Revista del Centro de Investigación de la Universidad La Salle
Vol. 11, No. 44, julio-diciembre, 2015: 69-102
http://ojs.dpi.ulsa.mx/index.php/rci/

Clasificación de Malware mediante Redes Neuronales Artificiales

Malware Classification by Artificial Neural Networks

Lilia E. GONZÁLEZ-MEDINA
Universidad La Salle, Ciudad de México (México)
Roberto A. VÁZQUEZ[1]
Universidad La Salle, Ciudad de México (México)

Fecha de recepción: diciembre de 2014
Fecha de aceptación: julio de 2015

Resumen

En la actualidad, ningún antivirus cuenta con un mecanismo de detección y clasificación totalmente efectivo para abarcar los miles de virus y programas maliciosos que se generan a diario. Por otro lado, se sabe que la mayoría de esas muestras son variaciones de programas maliciosos que ya se tienen identificados y que por lo tanto tienen similitudes estructurales. En ese sentido, para detectar estos programas se necesitan métodos capaces de clasificar programas maliciosos nuevos o variantes de estos aunque no se cuente con información específica en una base de datos. En este trabajo de investigación, se presenta un método para la detección de programas maliciosos basado en el número de veces que el programa llama a diferentes funciones de cada biblioteca de enlace dinámico (Dynamic Link Libraries, DLL). Una vez que se construye el vector descriptivo de cada programa, se entrena una red neuronal artificial para clasificar variantes de programas maliciosos en las familias correctas. Para validar el desempeño de la metodología propuesta se utilizó una base de datos que contiene muestras actuales y reales de programas maliciosos del tipo gusanos y troyanos.

Palabras clave: Clasificación de programas maliciosos, redes neuronales artificiales, reconocimiento de patrones

[1] Correo electrónico: ravem@lasallistas.org.mx

Abstract

At present, any antivirus has a fully effective detection and classification mechanism to detect the thousands of virus and malware that are generated daily. Furthermore, it is known that most of these samples are variations of known malicious programs and hence have structural similarities. In this regard, to solve this problem, several classification methods to detect new malware or variants thereof are needed. This research presents a method for detecting malware based on the number of times the program calls different functions of each dynamic link library (DLL). Once the feature vector of each program is computed, an artificial neural network is trained to classify variants of malware. To validate the performance of the proposed methodology a database containing current and real samples of worms and Trojans is used.

Keywords: Malware classification, artificial neural networks, pattern recognition.

ISSN 1405-6690 impreso
ISSN 1665-8612 electrónico

Introducción

La tecnología se vuelve cada día más esencial en nuestras vidas por lo que la mayoría de los dispositivos electrónicos que utilizamos requieren de una computadora para funcionar. Dependemos de dispositivos como laptops, celulares, entre otros, para comunicarnos, trabajar y almacenar información que no podemos perder. Sin embargo, el aumento en el uso de estos dispositivos también significa que la cantidad de datos sensibles en la red (nombres, direcciones, cuentas bancarias, contraseñas, contactos, números telefónicos, transacciones), tanto de usuarios como de empresas, va creciendo significativamente.

De acuerdo con Kaspersky (Kaspersky Lab, 2012), PandaSecurity (Panda Security, 2012) y Symantec (Norton by Symantec, 2013), la mayor amenaza a la seguridad de esos datos confidenciales son los programas maliciosos (malware), debido a que, en la actualidad, la mayoría son desarrollados con fines lucrativos o para obtener información de forma ilegal.

Atraídos por la sensación de anonimato que da internet, los criminales buscan obtener beneficios económicos por medio de fraudes, extorsiones, suplantación de identidad, robos a cuentas bancarias, comercio de datos personales, espionaje empresarial, etc. Otros peligros asociados a los programas maliciosos son: daños o pérdidas totales de software o hardware, saturación de redes, consumo excesivo de los recursos de un sistema, etc. Además, en la actualidad cualquier persona, sin importar si posee o no el conocimiento informático necesario, puede elaborar, modificar y distribuir nuevos códigos maliciosos en grandes cantidades, usando kits de creación de código malicioso. Desafortunadamente, la falta de leyes relacionadas con programas maliciosos y crímenes virtuales o el planteamiento incorrecto de ellas, provoca que para aquellos que se dedican a crear este tipo de programas dañinos haya mucho que ganar y poco que perder.

2011 fue el año en el que más firmas nacionales fueron el objetivo de ataques cibernéticos, especialmente a causa de troyanos (Chávez, 2012). El reporte de seguridad de ESET Antivirus indica que México, donde el 82.5% de las empresas cuestionadas dijeron haber sido infectadas durante el año, es el país con la mayor cantidad de incidentes provocados por programas maliciosos en Latinoamérica. En segundo lugar se encuentra Ecuador con 77.88% y en tercer lugar Perú con 56.02% (ESET, 2011).

