Reporte técnico: Proyecto final de Sistemas Operativos y Laboratorio

# Información del Proyecto

* **Título del Proyecto:** **Implementación de un clasificador Naive Bayes para detección de archivos maliciosos en un entorno de sistema operativo Linux**
* **Curso/Materia:** Sistemas Operativos
* **Integrantes:** Nelson Puerta ([alcides.puerta@udea.edu.co](mailto:alcides.puerta@udea.edu.co))

William Torres ([walexander.torres@udea.edu.co](mailto:walexander.torres@udea.edu.co))

Clara Villadiego ([clara.villadiego@udea.edu.co](mailto:clara.villadiego@udea.edu.co)).

* **Fecha de Entrega:** 10 Julio 2025

# 2. Introducción

# 2.1. Objetivo del Proyecto

# El objetivo principal de este proyecto es diseñar e implementar una herramienta en lenguaje C capaz de clasificar archivos como benignos o maliciosos utilizando el algoritmo de clasificación Naive Bayes, con base en características extraídas de archivos del sistema Linux. El proyecto busca demostrar la aplicabilidad de la inteligencia artificial en el contexto de los sistemas operativos, particularmente en seguridad.

Este sistema operará de manera local, sin necesidad de conexión a servicios externos ni grandes recursos computacionales. La meta es construir una solución ligera, confiable y adaptable, útil tanto para propósitos educativos como para su eventual integración en entornos de monitoreo de seguridad.

# 2.2. Motivación y Justificación

# Hoy en día, la ciberseguridad es una de las principales preocupaciones dentro del campo de los sistemas operativos. Aunque Linux es considerado robusto, no está exento de amenazas como rootkits, scripts maliciosos o backdoors disfrazados de archivos comunes. Este proyecto nace de la necesidad de explorar formas alternativas y eficientes de protección, sin depender exclusivamente de herramientas comerciales o bases de datos de firmas. Al usar el algoritmo Naive Bayes —conocido por su bajo costo computacional y efectividad en clasificación de texto y documentos— se busca demostrar su aplicabilidad en entornos donde los recursos son limitados o donde se requiere una solución embebida. Además, al implementar este proyecto en C, se refuerzan conceptos clave de bajo nivel como manejo de memoria, acceso a archivos y procesamiento eficiente, fortaleciendo habilidades fundamentales en la ingeniería de software y sistemas.

# 2.3. Alcance del Proyecto

Este proyecto abarca el desarrollo completo de una aplicación de consola en C que:

* Extrae características relevantes de archivos del sistema Linux.
* Usa dichas características para alimentar un modelo Naive Bayes pre entrenado.
* Clasifica archivos nuevos como benignos o maliciosos.
* Imprime los resultados de clasificación y justificación probabilística.
* Permite evaluar nuevos archivos individualmente o en lotes.

Fuera del alcance quedan funcionalidades como:

* Entrenamiento dinámico del modelo desde la interfaz.
* Soporte para clasificación multiclase.
* Integración gráfica o web.
* Protección activa del sistema (es un detector, no un firewall).

