Machine learning: Análisis y evasión de malwares bajo un enfoque Heuristico.

Mauricio Jara, Instituto Tecnológico Cibertec , Noviembre del 2017
cln.mauricio@gmail.com

Abstract:

Currently machine learning is being used in different sectors and computer security is not immune to this. While security software improves its analytical approaches by applying machine learning (ML), attacks that violate these automatic analyzes are reproduced exponentially. The constant growth of new and more complex malwares makes it impossible to detect all these potential attacks together. The methods used for a better extraction of characteristics are exposed and thus achieve greater precision. This article presents what has been the progress in the analysis of malwares using ML and how cybercriminals can go unnoticed by the antivirus that apply these new techniques.

keywords *machine learning, análisis de malwares, evasión.*

I. Introducción

El surgimiento de nuevas herramientas y tecnologías hacen posible que las empresas obtengan una mejor seguridad de su información. Sin embargo del mismo modo el incremento de ataques de malwares es exponencial por el mismo hecho. Así pues se llega a un punto en el que es casi imposible abarcar las distintas opciones de un cibercriminal, con ataques cada vez más complejos y herramientas mucho más eficaces. Ante este problema muchas empresas ya han empezado a mirar hacia el **machine learning.** Esta es una opción bastante lógica si se piensa en lo impredecible que es el mundo de la seguridad informática y la variedad de ataques que existen. Cabe resaltar que los conocimientos para desarrollar un malware son cada vez menos exigentes; la cantidad de herramientas existentes hace esto posible, incluso con la posibilidad de comprar malwares en el mercado negro **[1]**.

El análisis de malware cuenta con métodos clasificados según la forma en la que se recolectan los datos. Estos son el análisis estático y dinámico.**[2].** Por un lado el análisis estático extrae información binaria para generar un patrón del malware. Esto resulta ser ineficiente y fácil de evadir con técnicas de ofuscación **[3]**. El análisis dinámico, por su parte, observa y controla el comportamiento del software en ejecución en un ambiente controlado; esta técnica puede no ser muy efectiva por el tiempo prolongado que requiere su análisis y la variedad de comportamientos que tiene el malware en diferentes hosts **[4];** sin embargo, su precisión es mayor al evaluar su comportamiento y no el código binario.

La principal meta del artículo es evaluar el potencial del Machine Learning en el análisis de malwares, con el uso de algoritmos de clasificación automática. Así mismo, se busca evaluar extracciones de los indicadores y características del malware para su posterior análisis. Finalmente se intentará responder la pregunta siguiente: *¿qué tan difícil es evadir un anti-malware que utiliza machine learning?*

II. Machine learning

El aprendizaje automático o aprendizaje de máquinas (del inglés, "Machine Learning") es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender

- *Wikipedia*

En relación con el Machine Learning (ML) se podría decir, entonces, que es un tipo de inteligencia artificial que tiene como finalidad enseñar a un ordenador a tomar decisiones de forma autónoma a través de un entrenamiento a un modelo basado en un algoritmo. Dentro del **ML** se pueden agrupar los algoritmos de diferentes formas, los cuales terminan siendo variaciones de 2 tipos en esencia: **Aprendizaje Supervisado** y **No Supervisado.** La diferencia radica en los datos que tenemos para analizar**[5].**

En el **Aprendizaje Supervisado** los datos que poseemos tienen una estructura definida. Las características y etiquetas están correctamente mapeados. El modelo basado en un algoritmo es

entrenado para después hacer una predicción. Dentro del aprendizaje supervisado se hallan principalmente los problemas de tipo **Regresión** y **Clasificación[6].**

Regresión: Tiene como función predecir un resultado continuo basado en diferentes criterios. (Por ejemplo: hallar el valor de un vehículo basado su antigüedad, modelo, etc)

Clasificación: Tiene como finalidad asignar una categoría como resultado final. (Por ejemplo: clasificar imágenes por su contenido).

Para el correcto análisis de malwares utilizando Machine Learning, se tuvieron que evaluar diferentes métodos de **clasificación**. A continuación, sin ahondar mucho en aspectos técnicos, se describe de forma general el detalle de estos métodos.