Los laboratorios antivirus reciben diariamente miles de muestras potencialmente dañinas y a pesar de la gran cantidad de recursos que invierten en la detección y clasificación de programas maliciosos, no les es posible abarcar las millones de variaciones y la gran cantidad de muestras únicas de programas maliciosos que existen en la red.

De todos los tipos de malware que existen, las amenazas más comunes, mes con mes, son los troyanos y los gusanos (Global Research y Analysis Team, Kaspersky Lab, 2012). Detectar nuevas versiones de troyanos, virus y gusanos informáticos, así como las variaciones de los ya existentes, puede evitar que se infecten miles de computadoras, que se multipliquen, provoquen daños, roben información y se reenvíen a otros usuarios. Aunque la mayor parte de las muestras sean variaciones de malware que ya ha sido identificado por las compañías antivirus, pequeñas modificaciones hechas a los programas son suficiente para permitirles evadir la detección (Walenstein, 2007).

El objetivo de esta investigación es encontrar un método de clasificación de malware que además sea fácil de implementar con las tecnologías actuales.

Para detectar la mayor cantidad posible de muestras maliciosas antes de que infecten o ataquen un dispositivo, se necesita un método de clasificación de programas maliciosos que sea eficaz y capaz de detectar nuevas variantes incluso sin tener información específica de ellas, es decir, que pueda reconocer patrones presentes tanto en las muestras conocidas como en las muestras nuevas. La técnica de clasificación propuesta, que determina la familia a la cual pertenecen las muestras de programas maliciosos (variable dependiente) por medio de la cantidad total de funciones importadas de cada biblioteca del sistema (variables independientes), cumple con todos estos requisitos.

El método que se propone en esta investigación se basa en la observación de que muestras de malware de una misma familia o derivadas de ella, tiene estructuras similares y por lo tanto comparten una gran cantidad de bibliotecas y funciones del sistema. Al representar esta característica de los programas maliciosos de forma numérica, es posible utilizar esta información con cualquier clasificador, así como con múltiples tipos y arquitecturas de redes neuronales artificiales.

Si el método prueba ser eficaz en la clasificación de malware, podría implementarse con otros métodos ya existentes para mejorar el funcionamiento de los antivirus actuales, o incluso ser la base de un mecanismo completo de detección y desinfección; y de esta forma, disminuir considerablemente los problemas causados por malware, tales como robo de información o daños a los equipos de cómputo y otros dispositivos. Si bien la metodología propuesta puede ser muy simple, los resultados obtenidos superaron los resultados de varios programas de antivirus comerciales.

ISSN 1405-6690 impreso
ISSN 1665-8612 electrónico

Estado del arte

El abrumador incremento que el malware ha tendido en los últimos años ha aumentado la necesidad de mecanismos de clasificación de código malicioso que sean rápidos y efectivos; es por esto que se ha dedicado una gran cantidad de trabajos de investigación al análisis, la detección y la clasificación de malware. Al implementar mecanismos automáticos, las variaciones de programas maliciosos conocidos pueden ser detectadas y descartadas, disminuyendo así la cantidad de malware que los analistas reciben y permitiéndoles concentrar su atención en muestras no reconocidas.

Estos mecanismos de clasificación son la base de los motores antivirus. Un antivirus es un programa cuyo propósito es detectar, identificar y eliminar todo tipo de códigos maliciosos (virus, gusanos, troyanos, rootkits, etc.) en un dispositivo. En general, los métodos de detección y clasificación de malware se pueden dividir en tres: basados en firmas, basados en el comportamiento y heurísticos.

Métodos basados en firmas

Fue la primera tecnología de detección de malware empleada por las compañías antivirus; actualmente sigue siendo utilizada por ser la más efectiva, aunque complementada con técnicas proactivas, es decir, basadas en heurística. Consiste en una base de datos que contiene una definición para cada muestra de malware analizada (Shevchenko, 2007). Una definición, también conocida como firma, contiene una secuencia única de caracteres, pueden ser fragmentos de código o cadenas de texto encontradas en el ejecutable, que identifican de forma única un malware. El funcionamiento es simple, el archivo a clasificar se compara con la firma de cada muestra en la base de datos, si existe una coincidencia, el archivo es malicioso.

