, open-source
Nmap	7.92	GNU General Public License (GPL)	Installé via apt-get, open-source
Gobuster	3.3.0	Apache License 2.0	Installé via apt-get, open-source
WPScan	3.8.14	GPLv3	Installé via apt-get, open-source
Metasploit Framework	6.1.4	BSD-3-Clause	Installé via apt-get, open-source
Wapiti	3.0.4	GPLv2	Installé via apt-get, open-source
WhatWeb	0.5.0	GPLv2	Installé via apt-get, open-source
SQLMap	1.5.12	GPLv2	Installé via apt-get, open-source
pymetasploit3	1.0.0	MIT License	Installé via pip, open-source
requests	2.31.0	Apache License 2.0	Installé via pip, open-source
Virtualenv (venv)	20.23.0	MIT License	Installé via apt-get, open-source
apt	-	GNU General Public License (GPL)	Utilisé pour installer des packages système, open-source
pip	23.1.2	MIT License	Gestionnaire de paquets Python, installé via apt-get, open-source

Chaque outil et composant utilisé dans ce projet est sous licence open-source, permettant une utilisation gratuite et modifiable dans le cadre du projet. Toutes les licences respectent les conditions de redistribution et d'utilisation conformément aux termes de ces licences.

7. Mode d'emploi

7.1. Durée de fonctionnement de la toolbox

La toolbox a été testée de nombreuses fois et sur différentes cibles. La durée de fonctionnement est donc très variable et dépendra de chaque cible. Plus la cible aura de vulnérabilité, plus la toolbox mettra de temps à analyser et exploiter la cible. Voici une liste de différents tests :

Serveur WEB	OS	Durée
GLPI	Debian 12	13 min
DVWA	KALI	17 min
AGORA	Ubuntu	15 min
MediaWiki	Debian 11	16 min

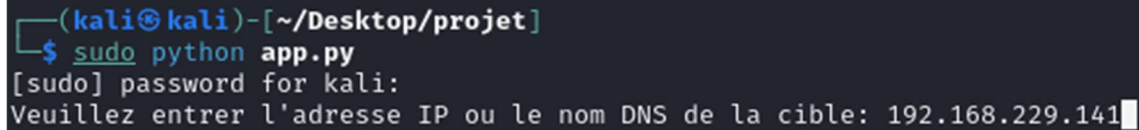
7.2. Prérequis techniques obligatoires

- Utilisation d'un environnement Kali Linux avec interface graphique disponible à cet endroit : <https://www.kali.org/get-kali/#kali-virtual-machines>
- L'environnement Kali Linux utilisé doit être à jour sinon certain paquet risque de ne pas fonctionner
- Réaliser l'ensemble des actions nécessaires de la toolbox dans un emplacement où l'utilisateur aura les droits de lecture et d'écriture et utiliser la commande « sudo »
- Télécharger la toolbox à cet emplacement :
https://github.com/simonjacquot/projet_sdv_toolbox.git
Exemple de commande :
git clone https://github.com/simonjacquot/projet_sdv_toolbox.git
- Se placer dans le répertoire de la toolbox
 - o cd projet_sdv_toolbox

7.3. Utilisation de la toolbox

- Lancement de l'outil :

- Commande : **sudo python app.py**
- Saisi de l'adresse IP ou du nom DNS de la cible, tel que :



```
(kali㉿kali)-[~/Desktop/projet]
$ sudo python app.py
[sudo] password for kali:
Veuillez entrer l'adresse IP ou le nom DNS de la cible: 192.168.229.141
```

