



**UNIVERSIDAD DE CASTILLA-LA MANCHA**  
**ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA**

**GRADO EN INGENIERÍA INFORMÁTICA**

**TRABAJO FIN DE GRADO**

**TECNOLOGÍA ESPECÍFICA DE COMPUTACIÓN**

Detección y Clasificación de Malware usando  
técnicas inteligentes

Rubén Donate Serrano

Febrero de 2018





**UNIVERSIDAD DE CASTILLA-LA MANCHA**  
**ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA**

**Departamento de Sistemas Informáticos**

**TRABAJO FIN DE GRADO**

**TECNOLOGÍA ESPECÍFICA DE COMPUTACIÓN**

Detección y Clasificación de Malware usando  
técnicas inteligentes

Autor: Rubén Donate Serrano  
Directores: José Luis Martínez Martínez  
José Miguel Puerta Callejón

Febrero de 2018



## Declaración de Autoría

Yo, pepito perez con DNI..... , declaro que soy el único autor del trabajo fin de grado titulado ..... y que el citado trabajo no infringe las leyes en vigo sobre propiedad intelectual y que todo el material no original contenido en dicho trabajo está apropiadamente atribuido a sus legítimos autores.

Albacete, a.....

Fdo: pepito perez



# Resumen

Este documento pretende ser un manual sobre el uso del paquete de  $\text{\LaTeX} 2_{\epsilon}$  `tfg-esiiab.sty`. Este paquete de estilo da el formato al documento según las normas de estilo establecidas por la Escuela Superior de Ingeniería Informática de Albacete, perteneciente a la Universidad de Castilla-la Mancha.





*A mis compañeros y alumnos.*



# Agradecimientos

Agradecer a todas aquellas personas que me han ayudado a aprender  $\text{\LaTeX 2}_{\epsilon}$ , bien sea por lo que me enseñaron, o por las preguntas que me transmitieron y que yo intenté solucionar.



# Índice general

<b>ÍNDICE DE FIGURAS</b>	<b>XI</b>
Lista de Figuras . . . . .	XIII
<b>ÍNDICE DE TABLAS</b>	<b>XIII</b>
Lista de Tablas . . . . .	1
<b>1. INTRODUCCIÓN</b>	<b>1</b>
1.1. Motivación . . . . .	1
1.2. Objetivos . . . . .	1
1.3. Estructura de la memoria . . . . .	1
<b>2. TÉCNICAS UTILIZADAS</b>	<b>3</b>
<b>3. ANTECEDENTES Y ESTADO DE LA CUESTIÓN</b>	<b>5</b>
3.1. Solución 1 . . . . .	5
<b>4. METODOLOGÍA Y DESARROLLO</b>	<b>7</b>
<b>5. EXPERIMENTOS Y RESULTADOS</b>	<b>9</b>
<b>6. CONCLUSIONES Y PROPUESTAS</b>	<b>11</b>
6.1. Conclusiones . . . . .	11
6.2. Trabajo futuro . . . . .	11
<b>BIBLIOGRAFIA</b>	<b>13</b>



# ÍNDICE DE FIGURAS





# ÍNDICE DE TABLAS



# Capítulo 1

## INTRODUCCIÓN

1.1. Motivación

1.2. Objetivos

1.3. Estructura de la memoria



## Capítulo 2

# TÉCNICAS UTILIZADAS



## Capítulo 3

# ANTECEDENTES Y ESTADO DE LA CUESTIÓN

Para el proyecto de Kaggle que se ha seleccionado para este Trabajo de Final de Grado, se he realizado una búsqueda para intentar encontrar soluciones para éste. De esta, se ha conseguido encontrar varias propuestas que se habían presentado para este mismo reto.

A continuación, se explicará el proceso que llevaron a cabo para desarrollar estas soluciones.

### 3.1. Solución 1

Esta primera solución fue realizada por Vishnu Chevli [\[1\]](#).

Además, es la mas sencilla de las soluciones que se van a comentar.

Esta solución se ha desarrollado para poder ser ejecutada de manera indistinta en Python 2 y Python 3 y consiste en primer lugar en descomprimir las dos bases de datos proporcionadas. Después, de los ficheros descomprimidos los que poseen la extensión .asm son descartados y los que poseen la extensión .bytes son comprimidos con gzip. Esto se realiza para reducir el espacio que ocupa las bases de datos en el disco duro, ya que el espacio de estas sin comprimir es de aproximadamente 1 Tb.

A continuación, se recorre y se accede a todos estos ficheros comprimidos para analizar en cada uno de ellos el fichero binario en formato hexadecimal que contiene. Este análisis consiste en contar las apariciones en éste de dos elementos en hexadecimal

para terminar guardándolos en un fichero csv dentro de un archivo comprimido. Este análisis se realiza de manera paralela para las dos bases de datos terminando así la extracción de características de estas.

El siguiente paso, consiste en construir el modelo y obtener la predicción para el mismo. Para ello se comienza leyendo el fichero proporcionado con las etiquetas para cada uno de los ficheros que forman la base de datos de train. A continuación, se recupera las características que se han extraído en el paso anterior y se crea un array de numpy de ceros cuyo índice es una tupla formada por el id del fichero en primer lugar y como segundo valor el número de posibles características mas una posición adicional para almacenar la etiqueta obtenida anteriormente.



## Capítulo 4

# METODOLOGÍA Y DESARROLLO



## Capítulo 5

# EXPERIMENTOS Y RESULTADOS



## Capítulo 6

# CONCLUSIONES Y PROPUESTAS

6.1. Conclusiones

6.2. Trabajo futuro



# Bibliografía

- [1] V. Chevli. (2015) Beating the benchmark for Microsoft Malware Classification Challenge (BIG 2015). [Online]. Available: <https://github.com/vrajs5/Microsoft-Malware-Classification-Challenge> 5





# **CONTENIDO DEL CD**