La principal desventaja de esta técnica radica en el hecho de que, para que un equipo pueda estar protegido contra amenazas, se debe estar actualizando la base de datos constantemente y el usuario debe descargar las actualizaciones tan seguido como sea posible; ya que para que un archivo sea detectado, su firma debe coincidir exactamente con alguna firma de la base de datos, es decir, no se puede detectar malware que no se encuentre en la base de datos. Otra desventaja es que cada variante debe contar con su propia firma. A este tipo de métodos se les conoce como protección reactiva, debido a que es necesario conocer de antemano un programa malicioso para que pueda ser detectado (ESET, 2011). Aunque está

técnica obtiene muy buenos resultados, la gran cantidad de variantes de malware que surge diariamente, provocó que este método se volviera lento y poco eficiente.

Existen métodos basados en la invocación de funciones de la interfaz de programación de aplicaciones. La investigación desarrollada por Abdulla (2012) utiliza las secuencias de llamadas API como entradas para un modelo que simula el fenómeno de Coestimulación que se produce en el interior de los sistemas inmunes humanos. Los autores proponen dos tipos de detectores: el primero evalúa las secuencias normales de llamadas API y el segundo se enfoca en cuatro grupos de comportamiento malicioso. Ambos detectores evalúan la muestra que se quiere clasificar y envían sus resultados al "Sistema Inmune", que incluye el uso de la distancia euclidiana como medida de similitud. Por último, para demostrar la eficacia de su enfoque con el modelo de coestimulación, comparan sus resultados con los obtenidos al utilizar dos redes neuronales: un mapa auto-organizado y un perceptrón multicapa con retropropagación.

Alazab, Venkataraman y Watters (2010), presentan un método automático de extracción y análisis de llamadas API para entender la conducta de programas maliciosos. Utilizan técnicas estáticas de detección de anomalías para determinar el comportamiento de un ejecutable y combinan la información obtenida con las llamadas API extraídas. Para llegar a la clasificación final en seis categorías de comportamiento sospechoso (buscar archivos para infectar, copiar o borrar archivos, obtener información del archivo, mover archivos, leer o escribir en un archivo, cambiar los atributos del archivo), los autores siguen una metodología de cuatro pasos: descomprimir el malware, desensamblarlo, extraer las llamadas API y partes importantes del código máquina del programa y finalmente, identificar las funciones del sistema con ayuda de la biblioteca MSDN para poder analizar el comportamiento. En sus resultados obtuvieron que el comportamiento más común de los programas maliciosos es infectar archivos mediante funciones que llevan a cabo lectura y escritura de archivos. Aunque su objetivo no es la clasificación de malware, sino entender su comportamiento, las funciones del sistema son la parte central de esta investigación y es por esto que se considera como un trabajo relacionado con el que se presenta en esta investigación.

Veeramani y Nitin (2012) proponen un método de clasificación que extrae las llamadas API que considera relevantes, para lo cual utilizan la frecuencia con la que son llamadas esas funciones como medida de selección, las funciones API llamadas con mayor frecuencia son descartadas, mientras que aquellas con menores valores de frecuencia son consideradas relevantes. El objetivo de los autores es identificar un conjunto de funciones que sean comunes en programas maliciosos y un conjunto de funciones comunes a programas

ISSN 1405-6690 impreso
ISSN 1665-8612 electrónico

normales. Como clasificador, los autores utilizaron una Máquina de Soporte Vectorial (Support Vector Machine, SVM), obteniendo un porcentaje de clasificación de 97.23%.

Zhang y Reeves (2007), presentan un enfoque para reconocer malware metamórfico por medio de fragmentos de código basados en las funciones del sistema llamadas por las muestras. Para determinar el grado de similitud entre dos patrones se utilizó un algoritmo de coincidencia, optimizado con el algoritmo Húngaro. Los resultados de similitud entre las muestras dieron porcentajes superiores al 95% en todos los casos; lo que, según los autores, demuestra que las modificaciones aleatorias causadas a las muestras no afectan la capacidad del método para reconocer la similitud entre el malware original y sus variantes.

Dahl (2013), propone una metodología basada en el número de llamadas a sistema y posteriormente realiza una selección de rasgos mediante proyecciones aleatorias. Una vez reducido el número de rasgos, una red neural es entrenada con esta información para la detección de malware. Después de entrenar la red neuronal los autores obtuvieron porcentajes de clasificación de hasta el 99.5%.

Alazab (2011) presenta una metodología basada en minería de datos para detectar familias de malware de reciente creación. Mediante el uso de diferentes técnicas de clasificación y el análisis del número de llamadas a sistema, la metodología propuesta logra alcanzar porcentajes de clasificación del 98.5%.