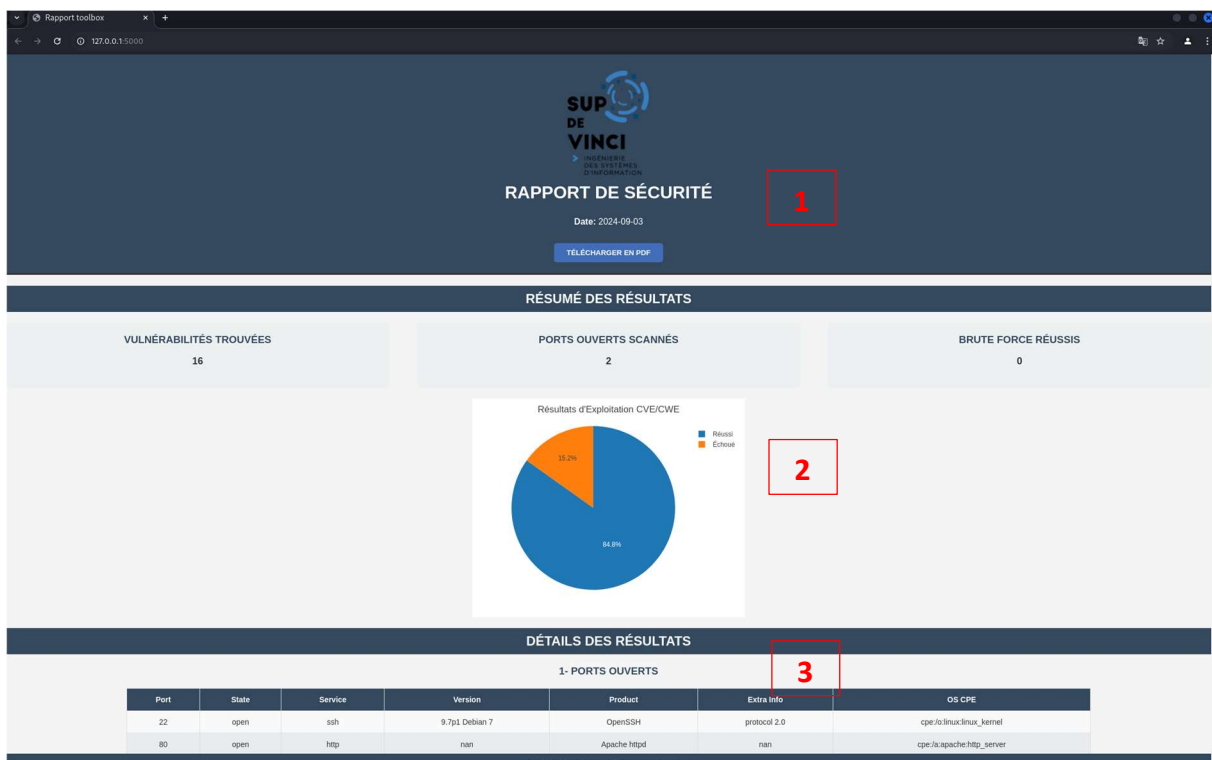
- Faire « Entrée » et patienter

Information importante : lors de la première exécution l'ensemble des outils et dépendances nécessaires seront installés. La durée d'exécution est donc plus longue.

- Lorsque les différentes analyses et attaques seront terminées, un serveur web sera automatiquement ouvert par le biais du navigateur par défaut

7.4. Compréhension du rapport

- Le rapport généré est divisé en trois parties :
 - 1 : L'en tête du rapport composé notamment d'un bouton permettant de générer et télécharger le rapport au format PDF ;
 - 2 : Le résumé des résultats composé de deux tableaux de bord importants, soit le nombre de vulnérabilités trouvées, de ports ouverts et brute force réussis, ainsi que le pourcentage de réussite d'exploitation des vulnérabilités trouvées ;
 - 3 : Le détails des résultats, soit l'ensemble des fichiers csv générés par tous les outils utilisés dans cette toolbox sous forme de tableau.



