UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

TECNOLOGÍA ESPECIFICA 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 ESPECIFICA 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 LaTeX 2_ε `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 LaTeX 2_ε, bien sea por lo que me enseñaron, o por las preguntas que me transmitieron y que yo intenté solucionar.

Índice general

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

1. ngram
2. randomForrestClassifier
3. numpy

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 presentadas para este reto. De esta búsqueda, se ha conseguido encontrar varias soluciones presentadas para este mismo reto. A continuación, se explicaran las soluciones presentadas que se han encontrado para este reto de Kaggle.

3.1. Solución 1

Esta primera solución que se va a comentar, fue realizada por Vishnu Chevli [1] y en la clasificación del Reto de Kaggle obtuvo una resultado en el ranking publico de 0.023121984 en la posición 90 y en el ranking privado 0.018856579 en la posición 72.

Esta solución se desarrollo para que pudiera ser ejecutada de manera indistinta en Python 2 y Python 3, es muy sencilla y consiste en:

En primero lugar, se descomprimir las dos bases de datos que se proporcionan. Estas bases de datos contienen dos tipos de ficheros que contienen la información obtenida de IDA al desensamblar la muestras en dos formatos, siendo estos formatos ASM para los ficheros con extensión .asm y binario en formato hexadecimal para los ficheros con extensión .bytes. Para esta solución solo se utilizaran los ficheros con extensión .bytes. Después, estos ficheros son nuevamente comprimidos en formato gzip de manera separada y separando las bases de datos en dos carpetas distintas. El motivo de realizar esto es para ahorrar espacio en disco, ya que el espacio de la base de datos completas con todos los tipos de ficheros sin comprimir es de aproximadamente 1 Tb.

El siguiente paso, consiste en extraer las características de los ficheros que se han generado en el paso anterior. Esta extracción consiste en generar un array donde cada posición corresponde a cada uno de los posibles uno ngrama 1, formado por dos caracteres en formato hexadecimal, incluyendo el ngram '??' que indica que el valor correspondiente no esta mapeado, y contar las veces que aparece cada uno de ellos en el fichero, para concluir, guardando los resultados en un fichero csv de manera separada para cada una de las bases de datos. Este proceso ser realizara de manera conjunta para las dos bases de datos y de manera paralela para 2 ficheros.

El siguiente paso, es construir el modelo que para este caso es un RandomForrestClassifier 2. Para ello, lo primero se tiene que realizar es obtener la clase para cada uno de los fichero, para lo cual se tiene que abrir y recorrer el fichero 'trainLabels.csv' que se proporciona y almacenar la información del mismo en un diccionario que posteriormente se utilizará. A continuar, se crea una matriz con numpy 3 de tamaño el numero de ficheros de la base de datos de train por el número de uno ngramas mas uno adicional para la clase, en este caso $10868 * 258$. Estos datos son obtenidos del fichero csv correspondiente a la base de datos de train para los valores de los uno ngramas y del diccionario creado anteriormente para la clase. Después, se crea el modelo con los valores por defecto, excepto la semilla que utiliza 123, el número de hilos que utiliza 5 y el nivel de información que muestra que utiliza la mas detallada 2. Por ultimo, solo queda entrenar el modelo pasando le la matriz de numpy separando las características y la clase, o lo que es lo mismo la matriz sin la ultima columna y la ultima columna por separado, siendo esta ultima columna la clase.

Para terminar, solo nos queda realizar la previsión para la base de datos de test. Para ello, ahí que repetir el proceso de recuperar las características creando una matriz con numpy 3 pero en este caso sin añadir un columna para la clase, también se crea un array donde se almacena el nombre del fichero en el mismo indice que la fila de la matriz, esto es así porque la predicción ha de ser identificada con es valor. Lo siguiente, es realizar la predicción para la base de datos de test. Para concluir, escribiendo los resultados en un fichero comprimido en el que guarda la información con el formato que se nos proporciona en el ejemplo, siendo este el id del fichero seguido de la probabilidad que sea cada una de las clases separado todo por comas.

3.2. Solución 2

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