1. K-Nearest Neighbor

Es un método de clasificación y predicción basado en la cercanía de K observaciones o *sample* **(FIG-I).** Esto quiere decir, que dentro de un conjunto de datos para un nuevo *sample* se buscan los más cercanos (si k=1, se considera Nearest Neighbor) y se realiza una clasificación basado en la estructura de datos recogidas.**[7]**

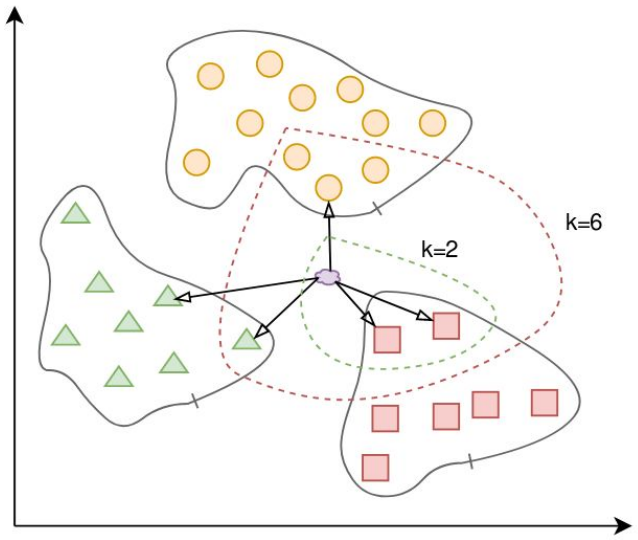

FIG I - Ejemplo de K-Nearest Neighbor. Para k=2 la clasificación es un *cuadrado*, mientras que para k=6 es un *círculo.*

Para su aplicación el nuevo *sample* es ubicado según sus características y es clasificado con respecto a los *samples* cercanos, para ello se mide la cercanía comúnmente a través de la **distancia euclidiana.**

2. Decision Tree

Es un método en el aprendizaje automático supervisado, que construye un árbol de decisión como modelo predictivo. La estructura se construye a partir de la raíz y se ramifica según las condiciones de sus atributos**[7].**

3. Random Forest

En general, muchos árboles de decisión son construidos y trabajan con un segmento del dataset de forma independiente. Esta colección de **Decision Trees** selecciona atributos para cada uno de forma aleatoria, y es entrenado sin tomar como referencia el resultado de los demás árboles. Funciona tanto en clasificación como en regresión.

4. Neural Network

Una red neuronal es una colección de neuronas artificiales que tienen diferentes entradas o *inputs* y a su vez una salida que está conectada a otras neuronas artificiales. Las entradas de estas neuronas son multiplicadas por un factor menor o igual a 1, y el resultado determina el *peso* de esta **[8].** Posteriormente son sumadas haciendo uso de una *función sigmoide*:

$$P(t) = \frac{1}{1 + e^{-t}}$$

Donde t = x , P(t) = Y

5. Algoritmo Naive Bayes

Es uno de los algoritmos más simples y fáciles de comprender. Naive Bayes es un clasificador basado en el teorema de Bayes, que trabaja de forma independiente los atributos de un *sample* para evaluar la probabilidad de las condiciones de un atributo en función a sus valores, con un conocimiento probabilístico de los datos previo. Su representación es la siguiente**[8]:**

$$p(C|F_1,\ldots,F_n) = \frac{p(C)\,p(F_1,\ldots,F_n|C)}{p(F_1,\ldots,F_n)}.$$

Se describe como **C** es la clase y $\mathbf{F_1}$ a $\mathbf{F_n}$ son las variables o datos nuevos ingresados. Entonces la probabilidad de **C** dado que $\mathbf{F_n}$ es verdadero, es igual a la probabilidad total de la clase(**C**) por la probabilidad de frecuencia de $\mathbf{F_n}$ siendo **C** verdadero, todo esto sobre la probabilidad total de $\mathbf{F_n}$.

El concepto general del aprendizaje supervisado se basa en 3 importantes etapas. La primera consiste en extraer los datos y etiquetarlos reconociendo el valor de cada uno, luego por medio del ***training,*** el modelo es entrenado con los patrones recogidos. Finalmente con datos totalmente nuevos se procede a probar el modelo y evaluar la predicción resultante.

Por otro lado, en el **Aprendizaje no supervisado** no existe una agrupación o etiquetado presente en los datos, siendo esta la función principal de este tipo de aprendizaje.

III. Malwares

El término malware proviene de la reducción de ***mal***icious soft***ware*** y puede tener muchas finalidades como comprometer equipos de cómputo, acceder a bases de datos o extraer información relevante. Solo en el 2016 se registraron más de **6 834 446** nuevos tipos de malwares**[9]**; lo que hace que sea en promedio más de **18 mil** nuevos malwares cada día, y en su mayoría afectan a equipos con sistemas operativos Windows.**[10]**
Hay diferentes tipos de malwares en la red que varían según su finalidad y la complejidad en su propagación, pero se pueden categorizar de la siguiente forma.

Adware/Spyware es un malware que tiene como finalidad enviar publicidad invasiva a su dispositivo. Casi siempre está acompañado de un **spyware,** el cual monitorea toda su información.

Trojans es un malware que a simple vista actúa como un software normal, pero tiene integrado funciones adicionales para comprometer su dispositivo.

Ransomware. Un malware que encripta los archivos de un computador con el fin de posteriormente pedir un rescate para poder recuperarlos.

