

UNIVERSIDAD DEL VALLE DE GUATEMALA

CC3094 - Security Data Science

Sección 10

Ing. Jorge Yass



Análisis de malware en archivos PE

Raúl Jiménez - 19017
Bryann Alfaro - 19372
Donaldo Garcia - 19682
Oscar Saravia - 19322
Diego Arredondo - 19422

GUATEMALA, 21 de Febrero de 2023

Índice:

Motivación :	2
Preguntas clave:	2
Revisión de la literatura:	2-4
Recolección de datos:	4
Limpieza de datos:	4-8
Bibliografía:	8

Motivación:

El objetivo principal del proyecto de análisis de malware en archivos PE implementando data science es utilizar técnicas y herramientas de ciencia de datos para mejorar la detección, el análisis y la comprensión del comportamiento de los programas maliciosos en archivos Portable Executable (PE), utilizados en sistemas operativos Windows.

El proyecto incluye la aplicación de técnicas de análisis estadístico y de aprendizaje automático para identificar patrones y características comunes en el código malicioso, y la extracción de información importante del archivo PE. También se utilizan técnicas de visualización de datos para representar los resultados del análisis y facilitar la comprensión de los patrones y tendencias.

Asimismo, se busca mejorar la eficacia del análisis de malware en archivos PE y proporcionar a los usuarios y a los expertos en seguridad informática una mejor comprensión de cómo funciona el malware, así como identificar nuevas amenazas y patrones de ataque. Además, el proyecto puede contribuir a la mejora de la seguridad informática en general al proporcionar una mejor comprensión de cómo prevenir y mitigar los ataques maliciosos en archivos PE.

Preguntas clave:

- ¿Cuáles son las características más relevantes al momento de clasificar un archivo PE como malware?
- ¿Son los requerimientos mínimos solicitados por el PE útiles para determinar si es un malware?
- ¿El tamaño del código puede ser significativo al momento de detectar un archivo como malware?

Revisión de la literatura:

La propuesta de **“Análisis comparativo de algoritmos de machine learning para detección de malware en aplicaciones de android”** por Victor Montenegro nos muestra cómo es que en una muestra de 15,945, siendo un 20% de las muestras benignas y 80% de las muestras malignas, se utilizaron varios algoritmos de machine learning para detectar el malware en aplicaciones. Para realizar la investigación se recolectaron datos de muestra de aplicaciones de android, se hizo una extracción de características, preprocesamiento de datos y luego procedieron a utilizar algoritmos de aprendizaje automático. Entre ellos los más destacados fueron Support Vector Machine, K-nearest Neighbor, Naive Bayes Decision tree, Random forest y Logic Regression. En los resultados que nos muestra Montenegro podemos observar que los que tienen mayor rendimiento en precisión y F1 son el random forest y el support vector Machine. (Montenegro, 2022)

De igual forma en el artículo Montenegro emplea la técnica de combinar varios algoritmos mediante un modelo de votación y esto mejora significativamente la detección de malware. Este artículo nos sirve para explorar la posibilidad de combinar varios algoritmos de machine learning específicamente los que muestran que en el estudio pasado tuvieron una mejora significativa. También nos guía en la forma en la que se podría hacer la limpieza de datos, para la extracción de cadenas de texto, características del archivo y la normalización de datos para tenerlo todo a una misma escala.(Montenegro, 2022)

Xuan y Duong proponen en su estudio **“Detecting Malware based on Analyzing Abnormal Behaviors of PE Files”** una forma de poder detectar el malware en archivos PE y encontrar ciertas características que presentan comportamientos anormales. (Xuan y Duong, 2021)

Esto se realizó con el fin de proponer una nueva forma de clasificación y no clasificar el malware tomando en cuenta solamente información que se recolectó en algún ambiente virtualizado, sino que comprender y analizar cada parte del archivo PE para poder construir distintos perfiles de comportamiento de malware. (Xuan y Duong, 2021)

Para el entrenamiento y prueba de los datos, se implementaron varios algoritmos. En primer lugar se utilizó random forest y supportive vector machine. En el enfoque de deep learning se utilizó Multi Layers Perceptron, Convolutional Neural Network y Long Short Term Memory. Para trabajar con los mismos se utilizaron 49,128 observaciones de las cuales 24,528 son malwares y 24,602 son archivos normales. (Xuan y Duong, 2021)

Como resultado general se observó que el algoritmo de CNN (Convolutional Neural Network) tuvo los mejores resultados, probablemente por el gran número de features a analizar (485 features). La manera en que este artículo apoya la investigación se basa en que se demostró que al utilizar las características que se obtienen de analizar los componentes de un PE file pueden ser de ayuda para clasificar el mismo como malicioso. (Xuan y Duong, 2021)

