Rapport de projet sécurité OS

Nom du projet : “Projet monitoring file system”
Membre du groupe :

- LASSOUED Hamza
- DAIDOU Issam
- GOMIS Kwency
- KONTE Amara
- MBEMBA MIAMISSA Yann

Introduction

Ce projet vise à créer un outil de surveillance des propriétés des fichiers et/ou dossier et de gestion des droits d'accès de ces derniers sur les systèmes d'exploitation Linux.

L'objectif principal est d'aider les administrateurs système à suivre les modifications apportées aux fichiers et/ou dossier et à gérer les autorisations d'accès, garantissant ainsi la sécurité et l'intégrité des données.

La gestion des droits d'accès aux fichiers est un aspect crucial de la sécurité informatique. Comprendre comment accorder et révoquer l'accès au fichiers et/ou dossiers permet de préserver l'intégrité des données et d'empêcher les accès non autorisés.

En voici quelques points essentiels sur l'importance de cette gestion :

- **Contrôle des accès :** les droits d'accès définissent la possession d'un fichier ou d'un dossier pour un utilisateur et un groupe d'utilisateurs. Ils permettent également de spécifier quelles actions les utilisateurs sont autorisés à effectuer sur les fichiers (lecture, écriture et exécution), en fonction de leur statut (propriétaire, membre du groupe propriétaire ou autre). La gestion fine des permissions garantit que seules les personnes autorisées peuvent accéder aux données.

- **Protection des données :** en limitant l'accès aux fichiers, on réduit les risques de fuites de données ou de corruption. Par exemple, un fichier contenant des informations sensibles ne devrait être accessible qu'à un nombre restreint d'utilisateurs autorisés.

- **Principe du moindre privilège :** la gestion des droits d'accès suit le principe du moindre privilège. Cela signifie que chaque utilisateur doit avoir uniquement les permissions nécessaires pour accomplir ses tâches. Ainsi, même si un compte est compromis, les dégâts potentiels sont limités.

- **Audit et traçabilité :** en attribuant des droits d'accès de manière structurée, il est plus facile de suivre les actions effectuées sur les fichiers. Les journaux d'audit permettent de détecter toute activité suspecte ou non autorisée.

En somme, la gestion des droits d'accès contribue à la sécurité globale des systèmes en protégeant les données et en limitant les vulnérabilités.

Objectifs du projet

Voici les fonctionnalités à implémenter :

- Surveillance des Propriétés des Fichiers :
    - Mettre en place un mécanisme de surveillance pour détecter les modifications apportées aux propriétés des fichiers, telles que les autorisations d'accès, les propriétaires, les dates de modification, etc.
    - Enregistrer les événements de modification dans un journal système (log) pour une analyse ultérieure. Cela permettra de suivre les changements et d'identifier toute activité suspecte.
- Ajout ou Suppression des Fichiers à Surveiller :
    - Implémenter une fonctionnalité permettant d'ajouter le chemin vers un fichier afin de surveiller ses propriétés.
    - Prévoir également la suppression d'un chemin si un fichier n'a plus besoin d'être surveillé.
- Gestion des Droits d'Accès :
    - Développer une interface utilisateur conviviale permettant aux administrateurs système de modifier les autorisations d'accès aux fichiers.
    - Offrir la possibilité de définir les autorisations par utilisateur, groupe d'utilisateurs et autres critères de sécurité. Cela permettra de personnaliser les droits d'accès en fonction des besoins spécifiques.
- Sécurité et Intégrité des Données :
    - Mettre en place des mécanismes de sécurité pour empêcher les accès non autorisés aux fichiers sensibles.
    - Intégrer des fonctionnalités de vérification d'intégrité pour détecter les modifications non autorisées des fichiers. Cela garantira que les données restent intactes et non altérées.
- Interface Utilisateur :
    - Concevoir une interface utilisateur conviviale pour :
        - Surveiller les événements de modification des fichiers.
        - Afficher les autorisations d'accès actuelles.
        - Modifier les autorisations au besoin.

En fin de compte, les livraisons attendues pour ce projet sont un outil fonctionnel de surveillance et de gestion des droits sur les fichiers, ainsi qu'un rapport de projet détaillant l'architecture du système, les fonctionnalités implémentées et les résultats des tests de sécurité.