Métodos basados en el comportamiento

Esta técnica consiste en analizar las acciones que lleva a cabo un programa durante su ejecución, estas acciones se comparan contra un conjunto de reglas o políticas que establecen cómo debe ser el comportamiento de una aplicación. El programa es bloqueado si realiza actividades que no están definidas como aceptables, como por ejemplo: tratar de escribir en un área de memoria restringida o en ciertas partes del registro del sistema.

Los métodos basados en comportamiento por lo general llevan a cabo la ejecución de programas maliciosos en un entorno virtual, relativamente controlado, llamado sandbox, que restringe la interacción del malware con el sistema operativo. En un entorno virtualizado o máquina virtual, se ejecuta un sistema operativo dentro de otro, como si estuviera instalado; al ser sólo una simulación, un malware no tiene acceso al hardware y por lo tanto no puede causar daños graves. Para obtener información con este método, se monitorean y registran todos los eventos del sistema operativo, tales como: la creación, modificación o eliminación de claves de registro, archivos y procesos, actividad en la red,

etc. Algunas de las investigaciones desarrolladas en los últimos años sobre métodos basados en comportamiento se describen a continuación.

Bayer (2009) proponen un método de clusterización que identifica y agrupa muestras de malware con un comportamiento similar. Estos patrones de comportamiento se obtienen mediante un análisis dinámico, donde las muestras son ejecutadas durante cierto tiempo para conocer las llamadas al sistema invocadas, actividades en la red, objetos y operaciones del sistema operativo, etc. Posteriormente se utiliza el algoritmo LSH (Locality Sensitive Hashing) con los patrones de comportamiento para crear clústeres con muestras que actúen de forma similar. Uno de los puntos fuertes de este trabajo es su facilidad para analizar y clasificar grandes cantidades de malware en un tiempo aceptable, más de 75 mil muestras en menos de tres horas, con un 14% de falsos positivos y con un requerimiento de memoria de no más de 3.7 GB durante todo el proceso.

Lee y Mody (2006) proponen un método de clasificación que ejecuta un malware durante cierto tiempo y captura su comportamiento como una serie de eventos. Para medir el grado de similitud en las secuencias de eventos de grandes cantidades de objetos, utilizan el razonamiento basado en casos (case-based reasoning) y una forma adaptada de la distancia Levenshtein. Para el entrenamiento, donde se crean los clústeres representativos, aplicaron una variación del algoritmo k-medias; para la clasificación, emplearon el criterio del vecino más cercano (nearest neighbour). En sus resultados, la precisión aumentaba de forma directamente proporcional a la cantidad de clústeres. Los autores también encontraron que la precisión mejoraba al analizar un mayor número de eventos, obteniendo el rango de error más bajo con 500 eventos. Sin embargo, en uno de sus experimentos, si incrementaban la cantidad de eventos a 1000, el error aumentaba; esto debido a que un análisis demasiado específico del comportamiento del malware daba como resultado la creación de varios clústeres de tan sólo una o dos muestras.

En Rieck (2008), su procedimiento para clasificar muestras abarca tres pasos: se recolectan muestras de malware y se monitorean, utilizan un antivirus para nombrar las muestras y se clasifican según su comportamiento, para después servir como entradas del algoritmo SVM (Support Vector Machines). Con base en las características del comportamiento de las muestras durante el tiempo de ejecución, se generan reportes. El número de ocurrencias de todas las cadenas que forman el reporte se guarda en un vector. Según sus resultados, 88% de los ejecutables fueron clasificados en la familia correcta, en el experimento de predicción de familias el 69% de las muestras no identificadas por el antivirus Avira Antivir fueron correctamente clasificadas por su enfoque y el 100% de ejecutables no malignos fueron detectados correctamente.

ISSN 1405-6690 impreso
ISSN 1665-8612 electrónico

Métodos heurísticos

Gheorghescu (2006) introduce un sistema de clasificación basado en comportamiento evolutivo que, usando una computadora de escritorio promedio, compara muestras de programas maliciosos con toda la colección de malware usando tres algoritmos de correspondencia aproximada; evaluando su tiempo de ejecución y requerimientos de memoria, bajo la premisa de que muestras de malware de la misma familia o derivadas del mismo código fuente, comparten una gran cantidad de bloques básicos de código.

Carrera y Erdélyi (2004) utilizan el desensamblador IDA Pro y el lenguaje de programación Python, para encontrar similitudes y diferencias entre las variantes y cepas de malware mediante la teoría de grafos. Su método consiste en reconocer características similares en la estructura del ejecutable, cuyo código es representado como grafo de control de flujo (Control Flow Graph), donde las funciones son los vértices y las llamadas entre funciones son las aristas.