En el artículo **“IMDS: intelligent malware detection system”**, desarrollado por Yanfang Ye, Dingding Wang, Tao Li, and Dongyi Ye., se muestra el desarrollo de un sistema inteligente de detección de malware, IMDS por sus siglas en inglés, en donde utilizaron una clasificación basada en la asociación orientada a objetos. El sistema IMDS consta de tres módulos principales, el primero de ellos es el analizador PE, el segundo está conformado por un generador de reglas OOA y por último un clasificador basado en reglas. (Ye, et.al, 2007)

Durante las pruebas del sistema, se evaluó el sistema en una gran colección de ejecutables que incluyen 12214 muestras benignas y 17366 maliciosas, además de brindar un estudio experimental sobre varios softwares de antivirus, así como técnicas de extracción de datos para la detección de malware utilizando una recopilación de datos. (Ye, et.al, 2007)

El artículo **"Malware Detection using Machine Learning and Data Science Techniques: A Survey"** es una revisión exhaustiva de la literatura existente sobre el uso de técnicas de aprendizaje automático y ciencia de datos para la detección de malware en archivos PE.

Los autores describen la importancia de la detección de malware y la necesidad de utilizar técnicas avanzadas para abordar los desafíos que plantea.

Los autores también describen técnicas de reducción de dimensionalidad, como el análisis de componentes principales (PCA) y la reducción de características basada en árboles (RFEV), que se utilizan para reducir la complejidad de los datos y mejorar el rendimiento del modelo.

Por último, los autores destacan las limitaciones y desafíos de estas técnicas, como la falta de datos de entrenamiento de calidad y la capacidad de los atacantes para evadir las técnicas de detección de malware. También se sugieren áreas de investigación futura, como el uso de técnicas de aprendizaje profundo y la exploración de nuevas fuentes de datos, como los registros de eventos del sistema y los datos de red. (Oualmakran, Y., El Marraki, M., Machkour, M., Ouassou, M., & Azizi, A, 2020)

En el artículo **“Malware Images: Visualization and Automatic Classification”**, los autores presentan un método efectivo para poder visualizar, y a la vez, clasificar malware con técnicas de procesamiento de imágenes. Estos autores recalcan el aspecto de “una manera efectiva de visualizar malware”, ya que esto presenta un avance para este desafío.

Los autores proponen una técnica basada en escalas grises y familias de malware, ya que en su observación de los resultados, indican que los malwares presentan una similitud en cuanto al diseño y textura. Gracias a estas similitudes se pueden utilizar métodos de clasificación utilizando características de imagen estandarizadas. (Nataraj et al. 2011)

Con esta metodología, se logra detectar en un 98% los casos de malware, en un total de 9458 imágenes y hasta 25 familias de malwares. Adicionalmente, también se presenta una buena resistencia ante técnicas populares de ofuscación, como lo es el cifrado de secciones. (Nataraj et al. 2011)

Recolección de datos:

La recolección de datos es un aspecto fundamental en cualquier proyecto de detección de malware en archivos PE (ejecutables de Windows). En este caso, los datos fueron recopilados de la página [Kaggle](#), que cuenta con una base de datos de 19,612 archivos PE para entrenamiento y 18 archivos de prueba.

Kaggle es una plataforma en línea que se utiliza para compartir conjuntos de datos y modelos de aprendizaje automático. En el caso de la detección de malware, Kaggle se ha convertido en una fuente valiosa de datos para los investigadores, estudiantes y profesionales en el área de tecnología. Los datos recopilados en Kaggle provienen de una variedad de fuentes, incluyendo repositorios de malware públicos, informes de investigadores de seguridad y empresas de antivirus.

La base de datos de Kaggle incluye una amplia variedad de tipos de malware, desde troyanos hasta virus y gusanos. Cada archivo PE se etiqueta con una etiqueta que indica si

el archivo es un archivo legítimo o un archivo malicioso. Estas etiquetas son cruciales para el entrenamiento y evaluación de los modelos de detección de malware.

Es importante realizar una limpieza y validación de los datos para garantizar que sean confiables y útiles para el análisis.

