

## Formato Manual de sistema proyecto



### Manual de sistema del Proyecto:

Inteligencia Artificial para la detección de malware

Información general

Fecha: viernes 08 de diciembre del 2023

Nombre del estudiante: Daniel Bustamante Lagart

Matricula del estudiante: ACM30299

Materia: Técnicas generales de ataques 1

Docente: César Antonio Ríos Olivares

Fecha de entrega: lunes 11 de diciembre del 2023

Resumen del proyecto: Como autor de este proyecto, mi compromiso radica en explorar las fronteras de la ciberseguridad mediante el desarrollo de sistemas de Inteligencia Artificial especializados en el análisis de malware.

# Índice

1. Introducción.....	3
2. Arquitectura del sistema.....	3
2.1 Objetivos y requisitos de la herramienta .....	3
2.2 Tecnologías usadas en el proyecto .....	3
2.3 Pseudocódigo de la herramienta .....	3
2.4 Diagrama de funcionamiento de la herramienta .....	3
3. Diseño del sistema .....	4
3.1 Creación de algoritmos de redes bayesianas y bosques aleatorios .....	4
3.4 Integrar Framework de Flask para hacerlo web .....	11
4.Pruebas del sistema .....	19
4.1 Url de flask en el navegador Google chrome .....	19
4.2 Probar herramienta con Lagarcry.exe (D2L-Desktop.exe) .....	19
4.3 Probar herramienta con ZoomInstaller.exe (Instalador de zoom original).....	20
5. Bibliografía .....	21

## 1. Introducción

Este manual de sistema proporciona información detallada sobre cómo funciona y cómo crear la herramienta de Inteligencia Artificial para detectar malware en binarios (archivos exe).

## 2. Arquitectura del sistema

### 2.1 Objetivos y requisitos de la herramienta

- Capacidad de extraer encabezados PE del binario (archivo exe)
- Crear algoritmo de Inteligencia Artificial con bosques aleatorios y entrenarlo
- Crear algoritmo de Inteligencia Artificial con redes bayesianas y entrenarlo (Solo para comprobar investigación)
- Ser amigable para el usuario (usar flask para página web local)

### 2.2 Tecnologías usadas en el proyecto

- Lenguaje de programación Python
- Framework de desarrollo web Flask
- Visual Studio Code IDE de programación
- Anaconda entorno de desarrollo que tienes muchas librerías de Python preinstaladas
- Google chrome buscador de internet

### 2.3 Pseudocódigo de la herramienta

Inicio

Subir .exe a la web

Extraer encabezados PE del .exe

Evaluar PE con el modelo entrenado de IA

Si es malicioso regresar el resultado de “Es malware”

Sino regresar el resultado “No es malware”

Fin

### 2.4 Diagrama de funcionamiento de la herramienta



### 3. Diseño del sistema

La herramienta se diseñó primero creando los archivos .py para probar su funcionalidad, al final se creó el entorno flask para un mejor manejo del usuario, a continuación, describiremos cada paso.

#### 3.1 Creación de algoritmos de redes bayesianas y bosques aleatorios

Paso 1. Crear Código redes bayesianas en Visual Studio Code abierto con Anaconda.

```
# Importación de Librerías
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.mixture import BayesianGaussianMixture
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
import warnings
warnings.simplefilter('ignore')
# Leer el Archivo con los Datos del Malware
df = pd.read_csv('dataset_malwares.csv')
# Eliminar Columnas no Deseadas
dropped_df = df.drop(['Name', 'Machine', 'TimeStamp', 'Malware'], axis=1)
# Visualizar la Distribución de Clases
ax = sns.countplot(df['Malware'])
ax.set_xticks([0, 1])
ax.set_xticklabels(['Not Malware', 'Malware'])
# Preparar Conjuntos de Entrenamiento y Pruebas
X = dropped_df
y = df['Malware']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)
print("Number of used features:", X_train.shape[1])
# Inicializar el Clasificador de Redes Bayesianas
clf = BayesianGaussianMixture(
    n_components=2,      # Número de componentes (clases)
    max_iter=500,        # Número máximo de iteraciones
    random_state=0,       # Semilla para reproducibilidad
    weight_concentration_prior_type="dirichlet_process" # Tipo de prior para la
concentración de pesos
)
# Ajustar el clasificador a los datos de entrenamiento
clf.fit(X_train)
# Predecir en el conjunto de prueba
y_pred = clf.predict(X_test)
# Imprimir Informe de Clasificación
print(classification_report(y_test, y_pred, target_names=['Not Malware', 'Malware']))
```

```
# Generar y Visualizar Matriz de Confusión
matrix = confusion_matrix(y_test, y_pred)
ax = sns.heatmap(matrix, annot=True, fmt="d", cmap='Blues', cbar=False,
xticklabels=['Not Malware', 'Malware'], yticklabels=['Not Malware', 'Malware'])
ax.set_xlabel('Predicted Labels')
ax.set_ylabel('True Labels')
plt.show()
```

Paso 2. Guardar el archivo como Redes\_Bayesianas.py y correr el archivo.

Paso 3. Ver resultados.