Virus. Un programa malicioso que puede actuar como un virus biológico, pues es capaz de replicarse e infectar otros dispositivos.

Worms. Un malware que "devora" los archivos a su paso. Es capaz de autorreplicarse como lo hace un virus e infectar diferentes dispositivos.

Un malware puede infectar un dispositivo rápidamente. A través de un sitio web o una descarga de internet puede afectar desde un teléfono móvil hasta incluso un Smart TV. En solo los tres primeros meses de este año (2017) más de **260 000** usuarios fueron atacados por un ransomware, se realizaron más de **288 000** ataques al sector bancario y más de **un millón** de dispositivos móviles fueron infectados por un malware.**[9].** Es por ello que se sigue buscando nuevas y mejores formas de analizar y clasificar malwares.

1. Análisis de malwares

Actualmente el análisis de malwares tiene diferentes enfoques en los cuales trabajar. En la práctica diferentes técnicas son aplicadas en conjunto para obtener un resultado mucho más preciso. Según la forma de análisis se separa en 2 tipos, **análisis estático** y **dinámico.[11]**

Análisis estático: El malware no es ejecutado, se hace un análisis al *código muerto*. Esta información de forma individual o conjunta hace posible la detección del malware **[11]**. Es decir se extraen características del malware para posteriormente evaluar el impacto que tienen al momento de clasificarlo.

Análisis dinámico: Implica analizar la ejecución del malware. Se observan y controlan los procesos del software en tiempo de ejecución **[11].**

2. Detección de Malwares

Los antimalwares tradicionales hacen uso de diferentes procedimientos para detectar posibles amenazas. Estos métodos se pueden categorizar de la siguiente forma:

Signature-based: Un patrón o conjunto de características del software malicioso (**firma)** es extraído y añadido a las base de datos de firmas del antimalware; este luego cuando detecta un archivo con estas características, lo considera un malware **[11].** Este método es poco eficaz por no controlar y detectar malwares ofuscados o encriptados, y por el constante crecimiento de malwares, lo que hace casi imposible tener una base de datos de todos los patrones de todos los malwares existentes. Bajo un enfoque **heurístico[12],** el análisis estático se vuelve más eficiente, ya que no se aplica solo una coincidencia en el patrón de la firma, sino además se identifican reglas para detectar un comportamiento inusual.

Behavior-based: Detección basado en el comportamiento del malware durante su ejecución. Un antimalware con este tipo de detección es capaz de analizar el correcto comportamiento de un software. Si este hace algo que normalmente no debe hacer, será marcado como un peligro potencial **[13].** Este proceso suele tomar más tiempo y ser inexacto, ya que el malware puede actuar diferente según el host donde se encuentre.

3. Evasión

La evasión de antimalwares ha evolucionado a través del tiempo. Se han llevado a cabo diferentes tipos de métodos que permiten esto. El objetivo es obtener invisibilidad ante los softwares de detección de malwares y evitar la ingeniería inversa **[14].**

Encriptación: Se compone de dos secciones "*decryption loop*"**(FIG II),** el que realiza a su vez la encriptación y desencriptación; y el *"main body"* que es básicamente todo el código del malware.

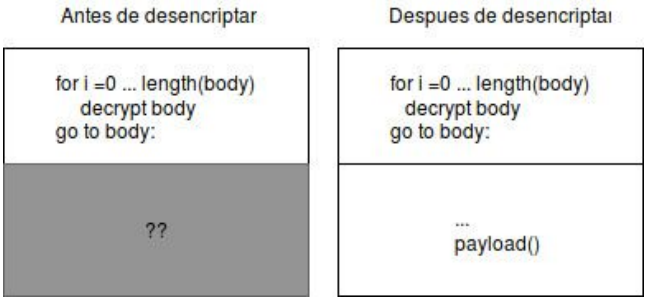

FIG II - Arquitectura de un malware.

En la encriptación más simple encontramos una transformación "*byte to byte*", que puede ser incremental (INC), negación (NEG), rotación izq/der (ROL o ROR), etc. En encriptaciones más complejas se puede hacer uso de operadores ADD o XOR **(FIG III)** e incluir una *clave* para realizar la encriptación.