El dataset se encuentra bajo la licencia CC0: Public Domain, esto hace referencia a que cualquier obra realizada por científicos, educadores, artistas y otros creadores y propietarios de contenido protegido por derechos de autor o bases de datos ahora puede utilizarse de manera pública para cualquier propósito, incluyendo la mejora o reutilización de las obras. (CreativeCommons, s.f)

Limpieza de datos:

Como proceso inicial se decidió mostrar las observaciones para verificar que los datos están siendo leídos de la manera correcta

train.head()

0.0s

Python

	Name	e_magic	e_cblp	e_cp	e_crc	e_cparhdr	e_minalloc	e_maxalloc	e_ss	e_sp	...	SectionMaxChar	SectionMainChar	DirectoryEntryImport	DirectoryEntryIn
0	VirusShare_a878ba26000edaac5c98eff4432723b3	23117	144	3	0	4	0	65535	0	184	...	3758096608	0	7	
1	VirusShare_ef9130570fddc174b312b2047f5f4cf0	23117	144	3	0	4	0	65535	0	184	...	3791650880	0	16	
2	VirusShare_ef84cdeba22be72a69b198213dada81a	23117	144	3	0	4	0	65535	0	184	...	3221225536	0	6	
3	VirusShare_6b13608e60ebc16cbcf6ed5467d469e	23117	144	3	0	4	0	65535	0	184	...	3224371328	0	8	
4	VirusShare_2cc94d952b2efb13c7d6bbe0dd59d3fb	23117	144	3	0	4	0	65535	0	184	...	3227516992	0	2	

5 rows × 19 columns

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

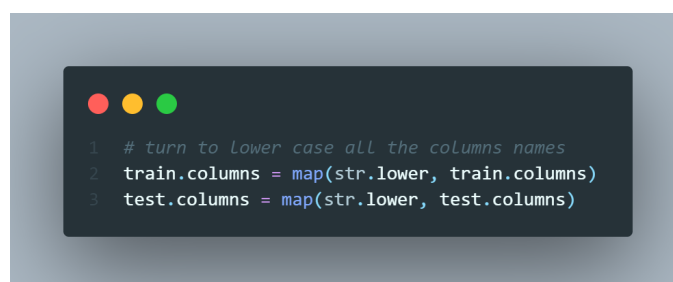
77

78

79

imagen #1: Muestra de los primeros 5 elementos del dataset de entrenamiento

Al observar las columnas, se observó la presencia de variables que empezaban con minúsculas y otras con mayúsculas, por lo que se decidió estandarizar esto de igual forma



```
1 # turn to lower case all the columns names
2 train.columns = map(str.lower, train.columns)
3 test.columns = map(str.lower, test.columns)
```

imagen #2: Estandarización de nombre de variables

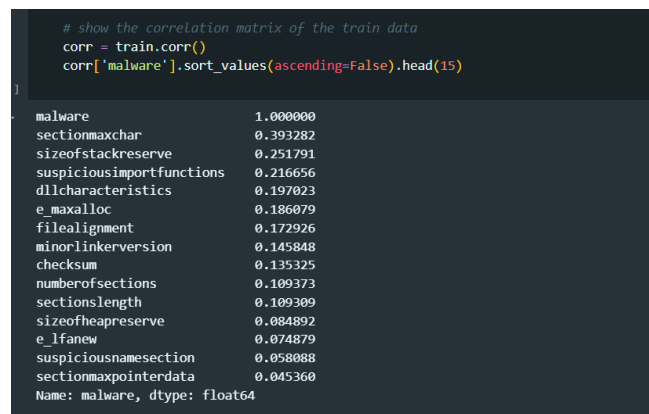
Seguidamente, se removieron los valores NA en el caso de que existieran para evitar tener datos que no son significativos dentro de nuestro dataset.



```
1 train.dropna(inplace=True)
2 test.dropna(inplace=True)
3 train.shape
```

imagen #3: Eliminación de valores NA en los datasets

Para poder escoger las variables a utilizar, se optó por realizar un análisis de correlaciones tomando en cuenta la columna target de *malware* y tomando las primeras 15 correlaciones con el valor más alto.



```
# show the correlation matrix of the train data
corr = train.corr()
corr['malware'].sort_values(ascending=False).head(15)
```

malware	1.000000
sectionmaxchar	0.393282
sizeofstackreserve	0.251791
suspiciousimportfunctions	0.216656
dllcharacteristics	0.197023
e_maxalloc	0.186079
filealignment	0.172926
minorlinkerversion	0.145848
checksum	0.135325
numberofsections	0.109373
sectionslength	0.109309
sizeofheapreserve	0.084892
e_lfanew	0.074879
suspiciousnamesection	0.058088
sectionmaxpointerdata	0.045360

Name: malware, dtype: float64

imagen #4: Correlaciones más altas con variable malware

Con esta información ya procesada, se procedió a realizar un mapa de calor para poder observar las correlaciones de las nuevas variables del dataset a utilizar.