Technologies utilisées

Le programme Python que nous avons développé repose sur plusieurs technologies essentielles pour sa mise en œuvre.

- **Le module os :** il nous permet d'interagir avec le système d'exploitation. Nous l'utilisons pour effectuer des opérations telles que l'exécution de commande Linux a partir de Python, la gestion des chemins des fichiers et/ou dossier.
- **Les modules pwd et grp :** ils nous permettent d'accéder aux informations sur les utilisateurs et les groupes du système. Nous pouvons ainsi récupérer la liste de tous les utilisateurs ainsi que tous les groupes sur le système.
- **Le module logging :** il facilite la journalisation des événements dans le fichier de logs. Nous l'utilisons pour enregistrer des informations importantes concernant les événements concernant les fichiers et/ou dossiers (création, modification, suppression,...).
- **Le module datetime :** il nous permet de manipuler les dates et les heures. Nous l'utilisons pour obtenir les horodatages des événements sur le système.
- **La bibliothèque watchdog :** elle fournit des outils pour surveiller les modifications des fichiers et des dossiers. Nous utilisons les classes **Observer** et **FileSystemEventHandler** pour détecter les événements tels que la création, la modification ou la suppression de fichiers et/ou dossiers.
- **La bibliothèque tkinter :** elle est utilisée pour créer une interface utilisateur graphique (GUI). Nous l'avons intégrée pour permettre aux administrateurs système d'interagir avec l'outil de surveillance et de gestion des droits d'accès.
- **Le module multiprocessing :** il nous permet d'implémenter des processus parallèles. Nous l'utilisons pour gérer la surveillance continue des fichiers tout en maintenant la réactivité de l'interface utilisateur.

En combinant ces technologies, notre programme offre une solution robuste pour surveiller les propriétés des fichiers et gérer les droits d'accès sur les systèmes d'exploitation Linux.

Conception et architecture

Interface graphique (GUI) avec Tkinter :

Lors de la conception d'une application Python dotée d'une interface graphique à l'aide de Tkinter, on suit une structure spécifique. Voici les éléments clé de cette structure :

- **Fenêtre Principale (**communément appelé **Root Window) :**
  - La fenêtre principale est la base de l'interface graphique. Elle est initiée en créant une instance de la classe Tk de Tkinter.
  - Cette fenêtre est le conteneur principal pour tous les autres widgets (éléments graphiques) qu'on ajoutera.

- **Les widgets**
  - Les widgets sont les éléments de l'interface graphique tels que les boutons, les titres, les listes,...
  - Après avoir créé les widgets, on les ajoute à la fenêtre principale à l'aide de méthode tel que pack() qui est souvent utilisé dans cette application.

- **Boucle Principale (Mainloop) :**
  - Après avoir créé la fenêtre principale et ajouter des widgets (boutons, titres, listes,...), on doit lancer la boucle principale en appelant la méthode mainloop().
  - La boucle principale écoute les événements provenant de l'utilisateur (clics de souris, saisie au clavier, etc.).
  - Tant que la boucle principale est active, l'application reste réactive et peut gérer les interactions avec l'utilisateur.

- **Gestion des événements :**
  - Tkinter gère les événements (par exemple, clics de souris, touches du clavier pressées) via des gestionnaires d'événements.
  - On peut associer des fonctions (appelées callbacks) aux événements spécifiques d'un widget.
  - Par exemple, lorsque l'utilisateur clique sur un bouton, le gestionnaire d'événements appelle la fonction associée.
  - Cette fonctionnalité est largement utilisée pour accomplir ce que l'utilisateur a demandé en prenant en compte ses choix et les données qu'il a fournies.

- **Organisation des Widgets :**
    - On peut organiser les widgets dans des fenêtres distinctes (frames).
    - Les fenêtres servent de conteneurs pour regrouper les widgets.

- **Styles et Mises en Forme :**
    - On peut personnaliser l'apparence des widgets en utilisant des styles (couleurs, taille de la police, style de la police, etc.).