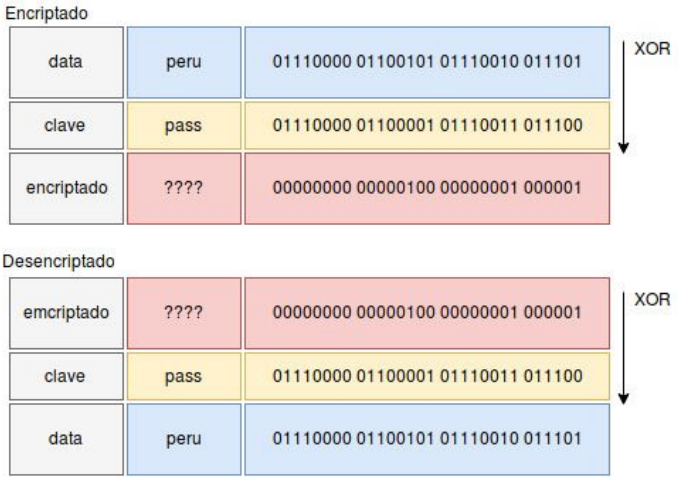

FIG III - Encriptación/Desencriptación de texto añadiendo una clave y utilizando el operador XOR.

Oligomorfismo o semi-polimorfismo establece una encriptación más compleja al código del malware. Su arquitectura contiene múltiples encriptadores que son escogidos de forma aleatoria según el host donde se encuentre **(FIG IV).**

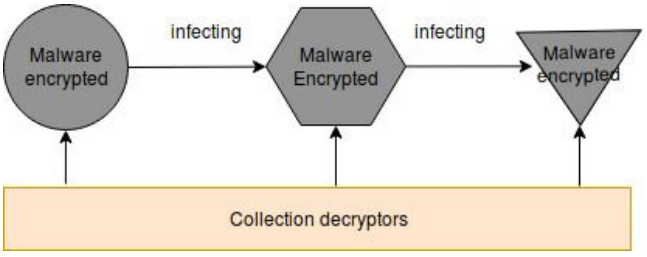

FIG IV - El malware cambia de encriptación en el proceso de infección a otro host basado en una colección de *decryptors.*

Polimorfismo Cambia el "main body" del malware en cada iteración, altera sus características para evadir detecciones; sin embargo, se sigue manteniendo las funciones y algoritmos establecidos inicialmente. Su principal función no es tener una colección de encriptadores si no generarlos de forma constante.

Metamorfismo A diferencia del resto, el metamorfismo no tiene un "encrypt/decryptor" que realice la tarea de encriptar el código y luego desencriptar. En este caso, es todo el código del "main body" el que cambia en cada iteración, lo que hace mucho más difícil su detección.

4. Ofuscación

Por otro lado tenemos la ofuscación o técnica para hacer que el código binario sea difícil o imposible de leer por otro software. Ello es así para evitar que sea interpretado o en muchas veces solo para evitar una copia o robo del código fuente. Esto es aplicado de diferentes formas, desde insertar código "*basura"* hasta permutar funciones. **[14].** Ello queda explicado en la siguiente imagen:

Binary Code Sequence	Assembly Code	
C7060F000055	mov	[esi], 5500000Fh
C746048BEC5151	mov	[esi+0004],5151EC8Bh
String Signature:	C7060F000055C746048BEC5151	

Binary Code Sequence	Assembly Code	
BF0F000055	mov	edi,5500000Fh
893E	mov	[esi],edi
5F	pop	edi
52	push	edx
B640	mov	dh,40
BA8BEC5151	mov	edx,5151EC8Bh
53	push	ebx
8BDA	mov	ebx,edx
895E04	mov	[esi+0004],ebx
String Signature:	BF0F000055893E5F52B640BA8BEC5151538BDA895E04	

FIG V - Inserción de código *basura*. La función sigue siendo la misma pero la forma en que se realiza es diferente, lo que da como resultado que sus firmas sean totalmente diferentes. (Imagen extraída de [14])

De esta forma, una colección de malwares puede tener el mismo funcionamiento y una sola finalidad; sin embargo las firmas son totalmente distintas, lo que en resumen hace mucho más complicada su detección.

IV. ML en el análisis de malwares

Una investigación reciente **[15]** realizada en Nigeria, utilizó el algoritmo *K-Nearest Neighbors* para detectar malwares en dispositivos android. El *paper* publicado describe el entrenamiento del modelo considerando las características (*features*) de un dispositivo android en un estado de normalidad y un dispositivo infectado con un malware, para luego analizar el comportamiento basado en estos resultados (llamadas, mensajes, estado del teléfono), con una precisión de clasificación de **93.75%** .

Una investigación más reciente **[16]** en el Reino Unido, enfocada en los mismos dispositivos, aplicó una metodología basada en la clasificación Bayesiana y el análisis estático. Se consideró además la recolección de características de una aplicación en android (apk) utilizando ingeniería inversa para construir el dataset, con esto se obtuvo 58 caracteristicas (*features*) de los apk que posteriormente fueron filtrados por una función de selección de *features.* Los resultados muestran que al entrenar el modelo con 15 a 20 *features* demuestra una mayor eficacia en la clasificación de malwares en contraste con la detección tradicional basado en firmas.

De esta forma, queda claro que el Machine Learning aplicado al análisis de malwares y con un correcto entrenamiento, puede resultar una mejor opción frente a los análisis tradicionales.

