**PAI. 2**

VERIFICADORES DE INTEGRIDAD EN LA   
TRANSMISIÓN PUNTO-PUNTO PARA ENTIDAD   
FINANCIERA

El Documento se realiza a requerimiento de Profesor Ángel Jesús Varela Vaca de la asignatura de SSII de la Universidad de Sevilla con el fin de proceder a presentar el informe final del proyecto titulado “Verificadores De Integridad En La   
Transmisión Punto-Punto Para Entidad   
Financiera”.

**Realizado por:**

François Barberot

Gustavo Antonio Souza De Barros

Pepe Ortiz Roldan

**Fecha:**

22/10/23

**HOJA DE CONTROL DEL DOCUMENTO**

Documento: **PAI1-STXTrabajoY**

Denominación:  **Sistema De Detección De Intrusos (HIDS) Para Almacenamiento Masivo Basado En Verificadores De Integridad**

Edición: **01**

Fecha: **8** de octubre 2023

Realizado por: **François Barberot y Gustavo Antonio Souza De Barros**

Aprobado por: Ángel Jesús Varela Vaca (Universidad de Sevilla)

**CONTROL DE MODIFICACIONES**

|  |  |  |
| --- | --- | --- |
| Etapas | Fecha | Comentarios |
| 01 | 03/10 | introducción |
| 02 | 04/10 | Planificación de todo el proyecto (Diagrama de Gantt) |
| 03 | 04/10 | Diseño del sistema & Implementación |
| 04 | 05/10 | Plan de pruebas sobre cada método |
| 05 | 05/10 | Plan de pruebas para todo el sistema |
| 06 | 06/10 | Archivo de Incidencias de la medición |
| 07 | 07/10 | Conclusión & bibliografía / webgrafía |
| 08 | 08/10 | Corrección y finalización del informe |

**INDICE**

[1 Introducción 3](#_Toc147422143)

[2 Planificación 3](#_Toc147422144)

[3 Desarrollo del proyecto 4](#_Toc147422145)

[3.1 Diseño del sistema 4](#_Toc147422146)

[3.2 Implementación 5](#_Toc147422147)

[4 Pruebas del sistema desarrollado 5](#_Toc147422148)

[4.1 Plan de pruebas sobre cada método 5](#_Toc147422149)

[4.2 Plan de pruebas para todo el sistema 5](#_Toc147422150)

[4.3 Archivo de Incidencias de la medición 5](#_Toc147422151)

[5 Propuestas de mejoras 5](#_Toc147422152)

[6 Conclusiones 5](#_Toc147422153)

[7 Bibliografía y Webgrafía 5](#_Toc147422154)

# Introducción

Este documento constituye el informe detallado del proyecto relacionado con la creación de un HIDS (Host intrusión Detection System) titulado "Sistema De Detección De Intrusos (HIDS) Para Almacenamiento Masivo Basado En Verificadores De Integridad".

El objetivo principal de este proyecto es establecer un sistema capaz de detectar cambios sospechosos que puedan indicar un riesgo potencial mediante el monitoreo activo de las actividades en el host. Más específicamente, esto implica comprobar la integridad de los archivos binarios a diario.

¿Por qué optamos por utilizar un HIDS en lugar de un NIDS (sistema de detección de intrusiones en la red)? Esta decisión se basa en varias razones fundamentales:

Los HIDS están altamente capacitados para detectar amenazas internas.

Tienen la capacidad de asociar usuarios y programas con sus acciones específicas en el sistema.

Los HIDS se centran exclusivamente en proteger el host donde se implementan, limitando su huella en términos de recursos informáticos y minimizando el impacto en el rendimiento general del sistema.

Sin embargo, es esencial tener en cuenta las posibles desventajas inherentes al uso de un HIDS en lugar de un NIDS:

HIDS puede tener tiempos de respuesta más lentos en comparación con NIDS porque se basan en el análisis de registros de actividad y cambios en el sistema de archivos, lo que significa que detectan ataques después del hecho, cuando ya han ocurrido, potencialmente demasiado tarde para tomar medidas efectivas.

En caso de que se produzca un ataque importante que comprometa el host, HIDS se vuelve vulnerable y es posible que no pueda generar alertas.