# 3. Marco Teórico / Conceptos Fundamentales

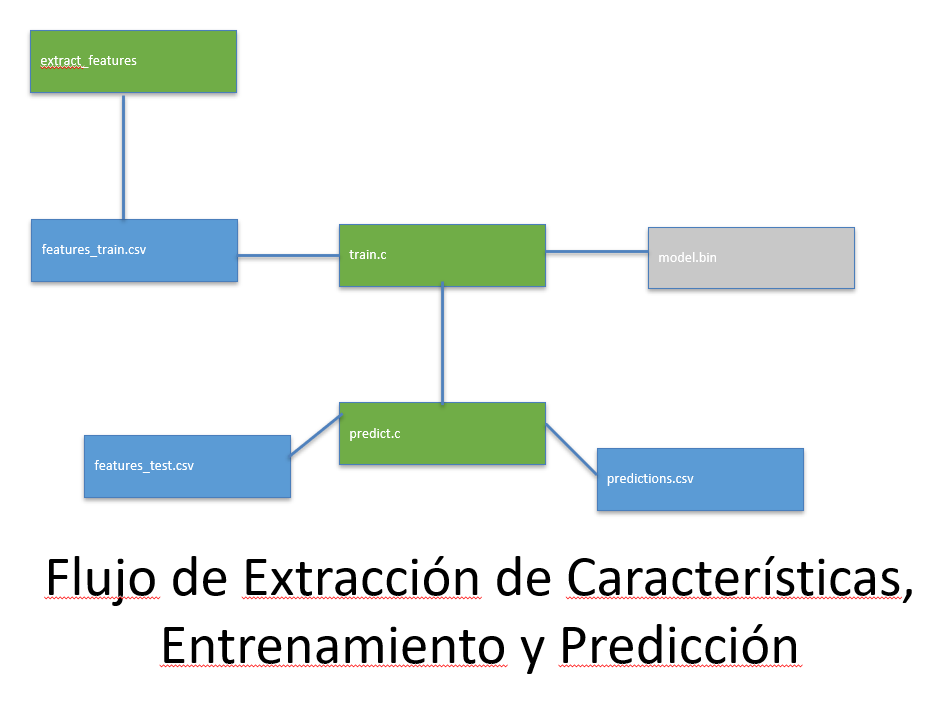
* **Naive Bayes:** Es un algoritmo de clasificación supervisada basado en el teorema de Bayes, el cual permite calcular la probabilidad de que una muestra pertenezca a una determinada clase en función de sus características. Su principal suposición es que todas las variables (o atributos) son condicionalmente independientes entre sí, dado el valor de la clase. A pesar de esta simplificación, Naive Bayes ha demostrado ser sorprendentemente eficaz, especialmente en tareas de clasificación de texto, detección de spam y análisis de sentimientos. En este proyecto, se utiliza para estimar la probabilidad de que un archivo sea malicioso o benigno, calculando las probabilidades de cada clase en función de características como el tipo de archivo, permisos, estructura binaria, cadenas de texto contenidas y otros atributos técnicos.
* **La extracción de características**: Es una fase crítica en cualquier sistema de clasificación, ya que define la calidad de los datos que se ingresarán al modelo. En este caso, se implementó un proceso de análisis estático de archivos utilizando herramientas nativas del sistema operativo Linux como file (para identificar el tipo de archivo), readelf (para examinar estructuras internas de archivos ELF), stat (para recuperar metadatos como permisos, tamaño y fechas de modificación) y strings (para extraer cadenas de texto legibles). A partir de estas herramientas, se recopilaron atributos como: si el archivo es ejecutable, si contiene referencias sospechosas, si posee estructuras típicas de malware, entre otros. Estas variables se transformaron en datos cuantificables que sirvieron como entrada para el clasificador Naive Bayes.
* **Clasificación binaria**: es una técnica de aprendizaje automático que busca categorizar una instancia en una de dos clases posibles. En este proyecto, las dos clases consideradas son: **"malicioso"** y **"benigno"**. A partir de las características extraídas de un archivo, el modelo calcula la probabilidad de que dicho archivo pertenezca a cada clase y lo asigna a la más probable. Esta forma de clasificación es directa, eficiente y apropiada para tareas donde el objetivo es detectar la presencia o ausencia de una amenaza, como es el caso en sistemas de detección de malware.
* Este proyecto establece una conexión directa con los contenidos del área de sistemas operativos, ya que involucra varios aspectos fundamentales. En primer lugar, requiere interacción directa con el sistema de archivos, accediendo a rutas del sistema y leyendo atributos de los archivos. También implica análisis de permisos, lo cual es vital para detectar si un archivo tiene privilegios inusuales o elevados. Además, se hace uso intensivo de comandos de consola y herramientas nativas para extraer información técnica, lo que refuerza el conocimiento del entorno UNIX/Linux. Finalmente, al estar desarrollado en C, se aplican principios de gestión de memoria eficiente, acceso a archivos a bajo nivel y optimización de recursos, todos ellos conceptos clave en la implementación de sistemas robustos y seguros.