- La partie « DETAILS DES RESULTATS » est composée de :
 - La liste des ports ouverts et détectés par l'outil NMAP ;

1- PORTS OUVERTS						
Port	State	Service	Version	Product	Extra Info	OS CPE
22	open	ssh	9.7p1 Debian 7	OpenSSH	protocol 2.0	cpe:/o:linux:linux_kernel
80	open	http	nan	Apache httpd	nan	cpe:/a:apache:http_server

- Les informations concernant l'application utilisée par l'outil WhatWeb ;
(Ces données permettront de mieux cibler les attaques afin de récolter un maximum de données et d'évaluer avec précision la sécurité mise en place)

2- RÉSULTATS WHATWEB	
URL	Application
http://192.168.229.141:80	GLPI

- L'ensemble des url trouvée par l'outil Gobuster ;

3- SCAN GOBUSTER		
URL	Status Code	Size
/docs	301	236
/hosts	301	237
/database	301	240
/javascript	301	242
/external	301	240
/config	301	238
/vulnerabilities	301	247
/http://www	404	196
/http://youtube	404	196
/http://blogs	404	196
/http://blog	404	196
/http://www	404	196
/server-status	403	199
/http://community	404	196
/http://radar	404	196
/http://jeremiahgrossman	404	196
/http://weblog	404	196
/http://swik	404	196

- Le résultat du Brute Force réalisé par l'outil HYDRA ;
(Les dictionnaires utilisés sont orientés « identifiant et mot de passe par défaut des applications web »)
- Un extrait du scan des vulnérabilités par l'outil Wapiti ;

5- SCAN WAPITI						
Category	Module	Path	Parameter	Info	WSTG Reference	HTTP Request
Content Security Policy Configuration	nan	/	nan	CSP is not set	nan	nan
Content Security Policy Configuration	nan	/	nan	CSP is not set	nan	nan
Content Security Policy Configuration	nan	/	nan	CSP is not set	nan	nan
Content Security Policy Configuration	nan	/	nan	CSP is not set	nan	nan
Content Security Policy Configuration	nan	/	nan	CSP is not set	nan	nan
Content Security Policy Configuration	nan	/	nan	CSP is not set	nan	nan
Content Security Policy Configuration	nan	/	nan	CSP is not set	nan	nan
Content Security Policy Configuration	nan	/	nan	CSP is not set	nan	nan

- L'ensemble des CVE et CWE détecté par l'outil Wapiti ;

6- VULNÉRABILITÉS (CVE/CWE)	
Type	ID
CWE	530
CWE	89
CWE	798
CWE	521
CWE	93
CWE	352
CWE	78
CWE	22
CWE	538
CWE	601
CWE	89
CWE	918
CWE	79
CWE	611
CWE	405
CWE	400

- Le résultat de l'exploitation des modules de l'outil Metasploit grâce aux éléments de l'application trouvés par l'outil WhatWeb ;

7- RÉSULTATS METASPLOIT					
Application	Module	Type	Résultat	Données sensibles	Mesures de sécurité
Apache	auxiliary/admin/appletv/appletv_display_video	auxiliary	réussi	Aucune	Active
Apache	auxiliary/admin/http/tomcat_administration	auxiliary	réussi	Aucune	Active
Apache	auxiliary/admin/http/tomcat_ghostcat	auxiliary	réussi	Aucune	Active
Apache	auxiliary/admin/http/tomcat_utf8_traversal	auxiliary	réussi	Aucune	Active
Apache	auxiliary/admin/http/trendmicro_dlp_traversal	auxiliary	réussi	Aucune	Active
Apache	auxiliary/analyze/crack_webapps	auxiliary	réussi	Aucune	Active
Apache	auxiliary/dos/http/apache_commons_fileupload_dos	auxiliary	réussi	Aucune	Active
Apache	auxiliary/dos/http/apache_mod_isapi	auxiliary	réussi	Aucune	Active
Apache	auxiliary/dos/http/apache_range_dos	auxiliary	réussi	Aucune	Active
Apache	auxiliary/dos/http/apache_tomcat_transfer_encoding	auxiliary	réussi	Aucune	Active
Apache	auxiliary/fileformat/odt_badodt	auxiliary	réussi	Aucune	Active
Apache	auxiliary/gather/apache_rave_creds	auxiliary	réussi	Aucune	Active
Apache	auxiliary/gather/apache_superse_cookie_sig_priv_esc	auxiliary	réussi	Aucune	Active
Apache	auxiliary/gather/cve_2021_27850_apache_tapestry_hmac_key	auxiliary	réussi	Aucune	Active
Apache	auxiliary/gather/impersonate_ssl	auxiliary	réussi	Aucune	Active
Apache	auxiliary/gather/zookeeper_info_disclosure	auxiliary	réussi	Aucune	Active
Apache	auxiliary/scanner/couchdb/couchdb_enum	auxiliary	réussi	Aucune	Active