Kinable y Kostakis (2010), estudian un método de clasificación de malware por clusterización que utiliza los algoritmos de k-medias y el método basado en densidad DBSCAN para comparar malware utilizando grafos. Al igual que en otros métodos, las funciones son los nodos y las llamadas entre funciones son las aristas. Para medir la similitud estructural entre los grafos, los autores emplearon la Distancia de Edición de Grafos, donde aquellas muestras con estructuras similares tenían que ser clasificadas en la misma familia. Su base de datos consistió en 194 muestras, clasificadas manualmente por analistas en 24 familias. En los experimentos realizados con una variación del algoritmo k-medias, los resultados obtenidos no fueron satisfactorios debido a que no fue posible repartir todas las muestras en clústeres definidos, de tal forma que se obtuvieran las 24 familias tal como en la clasificación manual. La clusterización por medio del algoritmo DBSCAN, en donde se descartan las muestras cuya pertenencia a un clúster no esté definida, dio mejores resultados. En la mayoría de los casos, cada clúster creado por el algoritmo DBSCAN representa a una familia de malware, sin incluir muestras pertenecientes a otras familias. Además de que la cantidad de muestras descartadas, consideradas ruido por el algoritmo, fue muy baja.

Análisis de los métodos descritos

Hasta ahora, las únicas técnicas de clasificación enfocadas en funciones del sistema se basan en alguno de los siguientes enfoques:

1. Secuencias de funciones para detectar comportamiento malicioso en un programa.

2. La frecuencia con la que son llamadas ciertas funciones.

Estos enfoques presentan al menos uno de estos inconvenientes:

- Al depender de la secuencia en la que las funciones son llamadas, se ven afectados por las técnicas empleadas por el malware polimórfico o metamórfico, que reordenan las llamadas API.

- Sólo se consideran ciertas funciones o secuencias de funciones y se ignoran muchas otras, que podrían aportar información importante de las muestras.

A pesar de que estos trabajos son interesantes y presentan buenos resultados, todos ellos tienen desventajas y limitaciones importantes, tales como: algunos clústeres formados tenían un porcentaje de similitud de poco más del 50%, el tiempo de ejecución puede no ser suficientemente largo para evaluar el comportamiento de las muestras, las muestras que dependen de un cierto tiempo o actividad pueden no ser evaluadas correctamente, son ineficientes en el manejo de grandes cantidades de malware, se requiere una gran cantidad de memoria o de tiempo, no toman en cuenta muestras metamórficas ni polimórficas o dependen del flujo de ejecución que puede cambiar dinámicamente debido a interrupciones o restricciones de acceso.

Esta investigación analiza el potencial de un método de clasificación de malware basado en la cantidad de funciones del sistema (APIs) por biblioteca de enlace dinámico (DLL). Este enfoque se eligió debido a que los métodos de clasificación seleccionados para los experimentos están basados en reconocimiento de patrones, por lo que trabajan exclusivamente con valores numéricos, y durante el análisis de la información que es posible extraer de un ejecutable, surgió la idea de contar la cantidad de funciones que los programas llaman de cada biblioteca y con esos valores crear un vector, que podría ser utilizado como entrada para distintos clasificadores y tipos de redes neuronales. Además, al considerar únicamente valores numéricos, sin importar tomar en cuenta exactamente qué función se esté llamando, se puede afirmar que el enfoque propuesto en esta tesis es

ISSN 1405-6690 impreso
ISSN 1665-8612 electrónico

independiente del comportamiento y no se ve afectado por las técnicas que reordenan la secuencia de llamadas API.

Es importante mencionar que, aunque en varios de los trabajos aquí mencionados se dice que es un problema tener que desensamblar las muestras debido a la dificultad que presenta obtener información si están comprimidas (packed) u ofuscadas, la funcionalidad de descompresión (unpacking) ya está disponible en la mayoría de los antivirus actuales y es independiente del método de detección y clasificación.

Redes Neuronales Artificiales

Las redes neuronales son sistemas formados por unidades de procesamiento conectadas entre sí, que son capaces de analizar grandes cantidades información con el fin de resolver problemas complejos, tales como: aprender, memorizar, reconocer, clasificar, etc. En los últimos años, se ha desarrollado una gran cantidad de trabajos de investigación enfocados en crear modelos que permitan resolver estos problemas de forma similar a como lo haría un humano; sin embargo, estos avances están limitados por la tecnología actual y el conocimiento insuficiente que se tiene del funcionamiento del cerebro.