V. Experimentación

Con este conocimiento ya expuesto, se puede reconsiderar la complejidad en la detección de malwares. Por ello, a continuación, se describen las pruebas de concepto realizadas utilizando **Machine Learning** para la clasificación de malwares. Primero se **evaluaron y estructuraron los datos de los PE (Portable Executable)**, luego de la evaluación se **extrajeron las características** más relevantes para entrenar con esto al modelo. Finalmente se controlaron y **evaluaron los resultados** de la aplicación. Para todas las pruebas de **ML** se utilizó la herramienta **scikit-learn**.

1. Enfoque

El procedimiento está diseñado para analizar una colección de datos entre malwares y archivos no maliciosos utilizando **ML**, con el fin de predecir si un software es un malware o no. Se realizaron diferentes pruebas en función a las características obtenidas de los archivos y al modelo de **ML** utilizado. Finalmente se evaluó su efectividad en **3 escenarios reales.**

Para la elaboración de estas pruebas se consideró realizar un análisis estático con un enfoque heurístico, es decir analizar el malware sin la necesidad de ejecutarlo y a su vez extraer las características más relevantes.

2. Colección de datos (dataset)

La extracción de atributos fue el primer paso que se realizó. Para ello se utilizaron archivos ejecutables en entornos windows o *PE files (Portable Executable)*. Se contó con una colección de 20 mil archivos entre malwares y archivos no maliciosos para su desarrollo. Los malwares fueron extraídos de **Virusshare** y los archivos benignos desde sistemas operativos Windows 7 y Windows Server 2012.

Obtener catracterśticas del *Portable Executable*. Un PE se compone de cabeceras y secciones **(FIG-VI)**, que describen el funcionamiento e instrucciones del archivo. Todos estos atributos son solo datos en unidad, pero representan en conjunto la huella digital de un *PE file*, de la misma forma se detalla el uso y asignación de los datos en la memoria.

```
[IMAGE_OPTIONAL_HEADER64]
0x108      0x0   Magic:                        0x20B
0x10A      0x2   MajorLinkerVersion:           0x9
0x10B      0x3   MinorLinkerVersion:           0x0
0x10C      0x4   SizeOfCode:                   0x13C00
0x110      0x8   SizeOfInitializedData:       0x43EB800
0x114      0xC   SizeOfUninitializedData:     0x0
0x118      0x10  AddressOfEntryPoint:         0xDF08
0x11C      0x14  BaseOfCode:                  0x1000
0x120      0x18  ImageBase:                   0x140000000
0x128      0x20  SectionAlignment:            0x1000
0x12C      0x24  FileAlignment:               0x200
0x130      0x28  MajorOperatingSystemVersion: 0x5
```

FIG VI - Ejemplo de Cabecera de un archivo PE, en el cual se detallan algunas características del software. Lo que luego pasará a convertirse en *features* para entrenar al modelo de ML.

Estructurar datos del *Portable Executable*. No solo es necesario obtener las características del malware, adicionalmente a esto, es prioritario organizar estos datos para que el patrón que represente al malware sea específico y clave para la clasificación.

En un primer análisis se extrajeron los atributos convencionales de un **PE**, luego de esto se procedió a evaluar diferentes aspectos de 2 partes fundamentales de los archivos: El **nombre de las secciones** y las **funciones** utilizadas.

NonCharNameSection. En el primer escenario el nombre de la sección no estaba codificado para su lectura. Esto es un patrón concurrente en los archivos maliciosos. Como se muestra en la **Tabla I** una cantidad elevada de archivos no cuenta con una codificación de lectura correcta.

SuspiciousNameSection. Los malwares analizados tienen una característica en común: son modificados haciendo uso de un empaquetador *(packers)*, por ello se decidió establecer un atributo que determine la existencia y frecuencia de los nombres de las secciones. (UPX, .Themida, etc) como se muestra en la **Tabla I.**

HideCodeSection. Haciendo uso de empaquetadores se puede ocultar la sección de código o ***code section*.** Un malware que no contenga esta información es frecuentemente un archivo malicioso.

SuspiciousImportFunctions. El uso de funciones (API) poco habituales **(Tabla II)** en unidad y en conjunto considerando su frecuencia, es también un determinante para la clasificación de un malware. Si bien es cierto estas funciones no son maliciosas, el patrón de su uso en grupo nos da una aproximación.