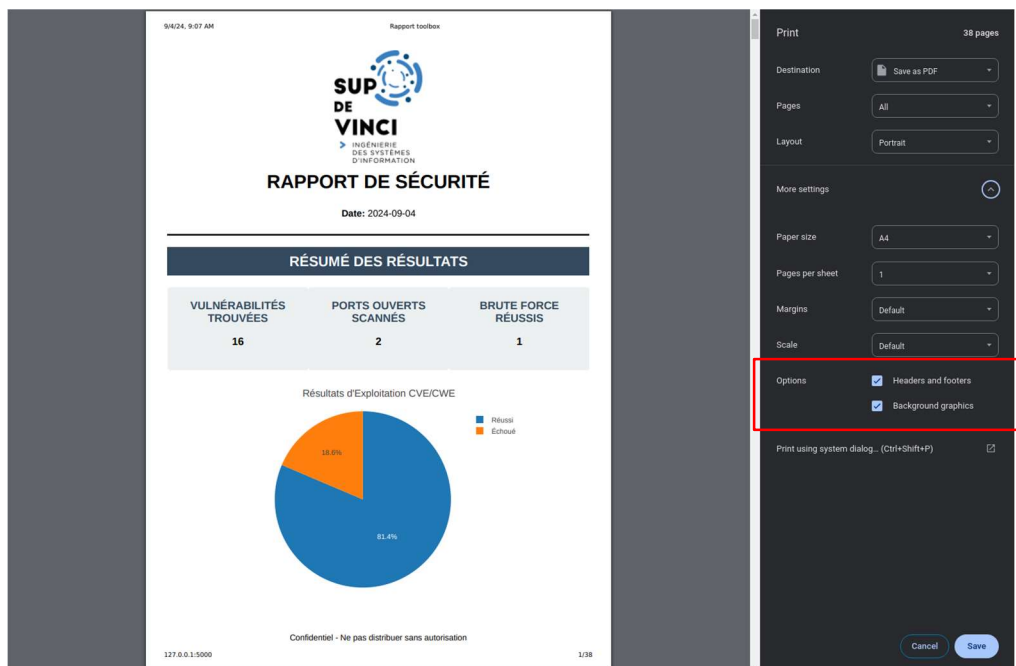
- Extrait du résultat de l'exploitation de chaque vulnérabilités CVE/CWE trouvées précédemment par les modules de l'outil Metasploit ;

8- EXPLOITATION CVE / CWE					
CVE/CWE	Module	Type	Résultat	Données sensibles	Mesures de sécurité
CWE 530	Aucun	nan	échoué	Aucune donnée récupérée	nan
CWE 89	auxiliary/admin/http/ctme_manageiq_evm_pass_reset	auxiliary	réussi	Aucune donnée récupérée	Active
CWE 89	auxiliary/scanner/http/totaljs_traversal	auxiliary	réussi	Aucune donnée récupérée	Active
CWE 89	auxiliary/scanner/ssl/ssl_version	auxiliary	réussi	Aucune donnée récupérée	Active
CWE 89	auxiliary/server/openssl_alchaintsforgery_mitm_proxy	auxiliary	réussi	Aucune donnée récupérée	Active
CWE 798	exploit/linux/http/klog_server_authenticate_user_unauth_command_injection	exploit	réussi	Aucune donnée récupérée	Active
CWE 521	Aucun	nan	échoué	Aucune donnée récupérée	nan
CWE 93	auxiliary/scanner/http/apache_flink_jobmanager_traversal	auxiliary	réussi	Aucune donnée récupérée	Active
CWE 93	auxiliary/server/jsse_skriptis_mitm_proxy	auxiliary	réussi	Aucune donnée récupérée	Active
CWE 93	auxiliary/server/openssl_alchaintsforgery_mitm_proxy	auxiliary	réussi	Aucune donnée récupérée	Active
CWE 93	exploit/linux/http/klog_server_authenticate_user_unauth_command_injection	exploit	réussi	Aucune donnée récupérée	Active