Una Red Neuronal Artificial (RNA) es un sistema de procesamiento paralelo donde las neuronas, unidades más simples representadas como nodos, están conectadas entre sí, en forma de grafo dirigido, para la solución de problemas concretos. Una RNA trata de simular el funcionamiento del cerebro humano, por lo que las neuronas deben almacenar conocimiento y aprender de la experiencia (Freeman, 1993). Una RNA se caracteriza por los patrones que presentan las conexiones entre neuronas (arquitectura), su método para determinar los pesos en las conexiones (algoritmo de aprendizaje) y su función de activación. También debe ser capaz de generalizar el conocimiento, es decir, tras aprender una serie de patrones, puede reconocer otros similares aunque no se hayan presentado antes.

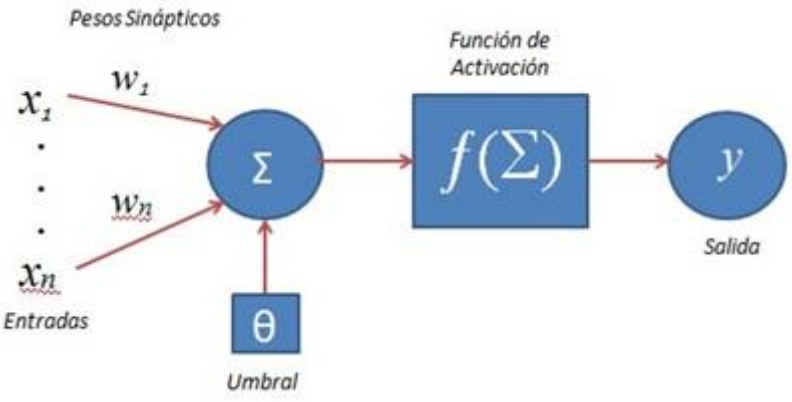
El modelo de un perceptrón simple se muestra en la Elaboración propia Como se puede ver, se tiene una serie de entradas x, que representan la información proveniente de otras neuronas o del exterior. Los pesos sinápticos w deben ser inicializados con valores aleatorios y representan las conexiones sinápticas entre las neuronas. El umbral, denotado por θ, es un valor fijo que se suma a las entradas pesadas. El umbral permite que las líneas de clasificación que dividen el espacio, puedan orientarse para separar vectores de entrada que pertenecen a clases distintas pero que no se encuentran en lados opuestos del origen.

Sin el umbral, las líneas de clasificación tendrían que pasar por el origen, es decir, no se podría ajustar para clasificar las entradas de la forma correcta.

Las entradas y los pesos se multiplican y se van sumando, como se muestra en la ecuación (1), que es el valor de activación de la neurona de salida (Aberdeen's Robert Gordon University, 2005).

$$y' = \sum_{i=1}^{n} w_i x_i + \theta \tag{1}$$

Figura 1
Modelo básico de un perceptrón



Elaboración propia

Por último, la salida definitiva y se produce al pasar y' por la función de activación, ecuación (2).

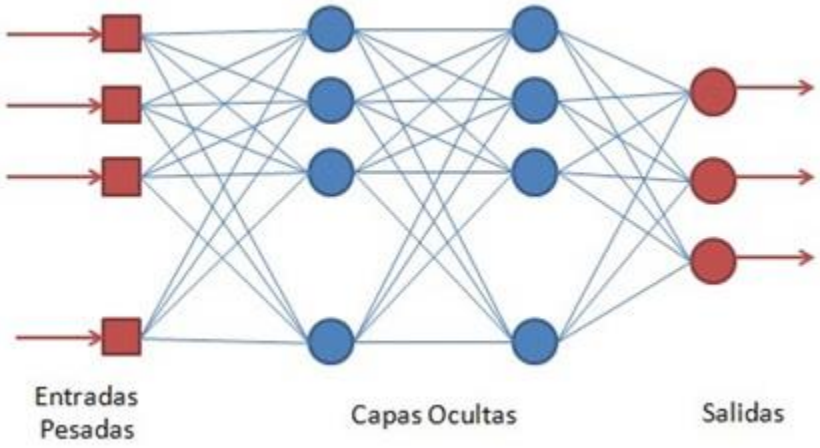
$$y = f(y') \tag{2}$$

A este proceso, donde los valores de entrada se van procesando de neurona en neurona hasta llegar a la capa final, se le conoce como propagación hacia adelante. Durante el aprendizaje, si la salida de la red no corresponde con el valor deseado, es decir, si hay un error de clasificación, es necesario modificar los pesos de las conexiones sinápticas.

