# Informe Proyecto HIDS

## Introducción

El Sistema de Detección de Intrusos Basado en el Host (HIDS) es una herramienta diseñada para monitorear y proteger la integridad de archivos en un sistema informático. Este informe proporciona una visión general de las decisiones clave, soluciones implementadas y resultados de pruebas del programa HIDS desarrollado.

## Esbozo del proyecto

La idea principal para este HIDS era que tenía que ser escalable y, por lo tanto, las decisiones de Diseño y de Estructura debían ir acordes a estas expectativas, por ello podríamos decir que los 3 pilares de este proyecto informático son los siguientes:

1. **Base de Datos SQLite:** Se eligió utilizar una base de datos SQLite para almacenar los hashes y marcas de tiempo de los archivos. SQLite es una base de datos incorporada en Python que ofrece un rendimiento eficiente y no requiere una configuración compleja.
2. **Hashing SHA-1:** El programa utiliza el algoritmo SHA-1 para calcular los hashes de los archivos. Aunque SHA-1 no es la opción más segura en términos de criptografía, como ya se ha visto en clases de teoría, es adecuado para detectar cambios en los archivos, asumiendo que el atacante no tiene la capacidad de vulnerar este tipo de firma.[[1]](#footnote-1)
3. **Configuración Externa:** Se implementó un archivo de configuración externa llamado "config.ini" para permitir la configuración dinámica de los directorios a monitorear. Esto facilita la adaptabilidad del programa a diferentes entornos.

El programa en sí tiene 3 opciones una vez se ejecuta:

1. **Agregar Directorio:** El programa permite a los usuarios agregar directorios que desean monitorear. Los directorios se almacenan en una lista y se actualizan en el archivo de configuración "config.ini".
2. **Inicializar Base de Datos:** Al seleccionar esta opción, el programa crea una base de datos SQLite y calcula los hashes de todos los archivos en los directorios especificados. Los resultados se almacenan en la base de datos.
3. **Actualizar HIDS:** Esta función compara los hashes almacenados en la base de datos con los hashes recalculados de los archivos en los directorios monitoreados. Si se encuentra una diferencia, se considera una modificación y se registra en un informe mensual. Esta opción también nos sirve para actualizar la propia base de datos también si llegase a encontrar un archivo en ese directorio que no estuviera en la base de datos

### Problemas y preocupaciones

Una de las principales preocupaciones que he tenido a la hora de diseñar el HIDS es la posibilidad de una modificación indebida del archivo config.ini o de la propia base de datos. He pensado en la posibilidad de encriptar esta base de datos después de cada operación que se hiciera en esta, y en consecuencia se desencriptara también el momento de antes de hacer cualquier operación; aunque debido a un problema en inicio de tipo técnico (todas las implementaciones de encriptado/desencriptado de SQLite que he encontrado están pensadas o exclusivamente, o sin mayores problemas para sistemas UNIX), que se tornó en un problema de falta de tiempo debido a que no he recibido ninguna ayuda de mis compañeros para este proyecto, no he sido capaz de solucionar esta vulnerabilidad.

Igualmente, a modo de “parche” se me ocurrió que también podía añadir a la base de datos una entrada para config.ini y otro para la base de datos, de tal forma que, si no puedo impedir que no modifiquen el archivo, por lo menos, debería de ser capaz de detectar esta modificación; esto sí que está implementado.

Por otra parte, respecto al objetivo nº 3 “El proceso de verificación se realizará a intervalos, en este caso diariamente y debe almacenarse un informe de un mes entero.”, se ha logrado generar un informe que vaya por cada mes, y que en este se registre solamente los archivos modificados indebidamente, pero no se ha conseguido automatizar este proceso, ya que para ello se requieren de permisos especiales que nuevamente son más complicados de conseguir en Windows.