- La partie de l'en-tête du rapport est notamment composé d'un bouton permettant de générer un rapport au format PDF :



- Une fois ce bouton sélectionné, une fenêtre pour imprimer le rapport s'ouvrira
 - **Important : il faut vérifier que le format soit en A4 et que les deux éléments encadrés en rouge ci-dessous soient bel et bien coché**



8. Gestion des Risques

La réalisation de tests d'intrusion, même automatisés, comporte des risques inhérents. Cette section présente une analyse approfondie de ces risques, ainsi que les mesures d'atténuation mises en place pour les minimiser.

8.1. Analyse des risques

8.1.1. Risques techniques

- **Faux positifs/négatifs** : Les outils automatisés peuvent générer des faux positifs (alertes sur des vulnérabilités inexistantes) ou des faux négatifs (non-détection de vulnérabilités réelles).
 - **Atténuation** :
 - **Utilisation de plusieurs outils complémentaires** : Combiner les résultats de différents outils de scan et d'analyse pour réduire le risque de faux positifs et de faux négatifs.
 - **Validation manuelle des résultats critiques** : Les vulnérabilités identifiées comme critiques doivent être vérifiées manuellement par un expert en sécurité pour confirmer leur existence et leur impact.
 - **Mise à jour régulière de la base de données de vulnérabilités** : S'assurer que les outils utilisent les dernières informations sur les vulnérabilités connues pour minimiser les faux négatifs.
 - **Configuration fine des outils** : Ajuster les paramètres des outils pour réduire les faux positifs, en tenant compte du contexte spécifique de l'entreprise Doli Industry.
- **Instabilité des outils** : Les outils open-source peuvent présenter des bugs ou des incompatibilités, ce qui peut entraîner des erreurs lors de l'exécution des tests.
 - **Atténuation** :
 - **Tests approfondis avant déploiement** : Effectuer des tests rigoureux de la toolbox dans un environnement de test isolé avant de la déployer en production.

- **Surveillance continue** : Mettre en place des mécanismes de surveillance pour détecter les erreurs et les problèmes lors de l'exécution des tests.
- **Utilisation de versions stables** : Privilégier les versions stables des outils open-source plutôt que les versions de développement, qui peuvent être moins fiables
- **Contribution à la communauté open-source** : Signaler les bugs et les problèmes rencontrés aux développeurs des outils open-source et, si possible, contribuer à leur correction
- **Dépendances externes** : La toolbox peut dépendre de services externes (API, bases de données) qui peuvent être indisponibles ou instables, ce qui peut perturber le bon fonctionnement de la toolbox
 - **Atténuation** :
 - **Identification des dépendances critiques** : Dresser une liste des services externes dont dépend la toolbox et évaluer leur criticité
 - **Mise en place de mécanismes de tolérance aux pannes** : Implémenter des mécanismes tels que le "fallback" (utilisation d'une alternative en cas d'indisponibilité) ou le "retry" (tentative de reconnexion automatique) pour gérer les interruptions de service
 - **Surveillance de la disponibilité des services externes** : Mettre en place des outils de surveillance pour détecter rapidement les problèmes de disponibilité des services externes et alerter les équipes concernées

8.1.2. Risques opérationnels

- **Impact sur la cible** : Les tests d'intrusion, même en mode "safe", peuvent perturber le fonctionnement normal de la cible ou causer des dommages involontaires, tels que des dénis de service, des pertes de données ou des interruptions de service
 - **Atténuation** :
 - **Obtenir une autorisation écrite avant de lancer les tests** : S'assurer que les tests sont autorisés par les responsables de la cible et qu'ils comprennent les risques