PS C:\Users\danie\OneDrive\Documentos\Proyecto Análisis de  
ve/Documentos/Proyecto Análisis de malware/Redes bayesiana

Number of used features: 75

	precision	recall	f1-score	support
Not Malware	0.25	0.99	0.40	1003
Malware	0.00	0.00	0.00	2920
accuracy			0.25	3923
macro avg	0.13	0.50	0.20	3923
weighted avg	0.06	0.25	0.10	3923

*Imagen1. Muestra de la precisión del algoritmo redes bayesianas de inteligencia artificial, el cual es de bajo rendimiento.*

Paso 4. Crear Código bosques aleatorios en Visual Studio Code abierto con Anaconda.

```
# Importación de Librerías
import numpy as np
import pandas as pd
import pickle
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
import os
import warnings
warnings.simplefilter('ignore')
# Leer el Archivo con los Datos del Malware
df = pd.read_csv('dataset_malwares.csv')
# Eliminar Columnas no Deseadas
dropped_df = df.drop(['Name', 'Machine', 'TimeStamp', 'Malware'], axis=1)
# Visualizar la Distribución de Clases
ax = sns.countplot(df['Malware'])
ax.set_xticks([0, 1])
```

```

ax.set_xticklabels(['Not Malware', 'Malware'])
# Definir Características de Interés
features = ['MajorSubsystemVersion', 'MajorLinkerVersion', 'SizeOfCode',
'SizeOfImage', 'SizeOfHeaders', 'SizeOfInitializedData',
            'SizeOfUninitializedData', 'SizeOfStackReserve', 'SizeOfHeapReserve',
'NumberOfSymbols', 'SectionMaxChar']
i = 1
# Visualizar Distribuciones de Características
for feature in features:
    plt.figure(figsize=(10, 15))
    ax1 = plt.subplot(len(features), 2, i)
    sns.distplot(df[df['Malware']==1][feature], ax=ax1, kde_kws={'bw': 0.1})
    ax1.set_title(f'Malware', fontsize=10)
    ax2 = plt.subplot(len(features), 2, i+1)
    sns.distplot(df[df['Malware']==0][feature], ax=ax2, kde_kws={'bw': 0.1})
    ax2.set_title(f'Not Malware', fontsize=10)
    i = i + 2
# Preparar Conjuntos de Entrenamiento y Pruebas
X = dropped_df
y = df['Malware']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)
print("Number of used features:", X_train.shape[1])
# Inicializar el Clasificador de Bosque Aleatorio
clf = RandomForestClassifier(
    n_estimators=100, # Número de árboles
    random_state=0, # Semilla para reproducibilidad
    oob_score=True, # Calcular puntuación OOB
    max_depth=16 # Profundidad máxima de los árboles
)
# Ajustar el clasificador a los datos de entrenamiento
clf.fit(X_train, y_train)
# Predecir en el conjunto de prueba
y_pred = clf.predict(X_test)
# Imprimir Informe de Clasificación
print(classification_report(y_test, y_pred, target_names=['Not Malware', 'Malware']))
# Generar y Visualizar Matriz de Confusión
matrix = confusion_matrix(y_test, y_pred)
ax = sns.heatmap(matrix, annot=True, fmt="d", cmap='Blues', cbar=False,
xticklabels=['Not Malware', 'Malware'], yticklabels=['Not Malware', 'Malware'])
ax.set_xlabel('Predicted Labels')
ax.set_ylabel('True Labels')
# Obtener la Importancia de las Características
importance = clf.feature_importances_
importance_dict = dict(zip(dropped_df.columns.values, importance))

```

```
sorted_importance = dict(sorted(importance_dict.items(), key=lambda x: x[1],
reverse=True))
# Visualizar la Importancia de las Características
plt.figure(figsize=(10, 20))
sns.barplot(x=list(sorted_importance.values()), y=list(sorted_importance.keys()),
palette='mako')
plt.xlabel('Importance Value')
plt.ylabel('Feature Name')
plt.title('Feature Importance in Random Forest Classifier')
#Guardar modelo
# Ajustar el clasificador a los datos de entrenamiento
clf.fit(X_train, y_train)
plt.show()
```

Paso 5. Guardar el archivo como IA Bosques Aleatorios.py y correr el archivo.

Paso 6. Ver resultados.

```
PS C:\Users\danie\OneDrive\Documentos\Proyecto Análisis de malware\IA Bosques aleatorios\
ve\Documentos\Proyecto Análisis de malware\IA Bosques aleatorios> python IA_Bosques_Aleatorios.py
Number of used features: 75
      precision    recall  f1-score   support

Not Malware      0.99      0.96      0.98      1003
   Malware       0.99      1.00      0.99      2920

   accuracy              0.99      3923
  macro avg       0.99      0.98      0.98      3923
 weighted avg     0.99      0.99      0.99      3923
```

*Imagen2. Muestra de la precisión del algoritmo bosques aleatorios, podemos apreciar que tiene muy buen rendimiento*

Paso 7. El dataset de entrenamiento seleccionado debe ser el mismo para que tenga validez la comparación.

VirusShare_a878ba2600edaac5c98eff4432723b3	23117	144	3	0
VirusShare_ef9130570fddc174b312b2047f5f4cf0	23117	144	3	0
VirusShare_ef84cdeba22be72a69b198213dada81a	23117	144	3	0
VirusShare_6bf3608e60ebc16cbcff6ed5467d469e	23117	144	3	0

*Imagen3. Dataset de la página <https://www.kaggle.com/amauricio/pe-files-malwares>*

Paso 8. Concluimos que el mejor algoritmo es el de bosques aleatorios por su rendimiento por lo que vamos a usarlo, para que podamos usar el algoritmo debemos guardar el entrenamiento en un archivo que nos ayuda a comparar otros

.exe y nos da respuesta si es malware o no, debemos escribir esto al final del modelo elegido.

```
# Guardar modelo en un archivo pickle
with open('modelo_IA.pkl', 'wb') as model_file:
    pickle.dump(clf, model_file)
```

### 3.3 Extractor de encabezados PE para binarios (Archivos .exe)

Para poder comparar un .exe con nuestra Inteligencia artificial debemos extraer 75 características importantes del encabezado PE por lo que hemos hecho un código en Python que lo hace, importante que no falten datos porque no funciona si eso pasa.