Surveillance des événements système avec watchdog :

Watchdog est une bibliothèque Python qui permet de surveiller les événements du système de fichiers, tels que les modifications, les ajouts ou les suppressions de fichiers. C'est l'outil parfait pour la surveillance des propriétés des fichiers et de gestion des droits d'accès sur les systèmes d'exploitation Linux. Voici comment nous l'avons intégré à notre application :

- Au moment où un chemin est configuré pour être surveillé, une classe est invoquée, et chaque fois qu'un événement se produit (tel que la création ou la suppression), l'une des fonctions ci-dessous est appelée :
    - **on_created :** lorsque watchdog détecte un événement de création, qu'il s'agisse d'un fichier ou d'un dossier, cette fonction est exécutée, ce qui entraîne l'écriture d'une ligne dans le fichier de journalisation (logs) indiquant la création d'un fichier ou d'un dossier, avec la date et l'heure de l'événement.
    - **on_moved :** lorsque watchdog détecte un événement de déplacement, qu'il s'agisse d'un fichier ou d'un dossier, cette fonction est exécutée, ce qui entraîne l'écriture d'une ligne dans le fichier de journalisation (logs) indiquant le déplacement d'un fichier ou d'un dossier en précisant l'ancien et le nouveau chemin, avec la date et l'heure de l'événement.
    - **on_deleted :** lorsque watchdog détecte un événement de suppression, qu'il s'agisse d'un fichier ou d'un dossier, cette fonction est exécutée, ce qui entraîne l'écriture d'une ligne dans le fichier de journalisation (logs) indiquant la suppression d'un fichier ou d'un dossier, avec la date et l'heure de l'événement.
    - **on_modified :** lorsque watchdog détecte un événement de modification, qu'il s'agisse d'un fichier ou d'un dossier, cette fonction est exécutée, ce qui entraîne l'écriture d'une ligne dans le fichier de journalisation (logs) indiquant la modification d'un fichier ou d'un dossier, avec la date et l'heure de l'événement, et en précisant :
        - Le nouvel utilisateur et groupe propriétaire du fichier ou dossier
        - Les nouveaux droits de l'utilisateur, du groupe et des autres

- La date et l'heure de la dernière modification du fichier ou dossier
- La date et l'heure du dernier accès au fichier ou dossier
- La taille du fichier

La journalisation avec logging :

Le module de journalisation en Python est un outil essentiel pour suivre les événements qui se produisent pendant l'exécution du programme, ainsi que pour garder un historique en enregistrant ces événements dans un fichier. Dans le contexte d'un outil de surveillance des propriétés des fichiers et de gestion des droits d'accès sur les systèmes d'exploitation Linux, cet outil est essentiel car il permet :

- L'enregistrement des événements pour les futurs diagnostics en cas de problème, ces événements permettent d'avoir un historique pour retracer les actions effectuées.
- La surveillance en temps réels qui permet de détecter rapidement les problèmes et les comportements anormaux

L'exécution de commande système avec os :

Le module os en Python est un outil essentiel pour interagir avec les fonctionnalités du système d'exploitation. Il permet de gérer l'arborescence des fichiers, d'accéder à des informations système et d'effectuer diverses opérations liées aux fichiers et aux dossiers. Ce module est essentiel car il permet :

- La vérification de l'existance d'un fichier ou d'un dossier
- La vérification si un chemin correspond à un fichier ou à un dossier
- L'obtention du dossier parent d'un fichier
- La récupération des propriétés d'un fichier ou dossier (propriétaire, droits, taille,...)
- Le changement des propriétés d'un fichier ou dossier en exécutant des commandes système tel que : chmod et chown

Les processus en parallèles avec multiprocessing :

Le module multiprocessing en Python joue un rôle essentiel dans la création de processus parallèles et la gestion de tâches simultanées. Les avantages de l'utilisation de ce module sont :

- L'isolation : chaque processus a son propre espace mémoire, ce qui évite les conflits de données entre les tâches.

- Il évite le verrouillage global de l'interpréteur (GIL) : contrairement au multithreading, les processus créés avec multiprocessing ne sont pas limités par le GIL, ce qui permet une utilisation efficace des ressources multicœurs.
- La robustesse : en cas d'erreur dans un processus, les autres processus ne sont pas affectés.

Fonctionnement

Utilisation