ISSN 1405-6690 impreso
ISSN 1665-8612 electrónico

El perceptrón multicapa es una serie de neuronas idénticas al modelo de la Elaboración propia, conectadas entre sí y agrupadas en capas de diferentes niveles, para formar una red de neuronas, como se aprecia en la figura 1.

Figura 1
Modelo de un perceptrón multicapa



Elaboración propia

Las capas pueden ser de tres tipos: de entrada, ocultas y de salida. Las neuronas de la capa de entrada se encargan solamente de recibir los patrones y propagarlos a las neuronas de la capa siguiente. La capa de salida proporciona la respuesta de la red para un patrón de entrada dado. Mientras que las capas ocultas son las que llevan a cabo el procesamiento de los patrones recibidos. Las neuronas se conectan hacia adelante con las neuronas de la siguiente capa, por lo que reciben el nombre de redes alimentadas hacia adelante o redes "feedforward". La función de activación utilizada en los experimentos con el perceptrón multicapa es la tangente sigmoidal hiperbólica, ecuación (3), que limita la amplitud de salida de la neurona a un rango de -1 a 1.

$$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{3}$$

El algoritmo de retropropagación, también llamado regla delta generalizada, se obtiene al aplicar el método del gradiente descendiente de tal forma que el error obtenido en las neuronas de la capa de salida es propagado hacia atrás para corregir el error en cada una de las neuronas de las capas ocultas de la red. El factor de aprendizaje α determina cuánto van a cambiar los valores de los pesos y de los umbrales; sin embargo se debe considerar que mientras más grande sea este valor, más inestable será el algoritmo y si el factor de aprendizaje es muy pequeño, el tiempo que tardará en llegar al error deseado va a ser muy largo.

Metodología propuesta

Existe una gran cantidad de proyectos de investigación enfocados en la clasificación de malware, donde la problemática es abordada desde diferentes perspectivas y los resultados obtenidos suelen ser satisfactorios; sin embargo, al analizar con distintos antivirus cualquier programa infectado, es evidente que ninguno de los métodos de clasificación empleados actualmente es totalmente eficaz.

Por otro lado, los procedimientos que requieren de un análisis del comportamiento de las muestras durante su ejecución, análisis dinámico, suelen requerir una mayor cantidad de recursos, tanto computacionales como de tiempo, debido a que se necesita usar distintos programas para monitorear actividades específicas, como la creación de claves de registro y el envío de paquetes de datos en la red. Por consiguiente, los enfoques estáticos son necesarios.

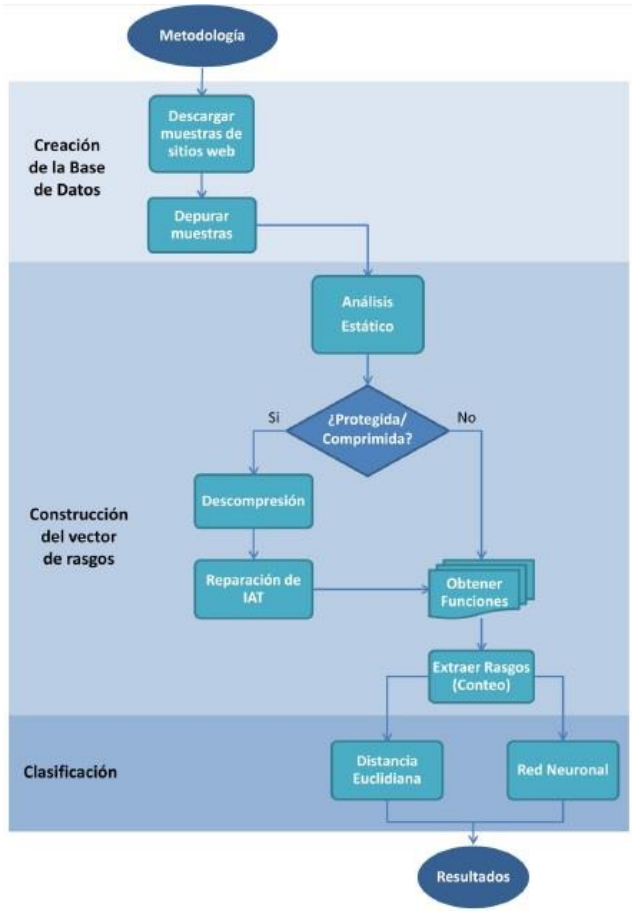
Esta investigación parte de la idea de que las muestras de malware pertenecientes a la misma familia, o derivadas de ella, comparten una gran cantidad de funciones del sistema. El objetivo de esta investigación es encontrar un método que sea capaz de clasificar muestras de malware de forma correcta, utilizando únicamente esas funciones y sus correspondientes bibliotecas (DLLs). Como se mencionó anteriormente, las funciones y bibliotecas del sistema, que forman el vector numérico, son las variables independientes y toman diversos valores según la muestra; la familia de malware es la variable dependiente, dado que su valor se obtiene según el patrón que tenga el vector de la muestra.