Paso 1. Crear el Código y guardarlo como Análisis de PE.py

```
import pefile
import pickle
#Traer el archivo que tiene el entrenamiento
with open('modelo_IA.pkl', 'rb') as model_file:
    clf = pickle.load(model_file)
#Funcion de extracción de encabezados PE
def extract_pe_features(file_path):
    pe=pefile.PE(file_path)
    features = [
        int(pe.DOS_HEADER.e_magic),
        int(pe.DOS_HEADER.e_cblp),
        int(pe.DOS_HEADER.e_cp),
        int(pe.DOS_HEADER.e_crlc),
        int(pe.DOS_HEADER.e_cparhdr),
        int(pe.DOS_HEADER.e_minalloc),
        int(pe.DOS_HEADER.e_maxalloc),
        int(pe.DOS_HEADER.e_ss),
        int(pe.DOS_HEADER.e_sp),
        int(pe.DOS_HEADER.e_csum),
        int(pe.DOS_HEADER.e_ip),
        int(pe.DOS_HEADER.e_cs),
        int(pe.DOS_HEADER.e_lfarlc),
        int(pe.DOS_HEADER.e_ovno),
        int(pe.DOS_HEADER.e_oemid),
        int(pe.DOS_HEADER.e_oeminfo),
        int(pe.DOS_HEADER.e_lfanew),
        #int(pe.FILE_HEADER.Machine),
        int(pe.FILE_HEADER.NumberOfSections),
        #int(pe.FILE_HEADER.TimeDateStamp),
        int(pe.FILE_HEADER.PointerToSymbolTable),
        int(pe.FILE_HEADER.NumberOfSymbols),
        int(pe.FILE_HEADER.SizeOfOptionalHeader),
```



```
int(pe.FILE_HEADER.Characteristics),
int(pe.OPTIONAL_HEADER.Magic),
int(pe.OPTIONAL_HEADER.MajorLinkerVersion),
int(pe.OPTIONAL_HEADER.MinorLinkerVersion),
int(pe.OPTIONAL_HEADER.SizeOfCode),
int(pe.OPTIONAL_HEADER.SizeOfInitializedData),
int(pe.OPTIONAL_HEADER.SizeOfUninitializedData),
int(pe.OPTIONAL_HEADER.AddressOfEntryPoint),
int(pe.OPTIONAL_HEADER.BaseOfCode),
int(pe.OPTIONAL_HEADER.ImageBase),
int(pe.OPTIONAL_HEADER.SectionAlignment),
int(pe.OPTIONAL_HEADER.FileAlignment),
int(pe.OPTIONAL_HEADER.MajorOperatingSystemVersion),
int(pe.OPTIONAL_HEADER.MinorOperatingSystemVersion),
int(pe.OPTIONAL_HEADER.MajorImageVersion),
int(pe.OPTIONAL_HEADER.MinorImageVersion),
int(pe.OPTIONAL_HEADER.MajorSubsystemVersion),
int(pe.OPTIONAL_HEADER.MinorSubsystemVersion),
int(pe.OPTIONAL_HEADER.SizeOfHeaders),
int(pe.OPTIONAL_HEADER.CheckSum),
int(pe.OPTIONAL_HEADER.SizeOfImage),
int(pe.OPTIONAL_HEADER.Subsystem),
int(pe.OPTIONAL_HEADER.DllCharacteristics),
int(pe.OPTIONAL_HEADER.SizeOfStackReserve),
int(pe.OPTIONAL_HEADER.SizeOfStackCommit),
int(pe.OPTIONAL_HEADER.SizeOfHeapReserve),
int(pe.OPTIONAL_HEADER.SizeOfHeapCommit),
int(pe.OPTIONAL_HEADER.LoaderFlags),
int(pe.OPTIONAL_HEADER.NumberOfRvaAndSizes),
getattr(pe, 'SuspiciousImportFunctions', 0),
getattr(pe, 'SuspiciousNameSection', 0),
getattr(pe, 'SectionsLength', 0),
getattr(pe, 'SectionMinEntropy', 0),
int(pe.sections[0].get_entropy()),
getattr(pe, 'SectionMinRawsize', 0),
int(pe.sections[0].SizeOfRawData),
getattr(pe, 'SectionMinVirtualsize', 0),
int(pe.sections[0].Misc_VirtualSize),
getattr(pe, 'SectionMaxPhysical', 0),
getattr(pe, 'SectionMinPhysical', 0),
int(pe.sections[0].VirtualAddress),
getattr(pe, 'SectionMinVirtual', 0),
int(pe.sections[0].PointerToRawData),
getattr(pe, 'SectionMinPointerData', 0),
getattr(pe, 'SectionMaxChar', 0),
getattr(pe, 'SectionMainChar', 0),
```

```

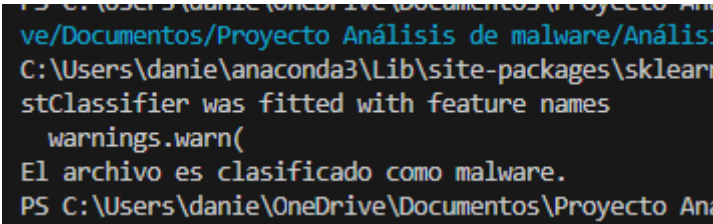
int(pe.OPTIONAL_HEADER.DATA_DIRECTORY[pefile.DIRECTORY_ENTRY['IMAGE_DIRECTORY_ENTRY_IMPORT']].VirtualAddress),

int(pe.OPTIONAL_HEADER.DATA_DIRECTORY[pefile.DIRECTORY_ENTRY['IMAGE_DIRECTORY_ENTRY_IMPORT']].Size),

int(pe.OPTIONAL_HEADER.DATA_DIRECTORY[pefile.DIRECTORY_ENTRY['IMAGE_DIRECTORY_ENTRY_EXPORT']].VirtualAddress),
    getattr(pe, 'ImageDirectoryEntryExport', 0),
    getattr(pe, 'ImageDirectoryEntryImport', 0),
    getattr(pe, 'ImageDirectoryEntryResource', 0),
    getattr(pe, 'ImageDirectoryEntryException', 0),
    getattr(pe, 'ImageDirectoryEntrySecurity', 0)
]
return features
#Ruta del archivo a analizar
ruta_archivo='D2L-Desktop.exe'
#Pasarle a la funcion la ruta del archivo
caracteristicas_exe = extract_pe_features(ruta_archivo)
#Variable que aloja el resultado del análisis
prediccion = clf.predict([caracteristicas_exe])
#Impresión del análisis
if prediccion[0] == 1:
    print("El archivo es clasificado como malware.")
else:
    print("El archivo no es clasificado como malware.")

