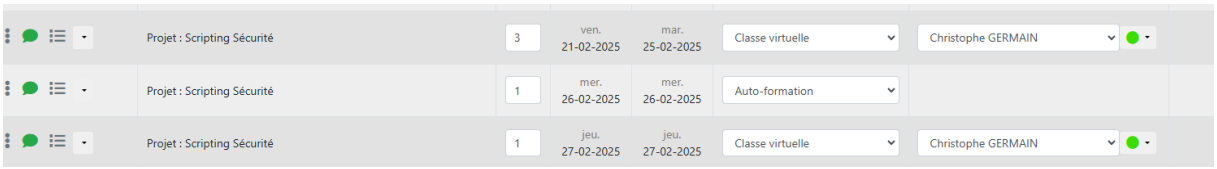
TP7 : Projet Scripting de Sécurité

Table des matières

Un projet combinant Python et Bash pour la sécurité peut être très puissant, car vous pouvez tirer parti de la simplicité de Bash pour des tâches système et utiliser Python pour des fonctionnalités plus complexes et flexibles.

I. Planning du projet :



II. Groupes :

- 5 groupes de 2

III. Date du rendu et organisation de la soutenance :

- Livrables à rendre le jeudi 27/02/2025 à 18h00
- Soutenance le 27/02/2025 durée présentation de 20 minutes + 10 de questions
  - Groupe 1 : 14h00
  - Groupe 2 : 14h40
  - Groupe 3 : 15h20
  - Groupe 4 : 16h00
  - Groupe 5 : 16h40

IV. Livrables :

- Invité votre référent cgermain@diginamic.fr : votre repo git : pseudo : **germain72**
- Dans le repo :
  - Faire un **readme.md** avec :
    - La présentation du groupe,
    - Les guides d'installation, d'architecture, de méthodologie.
  - Les scripts Bash & Python,
  - Documentations annexes,
  - Les recommandations de sécurités et les tests.
  - Le PWP comportera 20 slides maxi présentant :
    - L'organisation de l'équipe et du projet
    - La présentation des méthodes utilisées
    - La présentation de l'architecture
    - Présenter les scripts Bash et Python (rôles avec commentaires)
    - Copie des écrans comme démos justifiants vos travaux
    - Votre feedback

V. Exemples de types de scripts : Python et Bash

A. Script d'analyse et de surveillance de fichiers sensibles avec Python et Bash

Ce projet peut intégrer Bash pour la surveillance des fichiers sensibles et Python pour générer des rapports plus détaillés, envoyer des alertes par email, ou enregistrer les résultats dans une base de données.

- **Description** : Utilisez Bash pour surveiller les fichiers sensibles comme /etc/passwd, /etc/shadow, ou des fichiers de configuration, puis utilisez Python pour analyser les résultats et générer un rapport détaillé.
- **Bash** : Vérifiez les changements dans les fichiers, utilisez sha256sum ou md5sum pour générer des hachages.
- **Python** : Analysez les journaux ou les hachages et envoyez des alertes si des changements sont détectés. Vous pouvez aussi stocker les résultats dans un fichier JSON ou dans une base de données SQL pour un audit ultérieur.

1. Exemple de flux de travail :

- **Bash :** Vérifie les fichiers et génère un fichier de hachage.
- **Python :** Charge le fichier de hachage et le compare avec les précédents, puis génère un rapport en cas de modification.

B. Outil de gestion de mots de passe avec Python et Bash

Créez un gestionnaire de mots de passe utilisant Python pour la logique principale et Bash pour les tâches système comme l'automatisation du chiffrement des fichiers.

- **Description** : Le gestionnaire de mots de passe peut permettre à un utilisateur de stocker et récupérer ses mots de passe de manière sécurisée.
- **Python** : Utilisez Python pour chiffrer/déchiffrer les mots de passe en utilisant des bibliothèques comme cryptography ou pycryptodome.
- **Bash** : Utilisez Bash pour automatiser la gestion des fichiers chiffrés, créer des sauvegardes régulières et les stocker dans un endroit sécurisé.

1. Exemple de flux de travail :

- **Python :** Gère la logique de stockage et de récupération des mots de passe (chiffrement, déchiffrement).
- **Bash :** Automatise les sauvegardes et la gestion des fichiers chiffrés.

C. Détection des ports ouverts et des services vulnérables avec Python et Bash

Ce projet utilise Bash pour interagir avec des outils comme nmap ou ss pour scanner les ports, et Python pour analyser les résultats et identifier des vulnérabilités connues.

- **Description** : Le script pourrait scanner les ports ouverts sur une machine, et vérifier si des services vulnérables sont en cours d'exécution. Python permettrait d'analyser les résultats et d'envoyer des alertes si des services non sécurisés sont détectés.
- **Bash** : Utilisez des commandes comme nmap, netstat, ou ss pour scanner les ports et services.
- **Python** : Analyse les résultats du scan pour vérifier si les services sont vulnérables (par exemple, des versions de service connues pour avoir des failles de sécurité).

1. Exemple de flux de travail :

- **Bash :** Exécute un scan avec nmap pour identifier les ports ouverts.
- **Python :** Analyse les services trouvés, compare les versions avec une base de données de vulnérabilités, et génère des alertes si un service vulnérable est détecté.

D. Automatisation de la gestion des utilisateurs et de la sécurité des mots de passe

Un script Python et Bash pour gérer la création, la modification et la suppression d'utilisateurs, tout en vérifiant la sécurité des mots de passe selon certaines politiques.

