Projet 1 - Surveillance et gestion des processus suspects

Résumé :

Développer un script en temps réel qui surveille les processus en cours d'exécution sur un système Unix, détecte des anomalies potentielles (comme des processus cachés ou une utilisation CPU élevée), et génère un rapport de sécurité.

Étapes du projet :

1. **Surveillance des processus :**
    - Utiliser les commandes `ps`, `top`, et `kill` pour lister et surveiller les processus en cours.
    - Créer un script Bash qui capture des informations telles que l'ID de processus (PID), l'utilisation de la mémoire, l'utilisation CPU, l'utilisateur qui exécute le processus, et l'état du processus.
    - Générer un fichier journal contenant l'état actuel des processus (mis à jour toutes les X secondes).

2. **Détection d'anomalies :**
    - Développer un module de détection d'anomalies dans le script. Celui-ci doit inclure des critères tels que :
        - Processus utilisant plus de 80% du CPU sur une période prolongée.
        - Processus exécuté par un utilisateur non habituel (par exemple, un processus système exécuté par un utilisateur non root).
        - Processus sans parent (zombie) ou caché.
    - Lorsque l'une de ces conditions est remplie, le script doit générer une alerte dans le journal et envoyer une notification (par exemple via un email ou un message sur la console).
      Chaque entrée du journal doit inclure au minimum les informations suivantes :
        - Horodatage : La date et l'heure de l'événement.
        - Type d'anomalie : La nature de l'anomalie détectée (par exemple, "Utilisation CPU élevée", "Processus zombie", "Processus exécuté par un utilisateur non autorisé").
        - PID et nom du processus : Le numéro d'identification (PID) et le nom du processus concerné.
        - Utilisateur : L'utilisateur ayant lancé le processus.
        - Détail de l'anomalie : Informations spécifiques à l'anomalie (par exemple, pour une utilisation CPU élevée, indiquer le pourcentage de CPU utilisé).
        - Action entreprise : Ce que le script a fait en réponse (ex: "Processus tué", "Priorité réduite", "Notification envoyée").
        - Statut de la notification : Si un email ou une alerte console a été envoyée et à qui.

      Le journal peut être enregistré sous forme de fichier texte (par exemple `/var/log/process_monitor.log`).

3. **Gestion des processus suspects :**
    - Permettre à l'utilisateur de choisir l'action à entreprendre via une interface simple (du terminal). Ex: tuer le processus avec `kill`, ou baisser la priorité avec `renice`.
      Exemple plus détaillé :

```
Anomalie détectée : Utilisation CPU élevée
Processus : PID 1234 (python)
```

```
Utilisateur : user1
CPU : 85%

Quelle action souhaitez-vous entreprendre ?
1. Tuer le processus
2. Baisser la priorité (renice)
3. Ignorer

Entrez votre choix (1/2/3) :
```

- Inclure une option pour sauvegarder les informations sur le processus suspect avant de le terminer (pour une future analyse). Vous devez, par exemple :
  - Collecter les informations sur le processus : Utiliser des commandes comme `ps`, `top`, ou `pmap` pour récupérer les informations nécessaires avant de tuer le processus.
  - Stocker ces informations dans un fichier : Écrire les informations collectées dans un fichier log dédié, par exemple `/var/log/process_suspects.log`.
  - Exécuter l'action (tuer le processus) : Après avoir sauvegardé les informations, le script peut procéder à l'action choisie (par exemple, tuer le processus).

4. **Génération de rapports :**
   - Le script doit générer un rapport quotidien incluant les processus surveillés, les anomalies détectées, et les actions entreprises.
   - Le rapport devra inclure des statistiques comme la fréquence des anomalies, les utilisateurs les plus souvent impliqués, et des recommandations pour améliorer la sécurité du système.

Simulation d'anomalies :

Pour tester le bon fonctionnement de votre script, **vous devrez simuler des anomalies**.

Voici quelques suggestions :

- **Simulation d'une utilisation CPU élevée :** Utilisez des commandes comme `stress` (ou créez un programme en Python ou Bash qui boucle indéfiniment pour générer une charge CPU) pour simuler un processus consommant beaucoup de ressources.
- **Processus exécuté par un utilisateur inhabituel** : Lancez un processus système normalement réservé à `root`, mais cette fois exécuté par un utilisateur non autorisé pour déclencher une alerte.
- **Processus sans parent (zombie) ou caché** : Simulez des processus zombies ou masqués en utilisant des techniques pour laisser un processus orphelin ou dissimuler son exécution.