# Planificación

Como parte de nuestro proyecto utilizamos una herramienta para la gestión del tiempo: el diagrama de Gantt. Este diagrama proporciona una descripción clara de nuestra planificación de proyectos, identificando tareas y sus cronogramas. Desempeña un papel esencial en la organización, coordinación y seguimiento de nuestras actividades.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tareas | 25 sep | 26 sep | 27 sep | 28 sep | 29 sep | 30 sep | 1 oct | 2 oct | 3 oct | 4 oct | 5 oct | 6 oct | 7 oct | 8 oct |
| Fase 1: Preparación del proyecto |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1.1 Definición de Objetivos |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1.2 Creación de la Base de Datos |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fase 2: Elección de la Estrategia HIDS |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2.1 Elección de la Estrategia HIDS |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fase 3: Desarrollo de HIDS personalizado |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3.1 Método para calcular Hash |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3.2 Lectura de Archivos desde la Base de Datos |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3.3 Comparación de Hashes |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3.4 Gestión de Registros en Archivos y Carpetas |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fase 4: Automatización y pruebas |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4.1 Automatización del Programa |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4.2 Pruebas del Programa |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fase 5: Documentación et informe |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5.1 Redacción del informe |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Tuvimos muchos problemas desde el inicio del proyecto. Tuvimos la opción de crear nuestro propio HIDS o elegir uno ya existente modificándolo para que satisfaga las necesidades del cliente.

Nos decidimos bastante rápido por la segunda opción, pensando que ahorraríamos tiempo. Sin embargo, fue una mala decisión de nuestra parte.

En primer lugar, tuvimos que investigar un poco en Internet para descubrir cuál de los HIDS mejor valorados satisfaría las necesidades de nuestros clientes. Elegimos OSEEC Entre los HIDS de OSSEC que no querían funcionar en nuestro ordenador, y los problemas que tenían mis compañeros con otros HIDS, abandonamos la idea de utilizar un HIDS ya funcional para finalmente utilizar nuestros propios HIDS.

# Desarrollo del proyecto

## Diseño del sistema

En la etapa de Concepción de la Arquitectura de nuestro sistema HIDS, hemos optado por utilizar Python como lenguaje de programación principal. Python es ampliamente conocido por su simplicidad y potencia, lo que lo convierte en una elección lógica para implementar las funcionalidades clave de nuestro sistema.

Además, hemos seleccionado VS Code como nuestro entorno de desarrollo debido a su popularidad y su elegante diseño. VS Code proporciona un entorno de desarrollo altamente versátil que facilita la escritura de código, la depuración y la gestión de proyectos, lo que será fundamental para la implementación exitosa de nuestro sistema.

## Implementación

El programa está dividido en 4 archivos Python diferentes. Esto se ha hecho para facilitar la lectura y comprensión del código.

El primero, "main.py" es el punto de entrada para el programa principal.

El secundo, llamado "vérification.py" es el archivo que contiene la lógica para la comprobación de la integridad. El tercero, "recap\_incidents.py", contiene la lógica para resumir los incidentes. El archivo que contiene la lógica para la respuesta del servidor es el último archivo llamado "server\_response.py".

Los métodos implementados son:

* customer\_send\_file & server\_receive\_file: son funciones que calculan el hash SHA-1 de un archivo. El hash SHA-1 es una representación única de los datos del archivo y se utiliza para verificar si el archivo ha cambiado desde la última verificación.
* compare\_hashes: compara los hashes del archivo del cliente y del servidor para determinar si son iguales. Si los hashes son iguales, significa que el archivo no ha cambiado.
* run\_verification: es la función central de verificación. Abre el archivo del cliente y el archivo del servidor, calcula los hashes y luego llama a “server\_response ()” para registrar el resultado de la verificación en un archivo de registro.
* schedule\_verification: utiliza la biblioteca “Schedule” para programar la verificación en intervalos regulares (cada cierto minuto).
* run\_recap: es responsable de realizar la recapitulación de incidentes. Cada vez que se ejecuta, crea un directorio para la hora actual y mueve todos los archivos de registro a ese directorio. Esto permite mantener un registro organizado de los incidentes por hora.
* schedule\_recap: programa la ejecución de run\_recap () en intervalos regulares (cada cierta hora) utilizando la biblioteca Schedule.
* server\_response:
  + registra la respuesta del servidor en un archivo de registro. Recibe tres argumentos: el archivo de registro donde se registrará la respuesta, un valor booleano que indica si la verificación fue exitosa (True) o fallida (False) y la ruta del archivo verificado.
  + La función registra la fecha y hora actual, la ruta del archivo y el resultado de la verificación (éxito o falla) en el archivo de registro correspondiente.

# Pruebas del sistema desarrollado

## Plan de pruebas sobre cada método

## Plan de pruebas para todo el sistema

## Archivo de Incidencias de la medición

# Propuestas de mejoras

# Conclusiones

# Bibliografía y Webgrafía