- **Description** : Ce projet pourrait automatiser la gestion des utilisateurs tout en appliquant des politiques de sécurité comme l'expiration régulière des mots de passe, ou la vérification de la complexité des mots de passe.
- **Bash** : Utilisez useradd, usermod, et passwd pour créer et modifier des utilisateurs.
- **Python** : Vérifiez si les mots de passe sont conformes à des règles de sécurité (longueur, complexité, etc.) et générez des rapports.

1. Exemple de flux de travail :

- Bash : Crée un nouvel utilisateur avec des permissions spécifiques.
- Python : Vérifie la complexité du mot de passe, force l'utilisateur à changer son mot de passe après un certain temps et génère un rapport de conformité.

E. Script de surveillance des logs de sécurité (auth.log)

Utilisez un script Bash pour surveiller les fichiers de logs de sécurité (auth.log) et un script Python pour analyser les échecs de connexion et détecter des tentatives d'attaque.

- **Description** : Le script pourrait surveiller les échecs de connexion, détecter des comportements suspects comme des tentatives multiples de connexion échouée, puis générer des alertes.
- **Bash** : Utilisez grep, awk, et tail pour surveiller en temps réel le fichier de log.
- **Python** : Analyse les échecs de connexion pour générer des rapports détaillés et avertir l'administrateur du système en cas d'attaque potentielle.

1. Exemple de flux de travail :

- Bash : Surveille en temps réel auth.log pour détecter les échecs de connexion.
- Python : Analyse les événements et envoie une alerte par e-mail ou génère un rapport détaillé.

F. Automatisation des mises à jour de sécurité du système

Un projet qui utilise Bash pour mettre à jour automatiquement le système, et Python pour envoyer des rapports sur l'état des mises à jour et sur les éventuelles vulnérabilités corrigées.

- **Description** : Ce projet pourrait être un outil d'automatisation des mises à jour, qui s'assure que tous les packages de sécurité sont mis à jour régulièrement et génère un rapport détaillé sur les mises à jour appliquées.
- **Bash** : Exécute des commandes comme apt-get update ou yum update pour mettre à jour le système.
- **Python** : Analyse les logs des mises à jour et envoie un rapport par email à l'administrateur.

1. Exemple de flux de travail :

- Bash : Exécute des commandes de mise à jour système.
- Python : Récupère les logs de mise à jour et génère un rapport de sécurité.

VI. Projet à effectuer étape par étape :

A. Automatisation des Tests de Sécurité avec Kali Linux et le Scripting Bash / Python

1. Contexte

Kali Linux est une distribution spécialement conçue pour les tests d'intrusion et la sécurité informatique. Grâce à sa richesse en outils, Kali Linux est un choix privilégié pour les professionnels de la cybersécurité. Dans ce cas d'étude, nous allons automatiser certaines tâches de sécurité à l'aide de scripts Bash. Nous traiterons de tâches courantes telles que l'analyse des vulnérabilités, la gestion des utilisateurs, et la détection d'intrusions.

2. Objectif

L'objectif est de concevoir un script Bash qui peut exécuter les tests de sécurité de manière autonome. Ce script permettra de :

Scanner un réseau à la recherche de ports ouverts et de services vulnérables.

Vérifier la configuration de sécurité d'un système Linux (mots de passe, permissions, etc.).

Automatiser l'analyse de logs pour détecter des activités suspectes.

3. Prérequis

Avant de commencer, il est nécessaire d’avoir un environnement Kali Linux installé, avec des outils comme nmap, nikto, hydra, et fail2ban. Vous aurez également besoin de privilèges root pour effectuer certaines actions de sécurité.

B. Scanner les Ports et Services Ouverts avec nmap

L'un des premiers tests que l'on effectue lors d'un audit de sécurité est un scan des ports ouverts sur un hôte ou un réseau. Cela permet de savoir quelles portes sont accessibles à partir de l'extérieur.

1. **Créer le script Bash**
2. **Créer le script en Python**
3. **Donner l'explication du comportement et des résultats**

C. Vérifier les Mots de Passe et la Politique de Sécurité

La sécurité d'un système Linux dépend beaucoup de la gestion des utilisateurs et de leurs mots de passe. Vérifions si des mots de passe faibles sont utilisés, ou si des utilisateurs inutiles sont présents. (Voir V.B Outil de gestion de mots de passe avec Python et Bash).

1. **Créer le script Bash : (Check lors de la création des mots de passe)**
2. **Créer le script en Python (scanner pour les utilisateurs inutiles)**
3. **Donner l'explication du comportement et des résultats**

D. Analyser les Logs pour Détecter les Intrusions

Une autre tâche de sécurité consiste à analyser les fichiers de log pour identifier les signes d'une intrusion ou d'une tentative d'accès non autorisé. Le script suivant analyse les fichiers /var/log/auth.log et /var/log/syslog à la recherche de tentatives de connexion échouées.

1. **Créer le script Bash**
2. **Créer le script en Python**
3. **Donner l'explication du comportement et des résultats**

E. Automatisation avec Cron

Pour rendre ces vérifications plus efficaces, on peut automatiser l'exécution de ces scripts à l'aide de Cron. Cela permet de programmer les scans à intervalles réguliers pour détecter rapidement toute vulnérabilité ou anomalie.

1. **Ouvrir crontab**
2. **Ajouter tous les jours à 2h du matin pour lancer les scripts Bash créés dans les chapitres B, C & D.**
3. **Ajouter tous les jours à 8h30 du matin pour lancer les scripts python créés dans les chapitres B, C & D.**