Nombre de sección	Cantidad
invalid string	4264
.aspack	910
.Themida	329
UPX0/1/2/!	198
Otros	4310

Tabla I - Nombre de secciones extraídas de los archivos maliciosos. Se muestra un elevado número de nombres no codificados y empaquetadores habituales. Se omitieron los nombres de secciones comunes (.text, .data, etc)

Para el caso de las funciones se elaboró un proceso que extrajera las concurrencias de estas. Así pues, bajo la comparativa entre malwares y archivos no maliciosos se obtuvieron resultados que determinaron 3 características:

1. Se halló una clara diferencia entre la **cantidad** y **tipo de funciones** que utiliza un malware contra un archivo no malicioso.
2. Se pudo extraer más de 60 funciones asociadas a las **cantidades** y **promedios** de uso en cada malware.
3. Los malwares en proporción a los archivos no maliciosos hacen uso de funciones enfocadas a la **manipulación del sistema.**

FN Malwares	FN No Malware
AdjustTokenPrivileges	GetCurrentProcess
GetDlgItem	_initterm
LocalAlloc	_amsg_exit
EndDialog	_XcptFilter
LockResource	DeleteCriticalSection
LoadLibraryA	GetCurrentProcess
GetStartupInfoW	GetCurrentThreadId
AttachThreadInput	__CxxFrameHandler3
RegOpenKeyExA	LoadLibraryExW
__C_specific_handler	memcmp
GetAsyncKeyState	_wcsicmp
...	...

Tabla II - Nombre de funciones concurrentes en malwares contra funciones habituales en archivos no maliciosos. Catalogados y ordenados por la frecuencia de su uso.

Cabe resaltar que el uso de estas funciones no están directamente asociadas con los archivos maliciosos, sin embargo junto a la frecuencia y al uso de otras funciones en conjunto eleva el índice de probabilidad. Este primer paso, en síntesis, es el más relevante, puesto que determina qué atributos definen a un malware en comparativa a un archivo benigno. Con esta información recolectada se añadieron al dataset nuevos *features* que incluían estas características.

3. Entrenamiento

Con toda la información recogida se construyó el **dataset** que se usó para el entrenamiento, clasificando los *PE files* de la siguiente forma:

Clase	Valor
Malware	0
Archivo Benigno	1

Los resultados de entrenamiento por cada modelo entrenado fueron los siguientes:

Algoritmo	Score
KNeightbor	98.161 %
Decision Tree	99.029 %
Random Forest	**99.284 %**
Neural Network	75.383 %
Naive Bayes	40.909 %

Tabla III - Score del entrenamiento de los modelos de Machine Learning, utilizando scikit-learn. El mejor resultado lo obtuvo el algoritmo RandomForest.

Bajo un análisis **ROC** (Característica Operativa del Receptor)**(FIG VII)** para medir la precisión de los resultados, trazando la tasa de **falsos positivos** contra la tasa de **verdaderos positivos(Tabla IV)**, se obtuvo un resultado elevado en el área debajo de la curva (**AUC**) con **0.98.** Esto quiere decir que los resultados arrojaron una eficiente predicción en la clasificación de los datos de prueba.

Error	Tasa
Falso Positivo	0.202566 %
Falso Negativo	1.257862 %

Tabla IV - Tasa de falsos positivos y verdaderos positivos.

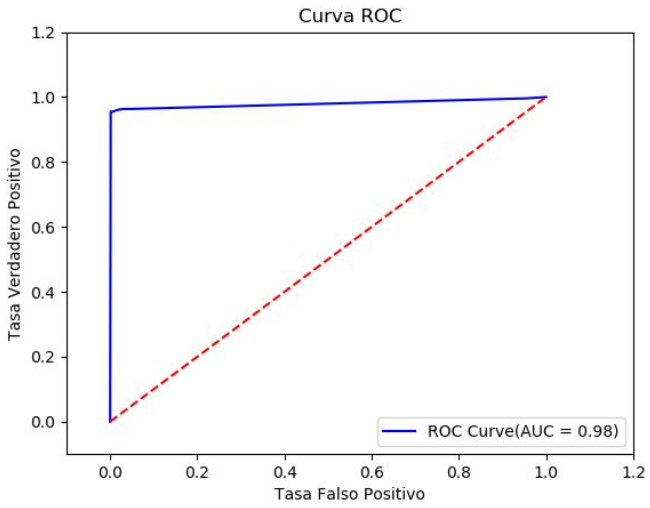

FIG VII - Análisis *ROC Curve*, con un resultado AUC de 0.98 para la clasificación de malwares.

Bajo el análisis de la matriz de confusión, **(FIG VIII)** para medir el performance del algoritmo, se pudo observar un índice elevado en la clasificación, con un mayor impacto en la detección de archivos maliciosos frente a los archivos benignos. Como se puede apreciar en el gráfico los errores de clasificación son muy bajos.