- **Définir clairement la portée et les limites des tests** : Spécifier les systèmes et les types de tests autorisés, ainsi que les actions à éviter
- **Utiliser des environnements de test isolés si possible** : Effectuer les tests dans un environnement de test séparé de la production pour minimiser l'impact sur les opérations normales
- **Surveiller l'impact sur la cible pendant les tests** : Mettre en place des outils de surveillance pour détecter tout impact négatif sur la cible et interrompre les tests si nécessaire
- **Fuite de données** : Les informations collectées lors des tests peuvent être sensibles (identifiants, mots de passe, données personnelles) et doivent être protégées contre tout accès non autorisé
 - **Atténuation** :
 - **Chiffrement des données sensibles** : Utiliser des algorithmes de chiffrement forts pour protéger les données sensibles stockées ou transmises par la toolbox
 - **Stockage sécurisé des résultats** : Stocker les résultats des tests dans un emplacement sécurisé avec un accès restreint
 - **Restriction d'accès aux données** : Limiter l'accès aux données aux personnes autorisées uniquement, en utilisant des mécanismes d'authentification et d'autorisation
 - **Mise en place de procédures de destruction sécurisée des données** : Définir des procédures claires pour supprimer les données sensibles de manière sécurisée après leur utilisation
- **Mauvaise utilisation de la toolbox** : La toolbox pourrait être utilisée à des fins malveillantes par des personnes non autorisées, internes ou externes à l'entreprise
 - **Atténuation** :
 - **Contrôle d'accès strict à la toolbox** : Limiter l'accès à la toolbox aux personnes autorisées uniquement, en utilisant des mots de passe forts et des mécanismes d'authentification à deux facteurs
 - **Journalisation des activités** : Enregistrer toutes les actions effectuées avec la toolbox, y compris les utilisateurs, les cibles et les résultats des tests, pour faciliter la traçabilité et l'audit
 - **Sensibilisation des utilisateurs** : Former les utilisateurs autorisés à l'utilisation responsable de la toolbox et aux bonnes pratiques de sécurité

En mettant en œuvre ces mesures d'atténuation, l'entreprise Doli Industry peut réduire considérablement les risques associés à l'utilisation de la toolbox de pentesting automatisé et garantir que les tests sont effectués de manière sécurisée et efficace.

8.2. Stratégie de gestion des incidents et des problèmes

La gestion efficace des incidents et des problèmes est essentielle pour assurer la continuité des opérations et minimiser l'impact des perturbations. La stratégie suivante sera mise en place :

- **Identification :**

- **Surveillance proactive :** Utilisation d'outils de surveillance pour détecter les anomalies dans le fonctionnement de la toolbox, telles que des erreurs d'exécution, des délais de réponse inhabituels ou des échecs de connexion aux services externes.
- **Rapports d'utilisateurs :** Encourager les utilisateurs à signaler tout problème rencontré lors de l'utilisation de la toolbox, qu'il s'agisse de dysfonctionnements, de résultats inattendus ou de suspicions de sécurité.

- **Analyse :**

- **Collecte d'informations :** Recueillir toutes les informations pertinentes sur l'incident, notamment les journaux d'événements, les captures d'écran, les témoignages d'utilisateurs et les données de surveillance.
- **Investigation approfondie :** Analyser les informations collectées pour identifier la cause racine de l'incident et évaluer son impact sur la sécurité et les opérations.
- **Classification :** Classer l'incident en fonction de sa gravité et de son impact potentiel, en utilisant une échelle prédéfinie (par exemple, critique, majeur, mineur).

- **Résolution :**

- **Plan d'action :** Élaborer un plan d'action pour résoudre l'incident, en tenant compte de sa cause racine et de son impact.
- **Mise en œuvre :** Mettre en œuvre les mesures correctives définies dans le plan d'action, en veillant à minimiser les perturbations pour les utilisateurs.