# 4. Diseño e Implementación

## 4.1. Diseño de la Solución

Para este proyecto se optó por una implementación monolítica en lenguaje scripting bash y C, para lo cual se requerían tres módulos para las diferentes fases, estos módulos se generaron de manera independiente debido a que tienen un uso muy definido que puede ser reutilizado para diferentes aplicaciones, igualmente se optó por esta arquitectura debido los lenguajes de bajo nivel tienen una comunicación más directa con el S.O y también permiten un mejor control de las funcionalidades.

Se utilizó el algoritmo de NaiveBayes debido a que es muy utilizado en sistemas similares y nos brinda un gran rendimiento. Dentro de las consideraciones de diseño que se abordaron es como hacer para garantizar el acceso a las propiedades de los archivos ya que estos pueden ser de diferentes tipos, incluso una limitante para este proyecto es que no es muy factible trabajar con archivos maliciosos reales para evitar afectación del sistema de desarrollo.



## 4.2. Tecnologías y Herramientas

* Lenguaje de programación: C
* Sistema operativo: Ubuntu Linux
* Editor de código: Visual Studio Code
* Comandos utilizados: file, readelf, stat, strings, hexdump, grep
* Procesamiento de datos: estructuras propias en C (listas, arreglos)

## 4.3. Detalles de Implementación

La implementación fue llevada a cabo en una estructura de carpetas para separar la información, archivos sh para formar todas las estructuras de datos necesarias para el cada proceso y archivos en lenguaje C con utilización de las librerías mayormente utilizadas para la funcionalidad de transferencia de data y para el procesamiento lógico, matemático y de machine learning.

Para la primera fase se utilizaron funciones de bash como get\_exec\_size(), lo cual nos permite acceder a las características necesarias para el análisis.

En la segunda fase se crean estructuras en C para almacenar los datos y poder procesarlos, y los programas que generan el modelo de machine learning también son realizados de manera manual principalmente con la librería math.h.

La fase tres también se realiza con código en C, haciendo uso de los archivos generados en las fases anteriores, usando funciones de lectura de archivos y ejecución con parámetros de los bloques ya realizados.

# 5. Pruebas y Evaluación

## 5.1. Metodología de Pruebas

Para este proyecto se generaron archivos de prueba ya que es complejo utilizar código malicioso para las pruebas de tipo académico.

## 5.2. Casos de Prueba y Resultados

Presente los casos de prueba más importantes que ejecutó y los resultados obtenidos. Puede usar para ello una tabla como la siguiente:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID Caso de prueba** | **Descripción del caso de prueba** | **Resultado esperado** | **Resultado obtenido** | **Éxito/Fallo** |
| CP-001 | Archivo de texto de prueba | Archivo Benigno | Archivo Benigno | Éxito |
| CP-002 | [Descripción] | [Resultado esperado] | [Resultado observado] | [Éxito/Fallo] |
| CP-003 | [Descripción] | [Resultado esperado] | [Resultado observado] | [Éxito/Fallo] |
| … | … | … | … | … |

## 5.4. Problemas Encontrados y Soluciones

* El acceso al sistema de archivos a bajo nivel es algo complejo de manejar, para solventarlo se recurrió a scripts que tienen una mayor flexibilidad.
* Trabajar con archivos reales de código malicioso no pudo ser solventado ya que el equipo tiende a impedir su procesamiento, se solventó en parte por la simulación de archivos creados manualmente.

# 6. Conclusiones

* Es importante recalcar la viabilidad de integrar diferentes ramas del conocimiento para lograr una solución de este calibre, con la información y herramientas disponibles.
* En este tipo de arquitecturas se deben tener en consideración bastantes criterios que pueden garantizar o impedir el cumplimiento de los objetivos de la aplicación.
* Hay varios aspectos por mejorar en cuanto a la mantenibilidad y la escalabilidad.

## 7. Referencias

* Mitchell, T. M. (1997). Machine Learning. McGraw-Hill.

Referencia clásica sobre algoritmos de aprendizaje supervisado, incluyendo Naive Bayes.

* ChatGPT (2025). Asistente de desarrollo e inteligencia artificial. OpenAI. https://chat.openai.com