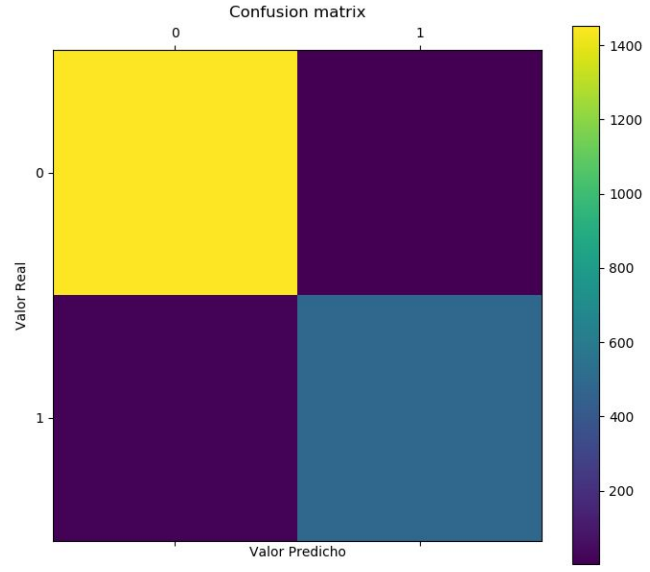

FIG VIII - *Confusion Matrix* o Matriz de Confusión, hace una comparativa entre los valores predichos y los valores reales.

Con este algoritmo de clasificación basado en los datos ingresados al modelo, se puede considerar una eficiente respuesta por parte del algoritmo. Bajo este enfoque y utilizando el modelo **RandomForest** se procedió a evaluar en escenarios reales la detección de malwares.

4. Evaluación

A continuación se detalla el procedimiento de evaluación en **3 diferentes escenarios**, para comprobar el rendimiento del clasificador. Esta pruebas se realizaron en comparación con un analizador online **(Virustotal)** y contrastar la información.

En el **Primer Escenario** se procedió a analizar malwares que han tenido su aparición este año (2017) y son relativamente nuevos. Para este caso analizamos un ransomware para windows: **GoldenEye.** La detección por parte de los antivirus tradicionales, **(FIG IX)** como era de esperarse, fue satisfactoria en su mayoría, siendo detectado como un ransomware, de la misma forma se analizó el mismo malware con el clasificador hecho en ML **(Tabla V),** obteniendo una clasificación correcta.

Nombre:	027cc450ef5f8c5f653329641ec1fed9.exe
Detecciones:	59 / 66
Fecha de análisis:	2017-11-24 04:10:59 UTC (hace 2 días, 16 horas)

FIG IX - Resultados de evaluación del ransomware GoldenEye. (Fuente:VirusTotal)

Probabilidad	Valor
P. Malware	**0.66671735389**
P. No Malware	0.33328264611

Tabla V - Resultados del clasificador de malwares.

En un **Segundo Escenario,** se decidió crear un virus desde cero, desarrollado en **lenguaje C,** en esta primera evaluación del virus no se utilizó ningún empaquetador, ofuscador o alguna técnica de evasión, se desarrolló utilizando métodos y funciones habituales y su única funcionalidad era crear una conexión inversa ***(reverse shell).*** Para ambos casos el resultado fue satisfactorio.**(FIG X)(Tabla VI).**

Nombre:	reverse_shell.exe
Detecciones:	19 / 67
Fecha de análisis:	2017-11-12 20:43:06 UTC (hace 0 minutos)

FIG X- Resultados de evaluación de un nuevo virus simple. (Fuente:VirusTotal)

Probabilidad	Valor
P. Malware	**0.743812773179**
P. No Malware	0.256187226821

Tabla VI - Resultados del clasificador de malwares.

Finalmente, en un **Tercer Escenario,** se modificó este malware utilizando técnicas de ofuscación, encriptación y empaquetadores. El malware en este caso está escrito en **C** con **powershell [17].,** realiza un *reverse shell* y su funcionamiento fue probado en un Sistema Operativo Windows 7 /Server 2012.
Según los resultados de esta tercera evaluación **(FIG XI)**, los antimalwares tradicionales no lograron detectar estas nueva variaciones realizadas en el archivo, por lo que el archivo malicioso es virtualmente indetectable ante la mayoría de antimalwares. Por otro lado se realizó la evaluación del malware con el clasificador de ML, y se obtuvieron resultados satisfactorios. **(Tabla VII)**

FIG XI - Resultados de evaluación de un nuevo virus. (Fuente:VirusTotal)

Probabilidad	Valor
P. Malware	**0.7812360807**
P. No Malware	0.21876391928

Tabla VII - Resultados del clasificador de malwares.

VI. Conclusion

Los resultados de las evaluaciones realizadas demuestran una mayor eficiencia en el análisis de malwares a través de una clasificación hecha con ML que integra el algoritmo **RandomForest**, frente a un análisis tradicional. Cabe resaltar que las predicciones de clasificación no son 100% confiables, sin embargo el índice de probabilidad correcta es elevado.