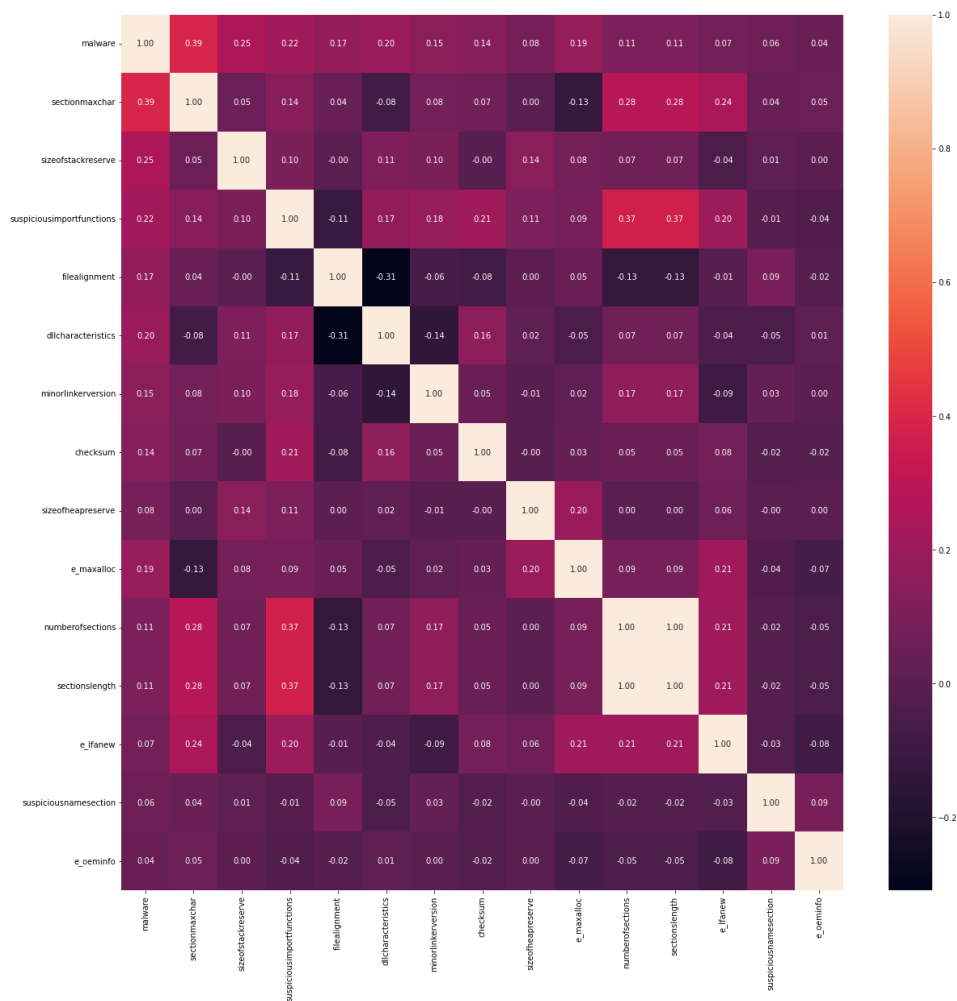


imagen #5: Mapa de calor de correlaciones

Luego de esto se utilizaron diagramas de caja de bigotes para determinar si había datos atípicos en las variables que tienen mayor correlación con la columna de malware. Observando la primer variable “sectionmaxchar” logramos observar que no se puede hacer una limpieza de datos atípicos. Con la segunda variable que tiene mayor correlación (sizeofstackreserve) se puede observar que existen algunos valores atípicos (mayores a 1) por lo cual se decide eliminar esas observaciones.