```

Paso 2. Ejecutar y ver resultados del código, usaremos D2L-Desktop.exe(malware lagarcry.exe) y ZoomInstalles.exe (Software benigno).

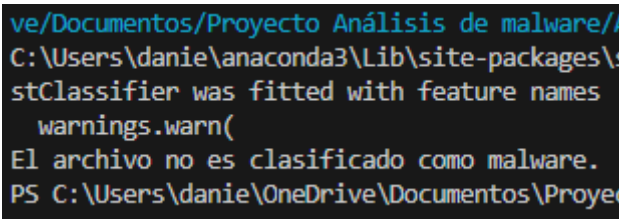


```

PS C:\Users\danie\OneDrive\Documentos\Proyecto Análisis de malware> python D2L-Desktop.exe
C:\Users\danie\anaconda3\Lib\site-packages\sklearn\metrics\classification.py:136: UserWarning:
stClassifier was fitted with feature names
warnings.warn(
El archivo es clasificado como malware.
PS C:\Users\danie\OneDrive\Documentos\Proyecto Análisis de malware>

```

Imagen4. Archivo analizado como malware (D2L-Desktop)=Lagarcry.exe



```

PS C:\Users\danie\OneDrive\Documentos\Proyecto Análisis de malware> python ZoomInstaller.exe
C:\Users\danie\anaconda3\Lib\site-packages\sklearn\metrics\classification.py:136: UserWarning:
stClassifier was fitted with feature names
warnings.warn(
El archivo no es clasificado como malware.
PS C:\Users\danie\OneDrive\Documentos\Proyecto Análisis de malware>

```

Imagen5. Archivo no clasificado como malware (ZoomInstaller.exe)=benigno.

### 3.4 Integrar Framework de Flask para hacerlo web

#### Paso 1. Instalar flask en nuestra carpeta

Pip install flask

Paso 2. Crea las carpetas templates, archivos\_subidos, IA y static, templates servira para alojar un html, IA para guardar tus IA, static para guardar las imágenes de tu html y archivos\_subidos será necesario para más adelante, también crea un archivo llamado app.py.

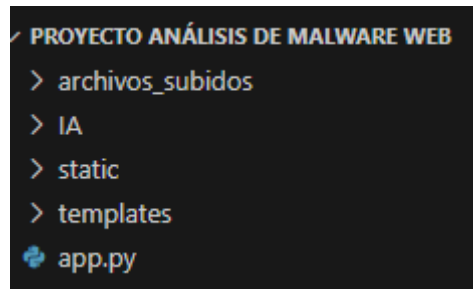


Imagen7. Estructura de archivos

Paso 3. En la carpeta IA guarda tus archivos de Inteligencia artificial creados y el más importante que es model\_IA.pkl.

Paso 4. En la carpeta templates guarda este html, es el que verá el usuario al usar la herramienta

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <title>Análisis estatico</title>
  <!-- Agrega el enlace al archivo CSS de Bootstrap -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfSpd3yD65VohhpucOmlASjC"
crossorigin="anonymous">
<style>
  /* Puedes agregar tu propio estilo CSS aquí */
  body {
    padding-top: 50px;
    background-color: #222831;
    color: #EEE;
    font-family: 'Arial', sans-serif;
  }
  h1{
    color: #00ADB5;
    font-size: 40px;
    font-style: normal;
    font-weight: 700;
```

```

    line-height: 96px; /* 100% */
  }
  p{
    font-size: 20px;
    font-style: normal;
    font-weight: 200;
    line-height: 20px;
  }
  button{
    border-radius: 24px;
    background: #00ADB5;
    backdrop-filter: blur(2px);
    font-size: 20px;
    font-weight: 300;
    margin-top: 30px;
    padding: 15px 50px
  }
  .inline{
    display:inline-block;
    text-align: center;
    vertical-align: middle;
  }
  form {
    max-width: 500px;
    margin: 0 auto;
    padding: 100px;
    background-color: #202020;
    border-radius: 8px;
    color: #00ADB5;
    text-align: center;
  }
  input[type="file"] {
    border-radius: 8px;
  }

</style>
</head>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
MrcW6ZMFYIzcLA8NI+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>
<!-- Barra de navegación -->
<nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
  <a class="navbar-brand" href="#" style="margin-left: 30px;" >Lagartini's IA análisis de
malware</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">

```

```

    <span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarNav">
  <ul class="navbar-nav ml-auto">
    <!-- Aquí poner menu -->
  </ul>
</div>
</nav>