Bajo este enfoque, se pueden considerar muchas más alternativas en la selección de *features* que puedan optimizar el entrenamiento y por ende su clasificación, así mismo encontrar nuevas formas de evaluar estas características.

El objetivo principal de este artículo fue realizar el mejor procedimiento para la extracción de atributos para una eficiente clasificación usando Machine Learning, y se han logrado resultados satisfactorios.
Sin embargo las técnicas de ofuscación y de evasión siguen en una creciente constante y aún es necesario seguir evaluando más y mejores técnicas de detección.
En la evaluación del tercer escenario se consiguió evadir la mayoría de antimalwares, con técnicas de ofuscación nada nuevas, mientras que el clasificador logró detectarlo; se puede considerar entonces, que como herramienta automática para el análisis de malware puede resultar muy eficiente.

Con respecto a la pregunta *¿qué tan difícil es evadir un anti-malware que utiliza machine learning?,* la respuesta después del análisis realizado sería: Simple, si se tienen los recursos y conocimientos adecuados y complicado para malwares que ya circulan o que su desarrollo fue genérico. La sentencia que finalmente deja es sí en algún momento el análisis de malware aplicando **Machine Learning** podrá superar el análisis crítico de un analista.

Para probar el funcionamiento del clasificador, se encuentra disponible en: http://rufo.pro/

VII. Referencias

1. Andy Greenberg. (2015). *New Dark-Web Market Is Selling Zero-Day Exploits to Hackers.* 2017, de Wired URL: https://www.wired.com/2015/04/therealdeal-zero-day-exploits/
2. E.Gandotra, D. Bansal, S.Sofat (2014) Malware Analysis. En *Malware Analysis and Classification: A Survey* pp 57-58.
3. A. Moser, C. Kruegel, and E. Kirda (2007) *Limits of Static Analysis for Malware Detection.* pp 2-4.
4. M. Egele, T. Scholte, E. Kirda and C. Kruegel (2012) *A Survey on Automated Dynamic Malware Analysis Techniques and Tools*, En: ACM Computing Surveys, Vol. 44, No. 2. pp 8-13.
5. David Fumo. (2015). *Types of Machine Learning Algorithms* 2017, de Towards Data Science URL : https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861
6. Sebastian Raschka. (2014). *Predictive modeling, supervised machine learning, and pattern classification*, de sebastianraschka URL: http://sebastianraschka.com/Articles/2014_intro_supervised_learning.html
7. Sayali D.Jadhav (2014). *Comparative Study of K-NN, Naive Bayes and Decision Tree Classification Techniques,* En: IJSR, Vol. 5, Issue 1. pp 1842-1844
8. A. Khan, B. Baharudin, L. Hong Lee, K. khan. (2010). *A Review of Machine Learning Algorithms for Text-Documents Classification,* En: Journal of Advances in information technology, Vol. 1, No. 1. pp 9-13.
9. R. Unuchek, F. Sinitsyn, D. Parinov, V. Stolyarov. (2017). *IT threat evolution Q1 2017. Statistics.* 2017, de SecureList. URL: https://securelist.com/it-threat-evolution-q1-2017-statistics/78475/
10. Ralf Benzmüller (2017) *Malware trends 2017.* 2017, de G Data Security Blog. URL: https://www.gdatasoftware.com/blog/2017/04/29666-malware-trends-2017
11. A. Damodaran, Fabio Di Troia, C. Aaron Visaggio, Thomas H. Austin, M. Stamp. (2017). *A Comparison of Static, Dynamic, and Hybrid Analysis for Malware Detection.* En: Journal of Computer Virology and Hacking Techniques. Vol. 13, Issue 1, pp 4-6
12. Aryeh Goretsky. (2010) *What are Heuristics?* . 2017, de Welivesecurity. URL: https://www.welivesecurity.com/2010/12/29/what-are-heuristics/
13. Michael Kassner. (2010). *How antivirus software works: Is it worth it?*. 2017, de TechRepublic. URL: https://www.techrepublic.com/blog/it-security/how-antivirus-software-works-is-it-worth-it/
14. B. Bashari Rad, M. Masrom, S. Ibrahim. (2012). *Camouflage in Malware: from Encryption to Metamorphism,* En: IJCSNS, Vol.12, No. 8. pp 75-79.
15. J. Abah, Waziri O.V, Abdullahi M.B, Arthur U.M and Adewale O.S. (2015). *A Machine Learning Approach to Anomaly-Based detection on Android Platforms,* En: IJNSA Vol.7, No.6, pp 15-32
16. Suleiman Y. Yerima, S. Sezer, G. McWilliams and Igor Muttik. (2013). *A New Android Malware Detection Approach Using Bayesian Classification,* En: IEEE 27th International Conference, pp 1-7
17. *Nishang - PowerShell for penetration testing and offensive security.* URL: https://github.com/samratashok/nishang