- **Validation** : Vérifier que l'incident est résolu et que les mesures correctives ont été efficaces.
- **Communication** :
 - **Notification** : Informer les parties prenantes concernées de l'incident et de son état d'avancement, en utilisant des canaux de communication appropriés (e-mail, téléphone, système de tickets).
 - **Transparence** : Fournir des informations claires et précises sur l'incident, en évitant « le jargon technique » et en expliquant les mesures prises pour le résoudre.
 - **Suivi** : Maintenir les parties prenantes informées de l'évolution de la résolution de l'incident jusqu'à sa clôture.
- **Amélioration continue** :
 - **Analyse post-incident** : Mener une analyse approfondie de l'incident pour identifier les points faibles dans la toolbox, les procédures ou la formation des utilisateurs.
 - **Mise en œuvre de mesures préventives** : Mettre en place des mesures pour éviter que des incidents similaires ne se reproduisent, en améliorant la toolbox, les procédures ou la formation.
 - **Documentation** : Documenter les incidents et les leçons apprises pour capitaliser sur l'expérience et améliorer la gestion des incidents futurs.

Cette stratégie de gestion des incidents et des problèmes, combinée à une analyse approfondie des risques, contribuera à garantir que la toolbox de pentesting automatisé est utilisée de manière sûre, efficace et responsable au sein de l'entreprise Doli Industry.

8.3. Maintenance et Évolution : Assurer la pérennité et l'adaptabilité de la toolbox

La toolbox de pentesting automatisé n'est pas un outil statique, mais un système dynamique qui doit évoluer en permanence pour rester efficace face aux menaces en constante mutation et aux avancées technologiques. Cette section détaille les procédures de maintenance et de mise à jour, ainsi que la vision stratégique pour l'évolution future de la toolbox, afin de garantir qu'elle réponde toujours aux besoins de sécurité de l'entreprise Doli Industry.

8.3.1. Maintenir la toolbox à jour : un processus continu

- **Mises à jour régulières des outils** : Les outils open-source qui constituent la toolbox sont constamment améliorés par leurs communautés respectives. Il est crucial de suivre ces mises à jour pour bénéficier des nouvelles fonctionnalités, des corrections de bugs et des renforcements de sécurité.
 - **Processus de mise à jour** :
 1. **Veille active** : Suivre les canaux de communication des projets open-source pour être informé des nouvelles versions et des correctifs de sécurité.
 2. **Tests en environnement isolé** : Tester rigoureusement les nouvelles versions dans un environnement de test dédié avant de les déployer en production, afin d'éviter toute perturbation.
 3. **Mise à jour de la documentation** : Maintenir la documentation de la toolbox à jour pour refléter les changements apportés par les mises à jour.
- **Adaptation aux nouvelles vulnérabilités** : Le paysage des menaces cybernétiques est en constante évolution, avec de nouvelles vulnérabilités découvertes régulièrement. La toolbox doit être capable de détecter et d'exploiter ces nouvelles failles.
 - **Veille technologique** : Suivre de près les publications de sécurité, les alertes de vulnérabilités et les bases de données de CVE pour rester à la pointe des dernières menaces.
 - **Développement de nouvelles capacités** : Ajouter de nouveaux modules ou scripts à la toolbox pour détecter et exploiter les nouvelles vulnérabilités.
 - **Adaptation des modules existants** : Mettre à jour les modules existants pour prendre en compte les nouvelles techniques d'attaque et de défense.

- **Correction des bugs et des problèmes** : L'utilisation de la toolbox peut révéler des bugs ou des comportements inattendus. Il est impératif de les corriger rapidement pour assurer la fiabilité et l'efficacité de l'outil.
 - **Suivi des incidents** : Mettre en place un système de suivi des incidents pour enregistrer et gérer les bugs et les problèmes signalés par les utilisateurs.
 - **Analyse et résolution** : Analyser les incidents pour en identifier la cause racine et mettre en œuvre les corrections nécessaires.
 - **Tests de non-régression** : Effectuer des tests de régression après chaque correction pour garantir que les modifications n'ont pas introduit de nouveaux problèmes.