<!-- Contenido principal -->
<div class="container mt-5" style="align-content: center;">
  <div class="inline">
    <h1>Bienvenidos a Lagartini's</h1>
    <p>Sube tu archivo .exe para analizar si es malware o no.</p>
    <!--<button>Sube tu archivo</button>-->
    <form action="/analizar" method="post" enctype="multipart/form-data">
      <input type="file" name="archivo" accept=".exe">
      <button type="submit">Sube tu archivo</button>
    </form>
    {% if resultado=='El archivo es clasificado como malware.' %}
    <div class="alert alert-danger d-flex align-items-center" role="alert">
      <svg class="bi flex-shrink-0 me-2" width="24" height="24" role="img" aria-label="Danger:"><use xlink:href="#exclamation-triangle-fill"/></svg>
      <div>
        <p>{{ resultado }}</p>
      </div>
    </div>
    {% endif %}
    {% if resultado=='El archivo no es clasificado como malware.' %}
    <div class="alert alert-success d-flex align-items-center" role="alert">
      <svg class="bi flex-shrink-0 me-2" width="24" height="24" role="img" aria-label="Success:"><use xlink:href="#check-circle-fill"/></svg>
      <div>
        <p>{{ resultado }}</p>
      </div>
    </div>
    {% endif %}
  </div>
  <div class="inline" style="padding-left: 200px;">
    <div id="carouselExampleInterval" class="carousel slide" data-bs-ride="carousel">
      <div class="carousel-inner" style="width: 300px;">
        <div class="carousel-item active" data-bs-interval="1000">
          
        </div>
        <div class="carousel-item" data-bs-interval="1000">
          
        </div>
      <div class="carousel-item">

```

```

        
    </div>
</div>
<button class="carousel-control-prev" type="button" data-bs-
target="#carouselExampleInterval" data-bs-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
</button>
<button class="carousel-control-next" type="button" data-bs-
target="#carouselExampleInterval" data-bs-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
</button>
</div>
</div>
<svg xmlns="http://www.w3.org/2000/svg" style="display: none;">
    <symbol id="check-circle-fill" fill="currentColor" viewBox="0 0 16 16">
        <path d="M16 8A8 8 0 1 1 0 8a8 8 0 0 1 16 0zm-3.97-3.03a.75.75 0 0 0-1.08.022L7.477
9.417 5.384 7.323a.75.75 0 0 0-1.06 1.06L6.97 11.03a.75.75 0 0 0 1.079-.02l3.992-
4.99a.75.75 0 0 0-.01-1.05z"/>
    </symbol>
    <symbol id="info-fill" fill="currentColor" viewBox="0 0 16 16">
        <path d="M8 16A8 8 0 1 0 8 0a8 8 0 0 0 0 16zm.93-9.412-1 4.705c-
.07.34-.029.533.304.533.194 0 .487-.07.686-.246l-.088.416c-.287.346-.92.598-1.465.598-
.703 0-1.002-.422-.808-1.319l.738-3.468c.064-.293.006-.399-.287-.47l-.451-.081.082-.381
2.29-.287zM8 5.5a1 1 0 1 1 0 2 1 1 0 0 1 0 2z"/>
    </symbol>
    <symbol id="exclamation-triangle-fill" fill="currentColor" viewBox="0 0 16 16">
        <path d="M8.982 1.566a1.13 1.13 0 0 0-1.96 0L1.165 13.233c-.457.778.091 1.767.98
1.767h13.713c.889 0 1.438-.99.98-1.767L8.982 1.566zM8 5.5c.535 0 .954.462.995l-.35
3.507a.552.552 0 0 1-1.1 0L7.1 5.995A.905.905 0 0 1 8 5zm.002 6a1 1 0 1 1 0 2 1 1 0 0 1
0 2z"/>
    </symbol>
</svg>

<!-- Agrega el enlace al archivo JavaScript de Bootstrap y al archivo jQuery -->
</body>
</html>

```

Paso 5. Usar en el archivo app.py este código en flask que ya tiene la función de la extracción de encabezados PE incrustado.

```

from flask import Flask, render_template, request
import pickle
import os

```

```
app = Flask(__name__)

# Cargar el modelo de IA al inicio de la aplicación
with open('IA/modelo_IA.pkl', 'rb') as model_file:
    clf = pickle.load(model_file)

app.config['STATIC_FOLDER'] = 'static'

def extract_pe_features(file_path):
    pe=pefile.PE(file_path)
    features = [
        int(pe.DOS_HEADER.e_magic),
        int(pe.DOS_HEADER.e_cblp),
        int(pe.DOS_HEADER.e_cp),
        int(pe.DOS_HEADER.e_crlc),
        int(pe.DOS_HEADER.e_cparhdr),
        int(pe.DOS_HEADER.e_minalloc),
        int(pe.DOS_HEADER.e_maxalloc),
        int(pe.DOS_HEADER.e_ss),
        int(pe.DOS_HEADER.e_sp),
        int(pe.DOS_HEADER.e_csum),
        int(pe.DOS_HEADER.e_ip),
        int(pe.DOS_HEADER.e_cs),
        int(pe.DOS_HEADER.e_lfarlc),
        int(pe.DOS_HEADER.e_ovno),
        int(pe.DOS_HEADER.e_oemid),
        int(pe.DOS_HEADER.e_oeminfo),
        int(pe.DOS_HEADER.e_lfanew),
        #int(pe.FILE_HEADER.Machine),
        int(pe.FILE_HEADER.NumberOfSections),
        #int(pe.FILE_HEADER.TimeDateStamp),
        int(pe.FILE_HEADER.PointerToSymbolTable),
        int(pe.FILE_HEADER.NumberOfSymbols),
        int(pe.FILE_HEADER.SizeOfOptionalHeader),
        int(pe.FILE_HEADER.Characteristics),
        int(pe.OPTIONAL_HEADER.Magic),
        int(pe.OPTIONAL_HEADER.MajorLinkerVersion),
        int(pe.OPTIONAL_HEADER.MinorLinkerVersion),
        int(pe.OPTIONAL_HEADER.SizeOfCode),
        int(pe.OPTIONAL_HEADER.SizeOfInitializedData),
        int(pe.OPTIONAL_HEADER.SizeOfUninitializedData),
        int(pe.OPTIONAL_HEADER.AddressOfEntryPoint),
        int(pe.OPTIONAL_HEADER.BaseOfCode),
        int(pe.OPTIONAL_HEADER.ImageBase),
        int(pe.OPTIONAL_HEADER.SectionAlignment),
```