## Explicación detallada del programa

Una vez comentado más o menos por encima el proyecto y sus dificultades, voy a explicar las funciones más importantes de este programa:

1. **Configuración y Preparación Inicial:**
   * El programa comienza importando las bibliotecas necesarias y configurando las rutas de archivos y directorios.
   * Se utiliza un archivo de configuración externa llamado "config.ini" para almacenar los directorios a monitorear.  
     Texto

     Descripción generada automáticamente
2. **Base de Datos SQLite:**
   * Se utiliza una base de datos SQLite para almacenar los hashes y marcas de tiempo de los archivos monitoreados.
   * La función **create\_database** crea la tabla "hashes" en la base de datos si no existe.  
     Texto

     Descripción generada automáticamente
3. **Cálculo de Hashes:**
   * La función **hashFile** se utiliza para calcular el hash SHA-1 de un archivo dado.  
     Texto

     Descripción generada automáticamente
   * Las funciones **hashFiles** y **compareHashes** se encargan de calcular y comparar los hashes de los archivos en los directorios especificados, respectivamente.  
     Texto

     Descripción generada automáticamente  
     Texto

     Descripción generada automáticamente
4. **Gestión de Directorios y Configuración:**
   * El programa permite a los usuarios agregar directorios a monitorear y los almacena en una lista.
   * La función **crearOActualizarConfigEntry** [[2]](#footnote-2)actualiza la base de datos con información sobre el archivo de configuración y la base de datos en sí.  
     Texto

     Descripción generada automáticamente
   * La funcion **agregarDirectorio** nos permite añadir un nuevo directorio al archivo config.ini
5. **Informe de Modificaciones:**
   * El programa registra las modificaciones detectadas en un informe mensual. La función **registrarModificacionEnInforme** se encarga de esto.  
     Pantalla de computadora con letras

     Descripción generada automáticamente con confianza media
6. **Menú de Opciones:**
   * Se implementa un menú de opciones interactivo que permite a los usuarios realizar las siguientes acciones:
     + Agregar directorios.
     + Inicializar la base de datos y calcular los hashes iniciales.
     + Actualizar el HIDS y detectar modificaciones.
     + Salir del programa.  
         
       Texto

       Descripción generada automáticamente

## Uso del Programa

1. Lo primero que haremos será elegir la opción 1, una vez se introduce el directorio, se puede ver que se notifica de la creación de dos nuevas entradas en la db.  
   Texto

   Descripción generada automáticamente
2. Ahora vamos a inicializar la DB.  
   Texto

   Descripción generada automáticamente
   1. Vemos a continuación que en efecto la DB ha sido creada.  
      Texto

      Descripción generada automáticamente
3. Ahora vamos a actualizar el HIDS sin haber hecho ningún cambio.  
   Texto

   Descripción generada automáticamente
   1. Vamos a ver qué es lo que pasaría si hacemos un cambio en un archivo.  
      Texto

      Descripción generada automáticamente  
      Se va a crear entonces el siguiente archivo de Informe para el mes  
      Interfaz de usuario gráfica, Texto

      Descripción generada automáticamente  
      Interfaz de usuario gráfica, Texto

      Descripción generada automáticamente  
      Este aviso no se quitará hasta que no volvamos a seleccionar la opción 2[[3]](#footnote-3):  
      Texto

      Descripción generada automáticamente

Como se ha visto, si bien no podría considerarse una implementación completa o profesional de un HIDS, es una aproximación bastante fiable al concepto de este, tanto su eficiencia a la hora de buscar (con la ruta de los directorios guardada en config.ini solo se tarda lo que tarde el equipo en leer la información en el disco), calcular (si bien SHA-1 no es seguro a la hora de encriptar archivos, puede sernos útil para este cometido, y de esta forma no sacrificar el tiempo de procesamiento) y a la hora de almacenar (usando SQLite podemos escalar cuanto sea necesario esta base de datos y usarla en las comprobaciones)

1. Se ha intentado con el algoritmo SHA-256 y SHA-512, pero los tiempos de espera para hashear tal volumen de archivos se disparaba considerablemente [↑](#footnote-ref-1)
2. Esto es lo que se ha comentado anteriormente, gracias a esta función podemos saber si ha sido modificada la db o el archivo config.ini de forma ilícita [↑](#footnote-ref-2)
3. Sé que el nombre de la opción puede inducir a dudas [↑](#footnote-ref-3)