8.3.2. Évolution future : anticiper les besoins de demain

La toolbox doit évoluer pour répondre aux besoins changeants de l'entreprise X en matière de sécurité. Voici quelques pistes d'évolution envisageables :

- **Nouveaux outils et fonctionnalités** :
 - Intégrer de nouveaux outils open-source ou commerciaux pour étendre les capacités de détection et d'exploitation des vulnérabilités.
 - Développer de nouvelles fonctionnalités pour répondre à des besoins spécifiques, comme l'analyse de la sécurité des configurations, la détection d'attaques en cours ou la génération de rapports personnalisés.
- **Interface utilisateur améliorée** :
 - Rendre l'interface web encore plus conviviale et intuitive.
 - Ajouter de nouvelles visualisations, des fonctionnalités de recherche et de filtrage, et des options de personnalisation pour faciliter l'analyse des résultats.
- **Intégration avec l'écosystème de sécurité** :
 - Connecter la toolbox à d'autres outils de sécurité utilisés par l'entreprise Doli Industry (SIEM, outils de gestion des vulnérabilités) pour une meilleure corrélation des données et une prise de décision plus éclairée.

La planification de ces évolutions futures sera guidée par :

- **Les retours d'expérience des utilisateurs** : Recueillir les commentaires des utilisateurs pour identifier les points forts, les points faibles et les besoins en nouvelles fonctionnalités.
- **L'évolution des menaces** : Suivre les tendances en matière de cybersécurité pour anticiper les nouvelles menaces et adapter la toolbox en conséquence.
- **Les nouvelles technologies** : Évaluer les nouvelles technologies et les nouveaux outils qui pourraient renforcer les capacités de la toolbox.

En adoptant une approche proactive de maintenance et d'évolution, l'entreprise Doli Industry s'assure que sa toolbox de pentesting automatisé reste un atout majeur pour sa stratégie de sécurité, lui permettant de protéger efficacement ses actifs numériques face aux menaces toujours plus sophistiquées.

9. Conclusion

9.1. Récapitulation des points clés

Ce rapport a présenté en détail le projet de développement d'une toolbox automatisée pour la réalisation de tests d'intrusion, conçue pour répondre aux besoins spécifiques de l'entreprise X en matière de sécurité. Nous avons exploré les choix technologiques qui sous-tendent la toolbox, la méthodologie rigoureuse adoptée pour sa réalisation, ainsi que les mesures mises en place pour gérer les risques et assurer la maintenance et l'évolution future de l'outil.

9.2. Conformité au cahier des charges

La toolbox, telle que conçue et développée, répond aux exigences du cahier des charges. Elle intègre une variété d'outils et de techniques de test d'intrusion, offre une interface utilisateur conviviale, génère des rapports détaillés et des visualisations, et contribue à améliorer la sécurité globale de l'entreprise X en identifiant et en facilitant la correction des vulnérabilités.

9.3. Perspectives

La toolbox de pentesting automatisé constitue un atout précieux pour l'entreprise Doli Industry, lui permettant de renforcer sa posture de sécurité et de protéger ses actifs numériques de manière proactive. Les perspectives pour ce projet sont prometteuses, avec des possibilités d'évolution pour répondre aux besoins changeants de l'entreprise et aux nouvelles menaces qui pourraient émerger.

- **Améliorations continues** : La toolbox sera régulièrement mise à jour et améliorée pour intégrer de nouveaux outils, de nouvelles fonctionnalités et pour s'adapter aux évolutions du paysage des menaces.
- **Intégration plus poussée** : La toolbox pourrait être intégrée à d'autres outils de sécurité de l'entreprise Doli Industry pour une meilleure corrélation des données et une gestion plus efficace des vulnérabilités.

- **Intelligence artificielle** : L'intégration de techniques d'intelligence artificielle pourrait permettre d'automatiser davantage certaines tâches, d'améliorer la détection des vulnérabilités et de prédire les attaques potentielles.

En somme, ce projet a permis de créer un outil puissant et flexible pour aider l'entreprise Doli Industry à renforcer sa sécurité. Grâce à une maintenance continue et à une évolution stratégique, cette toolbox continuera à jouer un rôle clé dans la protection des actifs numériques de l'entreprise face aux défis de la cybersécurité.