```

        int(pe.OPTIONAL_HEADER.FileAlignment),
        int(pe.OPTIONAL_HEADER.MajorOperatingSystemVersion),
        int(pe.OPTIONAL_HEADER.MinorOperatingSystemVersion),
        int(pe.OPTIONAL_HEADER.MajorImageVersion),
        int(pe.OPTIONAL_HEADER.MinorImageVersion),
        int(pe.OPTIONAL_HEADER.MajorSubsystemVersion),
        int(pe.OPTIONAL_HEADER.MinorSubsystemVersion),
        int(pe.OPTIONAL_HEADER.SizeOfHeaders),
        int(pe.OPTIONAL_HEADER.CheckSum),
        int(pe.OPTIONAL_HEADER.SizeOfImage),
        int(pe.OPTIONAL_HEADER.Subsystem),
        int(pe.OPTIONAL_HEADER.DllCharacteristics),
        int(pe.OPTIONAL_HEADER.SizeOfStackReserve),
        int(pe.OPTIONAL_HEADER.SizeOfStackCommit),
        int(pe.OPTIONAL_HEADER.SizeOfHeapReserve),
        int(pe.OPTIONAL_HEADER.SizeOfHeapCommit),
        int(pe.OPTIONAL_HEADER.LoaderFlags),
        int(pe.OPTIONAL_HEADER.NumberOfRvaAndSizes),
        getattr(pe, 'SuspiciousImportFunctions', 0),
        getattr(pe, 'SuspiciousNameSection', 0),
        getattr(pe, 'SectionsLength', 0),
        getattr(pe, 'SectionMinEntropy', 0),
        int(pe.sections[0].get_entropy()),
        getattr(pe, 'SectionMinRawsize', 0),
        int(pe.sections[0].SizeOfRawData),
        getattr(pe, 'SectionMinVirtualsize', 0),
        int(pe.sections[0].Misc_VirtualSize),
        getattr(pe, 'SectionMaxPhysical', 0),
        getattr(pe, 'SectionMinPhysical', 0),
        int(pe.sections[0].VirtualAddress),
        getattr(pe, 'SectionMinVirtual', 0),
        int(pe.sections[0].PointerToRawData),
        getattr(pe, 'SectionMinPointerData', 0),
        getattr(pe, 'SectionMaxChar', 0),
        getattr(pe, 'SectionMainChar', 0),

int(pe.OPTIONAL_HEADER.DATA_DIRECTORY[pefile.DIRECTORY_ENTRY['IMAGE_DIRECTORY_ENTRY_IMPORT']].VirtualAddress),

int(pe.OPTIONAL_HEADER.DATA_DIRECTORY[pefile.DIRECTORY_ENTRY['IMAGE_DIRECTORY_ENTRY_IMPORT']].Size),

int(pe.OPTIONAL_HEADER.DATA_DIRECTORY[pefile.DIRECTORY_ENTRY['IMAGE_DIRECTORY_ENTRY_EXPORT']].VirtualAddress),
    getattr(pe, 'ImageDirectoryEntryExport', 0),
    getattr(pe, 'ImageDirectoryEntryImport', 0),
    getattr(pe, 'ImageDirectoryEntryResource', 0),
    getattr(pe, 'ImageDirectoryEntryException', 0),
    getattr(pe, 'ImageDirectoryEntrySecurity', 0)
]
return features

```



```
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/analizar', methods=['POST'])
def analizar():
    # Obtener el archivo del formulario
    archivo = request.files['archivo']

    # Guardar el archivo en el servidor (puedes personalizar la ruta)
    ruta_archivo = 'archivos_subidos/' + archivo.filename
    archivo.save(ruta_archivo)

    # Realizar el análisis con el modelo de IA
    características_exe = extract_pe_features(ruta_archivo)
    prediccion = clf.predict([características_exe])

    # Determinar el resultado del análisis
    resultado = "El archivo es clasificado como malware." if prediccion[0] == 1 else "El archivo
no es clasificado como malware."

    # Devolver el resultado a la página
    return render_template('index.html', resultado=resultado)
```

Paso 6. En la carpeta static se debe guardar las imágenes del carrusel del html, la estructura final de los archivos quedo así.

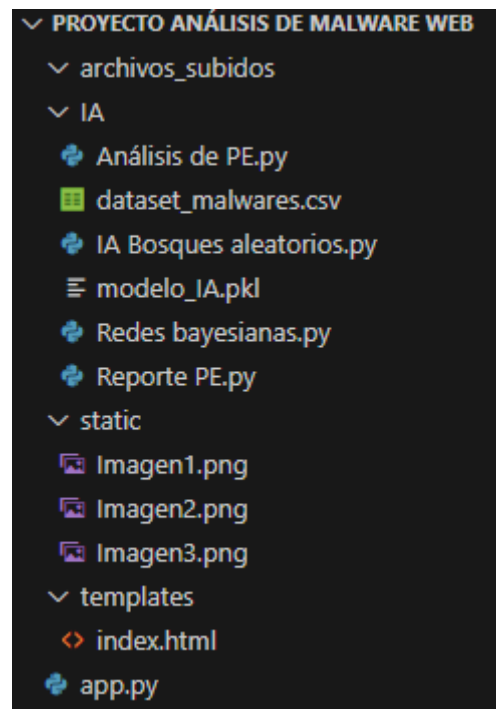


Imagen8. Estructura final de archivos

Paso 7. Ejecutar el archivo app.py e ir al url que nos da.

Python app.py