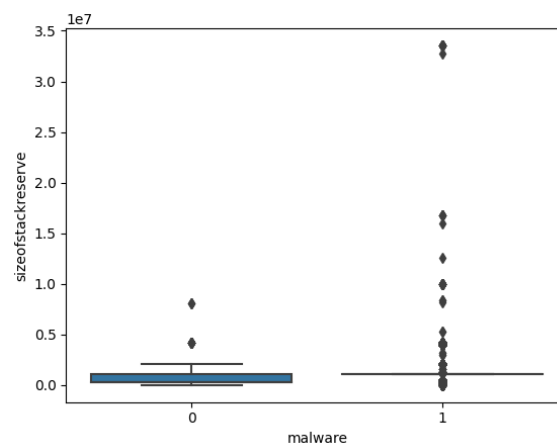


imagen #6: diagrama de caja de bigotes con variable sizofstackreserve

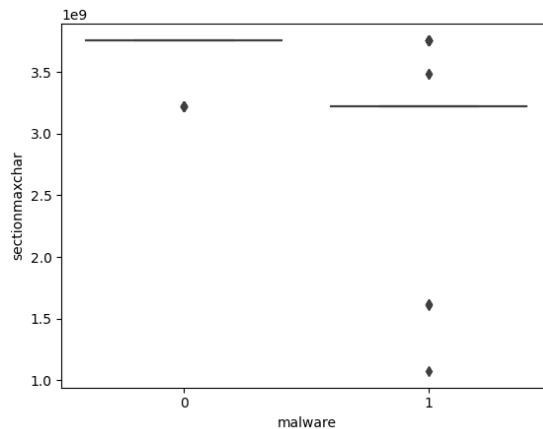


imagen #7: diagrama de caja de bigotes con variable sectionmaxchar

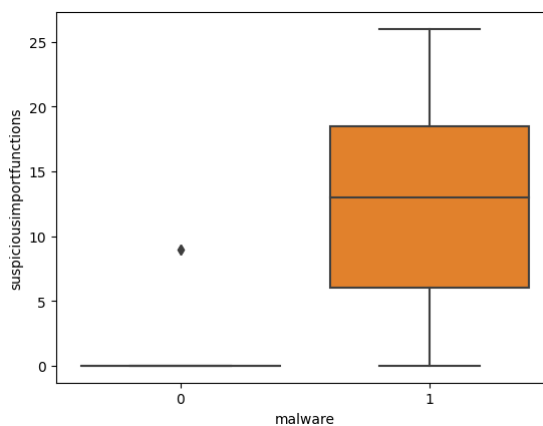


imagen #8: diagrama de caja de bigotes con variable suspiciousimportfunctions

Para finalizar la limpieza de datos se decidió analizar si el dataset cuenta con una medida de datos balanceada. Observando los valores de datos que tienen '0' y '1' en la columna 'Malware' logramos concluir que se necesitaba balancear los datos. Por lo tanto se decide utilizar 5,012 datos de cada observación que cuente con '0' y de igual manera con '1' en la variable de malware. Teniendo así un total de 10,024 observaciones.

```
# Remove some of the rows that has value "1" in the malware column to balance the data
train = train[train['malware'] == 0].sample(5012).append(train[train['malware'] == 1].sample(5012))
train['malware'].value_counts()

C:\Users\raula\AppData\Local\Temp\ipykernel_20128\967369620.py:2: FutureWarning: The frame.append method
as in a future version. Use pandas.concat instead.
  train = train[train['malware'] == 0].sample(5012).append(train[train['malware'] == 1].sample(5012))

0    5012
1    5012
Name: malware, dtype: int64
```

Literatura Citada:

- Montenegro, V. (2022) Análisis comparativo de algoritmos de machine learning para detección de malware en aplicaciones android. Extraído de: <https://repositorio.uss.edu.pe/handle/20.500.12802/10059>

- Montenegro, V. (2022) Analisis comparativo de algoritmos de machine learning para detección de malware en aplicaciones android. Extraído de: <https://repositorio.uss.edu.pe/bitstream/handle/20.500.12802/10059/Montenegro%20Guerrero%20Victor%20Agustin.pdf?sequence=1&isAllowed=y>
- Xuan, C, Duong, L. (2021). Detecting Malware based on Analyzing Abnormal behaviors of PE File. Extraído de: https://thesai.org/Downloads/Volume12No3/Paper_55-Detecting_Malware_based_on_Analyzing_Abnormal.pdf
- CreativeCommons. (s.f). CC0. Extraído de: <https://creativecommons.org/share-your-work/public-domain/cc0/>
- Ye, Y., Wang, D., Li, T., & Ye, D. (2007). *IMDS. Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '07*. doi:10.1145/1281192.1281308
- Oualmakran, Y., El Marraki, M., Machkour, M., Ouassou, M., & Azizi, A. (2020). Malware Detection using Machine Learning and Data Science Techniques: A Survey. *Journal of Information Security and Applications*, 55, 102622. doi: 10.1016/j.jisa.2020.102622
- Nataraj, Lakshmanan & Karthikeyan, Shanmugavadivel & Jacob, Grégoire & Manjunath, B.. (2011). Malware Images: Visualization and Automatic Classification. 10.1145/2016904.2016908.