```
PS C:\Users\danie\OneDrive\Documentos\Proyecto Análisis de malware
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in production.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 163-829-166
```

Imagen9. Resultado de la ejecución app.py el link de la url

## 4.Pruebas del sistema

### 4.1 Url de flask en el navegador Google chrome

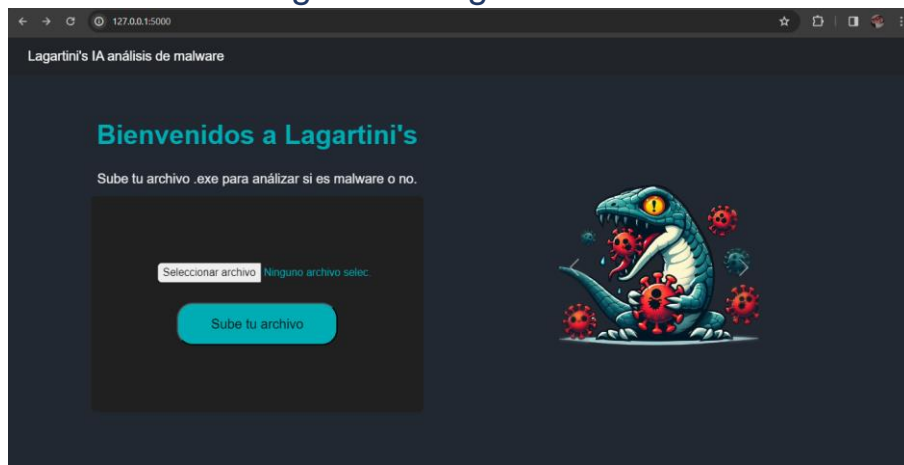


Imagen10. Inicio de la herramienta

### 4.2 Probar herramienta con Lagarcry.exe (D2L-Desktop.exe)

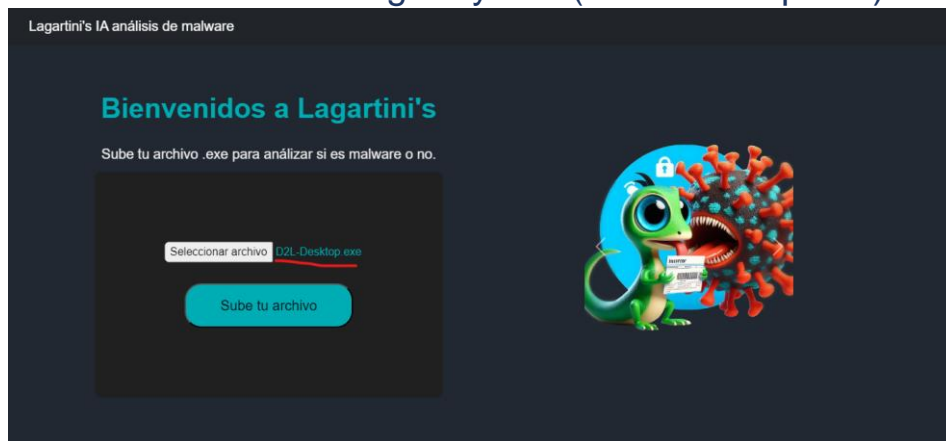


Imagen11. Subir archivo D2L-Desktop.exe

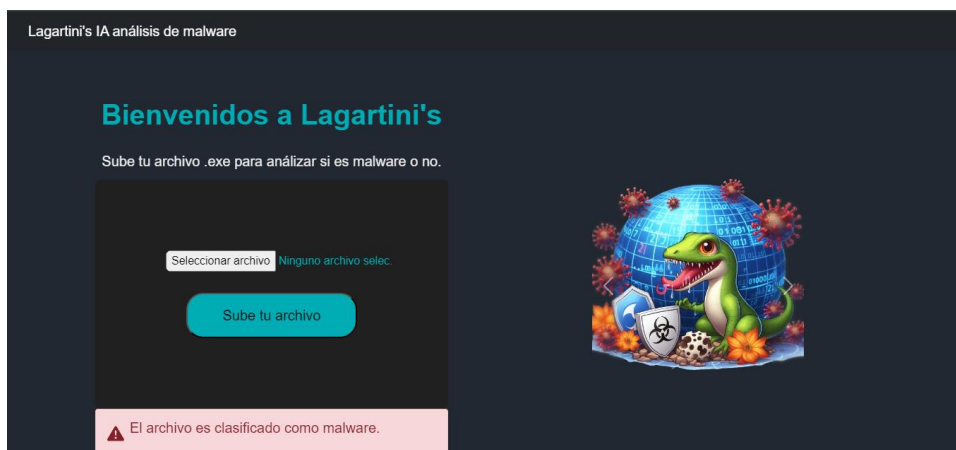
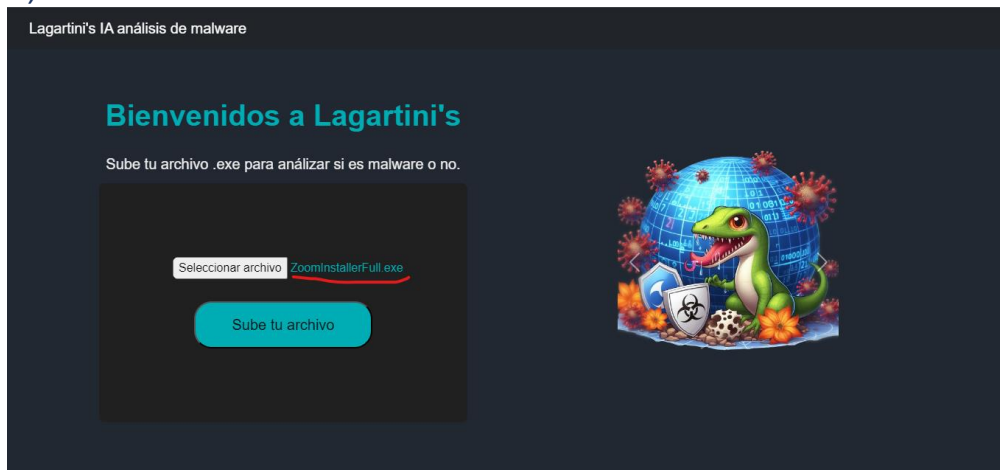
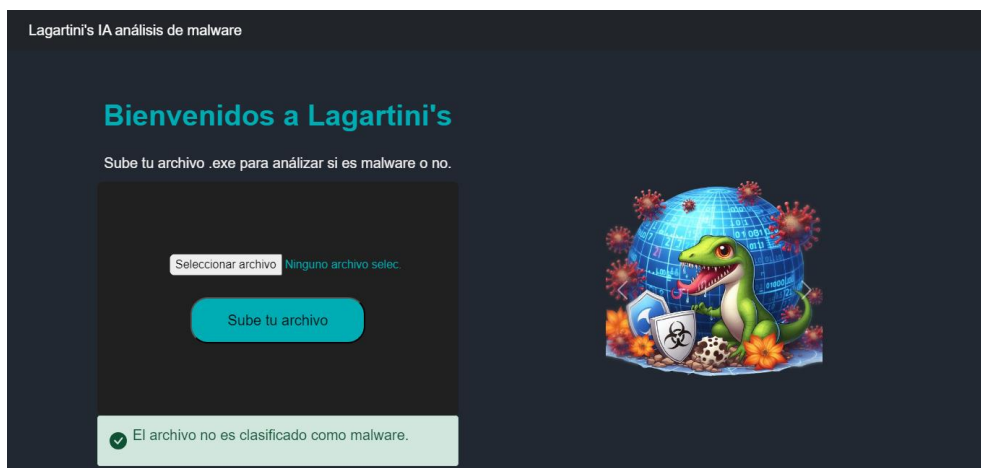


Imagen12. Resultado del análisis (Es malware)

### 4.3 Probar herramienta con ZoomInstaller.exe (Instalador de zoom original)



*Imagen13. Subir archivo ZoomInstallerFull.exe*



*Imagen14. Resultado del análisis (Benigno)*

## 5. Bibliografía

Staff, F. (2022, August 2). México registra 80,000 millones de intentos de ciberataques en 2022. Forbes México. <https://www.forbes.com.mx/mexico-registra-80000-millones-de-intentos-de-ciberataques-en-2022/>

Ehrhardt, M. (2023, January 18). Ciberataques e inflación, lo que más preocupa a las empresas. dw.com. <https://www.dw.com/es/ciberataques-e-inflaci%C3%B3n-lo-que-m%C3%A1s-preocupa-a-las-empresas/a-64430159>

Jiménez, M. M. (n.d.). Casos de ciberataques a empresas en 2022. <https://www.piranirisk.com/es/blog/ciberataques-empresas-en-2022>

7 formas en las que tus dispositivos se pueden infectar con malware. (2021, January 5). <https://www.welivesecurity.com/la-es/2021/01/05/formas-comunes-dispositivos-pueden-infectarse-con-malware/>

Software anti-malware frente a software antivirus: ¿Qué diferencia hay? (2023, April 19). [www.kaspersky.es](https://www.kaspersky.es). <https://www.kaspersky.es/resource-center/preemptive-safety/malware-remover-vs-antivirus-software>

Malwarebytes. (2020, June 3). ¿Es lo mismo antimalware y protección antivirus? Malwarebytes. <https://es.malwarebytes.com/antivirus/>

Sistemas EDR: qué son y cómo ayudan a proteger la seguridad de tu empresa | Empresas | INCIBE. (n.d.). <https://www.incibe.es/empresas/blog/sistemas-edr-son-y-ayudan-proteger-seguridad-tu-empresa>

¿Qué es XDR? (n.d.). Trend Micro. [https://www.trendmicro.com/es\\_mx/what-is/xdr.html](https://www.trendmicro.com/es_mx/what-is/xdr.html)

KeepCoding, R. (2022, August 23). Análisis estático de malwares | KeepCoding Bootcamps. KeepCoding Bootcamps. <https://keepcoding.io/blog/analisis-estatico-de-malwares/>

¿En qué consiste el análisis heurístico? (2023, August 18). [www.kaspersky.es](https://www.kaspersky.es/resource-center/definitions/heuristic-analysis).  
<https://www.kaspersky.es/resource-center/definitions/heuristic-analysis>

Chavez, J. J. S. (2023, October 3). ¿Qué es un malware? Tipos y cómo funcionan.  
<https://deltaprotect.com/blog/tipos-de-malware>

Técnicas avanzadas de ocultamiento de malware. (2017, March 7). [Slide show].  
PPT. <https://es.slideshare.net/dfpluc/tecnicas-avanzadas-de-ocultamiento-de-malware>

PE Explorer: EXE File Editor, DLL View Scan Tool for 32-bit Windows PE files.  
(n.d.). <http://www.pe-explorer.com/>

Karthikapadmanaban. (2023, February 15). Malware Detection using Random Forest. Kaggle. <https://www.kaggle.com/code/karthikapadmanaban/malware-detection-using-random-forest/input>

Samuel Mouro González (A Coruña, septiembre de 2022.). Técnicas de aprendizaje máquina para análisis de malware.  
[https://ruc.udc.es/dspace/bitstream/handle/2183/32112/MouroGonzalez\\_Samuel\\_TFG\\_2022.pdf?sequence=3](https://ruc.udc.es/dspace/bitstream/handle/2183/32112/MouroGonzalez_Samuel_TFG_2022.pdf?sequence=3)