En términos generales, el método propuesto consiste en representar una muestra de malware como un vector de características numéricas, donde cada característica contiene la cantidad de funciones API llamadas de una biblioteca de enlace dinámico distinta.

ISSN 1405-6690 impreso
ISSN 1665-8612 electrónico

En la figura 2 se muestra un diagrama con la descripción general de las tres etapas que conforman la metodología de esta investigación: la creación de la base de datos, la construcción del vector de rasgos y la clasificación de muestras.

Figura 2
Descripción general de la metodología propuesta



Elaboración propia

El tipo de malware que se usó en este proyecto únicamente incluyó muestras de gusanos y troyanos. También se descartaron otros tipos de malware como: backdoor, rootkit, spyware, exploits, adware, etc. Esto debido a los siguientes factores:

1. La dificultad que presenta encontrar muestras de esos tipos que pertenezcan a la misma familia.
2. Las amenazas más comunes son troyanos y gusanos.

Antes de incluir otros de los tipos existentes de malware, se quiso comprobar que este método funcionara con los dos más comunes.

Creación de la base de datos

Se consideró la opción de establecer un ambiente seguro usando una máquina virtual con un sandbox, que nos permitiría atrapar software malicioso y poder trabajar con él sin dañar o infectar la computadora. La desventaja es que no se tiene control sobre el tipo de malware que ataca y las probabilidades de conseguir varias muestras de la misma familia no eran tan altas.

Se tomó la decisión de buscar páginas web que tuvieran bases de datos de malware disponibles para descargar. Las muestras iniciales de malware fueron obtenidas de las páginas Offensive Computing (Quist, 2005) y VX Heavens (VX Heavens, 2012). No se descargaron archivos de forma aleatoria, se trató de buscar la mayor cantidad posible de variaciones de una familia. Sin embargo, se tuvieron algunas dificultades:

1. Puesto que ambas páginas adquieren una parte de sus muestras a partir de contribuciones de usuarios, algunas variaciones estaban clasificadas incorrectamente.

2. Muchas familias tenían menos de 3 variaciones.

3. Ambas páginas tenían muestras similares.

Construcción del vector de rasgos

Se procedió a extraer las funciones importadas mediante el análisis estático de cada ejecutable en una máquina virtual con el sistema operativo Windows XP Service Pack 3. Se instaló el antivirus Microsoft Security Essentials con el fin de limpiar el sistema después de la ejecución de una muestra de malware; se eligió este antivirus debido a su bajo consumo de recursos. Para evitar que el antivirus interfiriera con el análisis o eliminara las muestras de la base de datos, se configuró el programa para incluir en su lista de excepciones las carpetas donde se lleva a cabo el análisis y donde se almacenan las copias de los gusanos y troyanos. Si la imagen de la máquina virtual no podía ser reparada después de la ejecución de una muestra de malware, simplemente se regresaba a una de las imágenes anteriores, que son copias del sistema con el software necesario para el análisis pero sin infección.

ISSN 1405-6690 impreso
ISSN 1665-8612 electrónico

Para obtener las funciones que cada programa malicioso importa de las bibliotecas dinámicas del sistema (*.dll) se utilizó el desensamblador IDA Pro. Esas funciones fueron guardadas en archivos de texto, que en adelante serán referidos como archivos de funciones importadas.

Si IDA Pro no podía obtener las funciones importadas, posiblemente el archivo estaba comprimido (packed), esto se confirmaba con PEid y RDG Packer Detector, que detectan los compiladores y herramientas de compresión usados comúnmente en ejecutables. Posteriormente se usó OllyDbg para analizar aquellas muestras que IDA Pro no pudo abrir correctamente debido a que estaban protegidas con herramientas de compresión de ejecutables, también conocidas como packers, como los mencionados en el capítulo anterior, por ejemplo: UPX, PECompact y ASPack. Por último, para obtener las funciones importadas, se necesitó el software ImpRec, que se usó junto con OllyDbg para reconstruir las funciones importadas de un ejecutable Win32 protegido o comprimido, ver figura 3.

En algunos casos, muestras de la misma familia parecían ser significativamente diferentes entre sí. Para confirmar que en realidad pertenecían a una familia, las muestras fueron analizadas mediante alguno de los procedimientos que se explican a continuación:

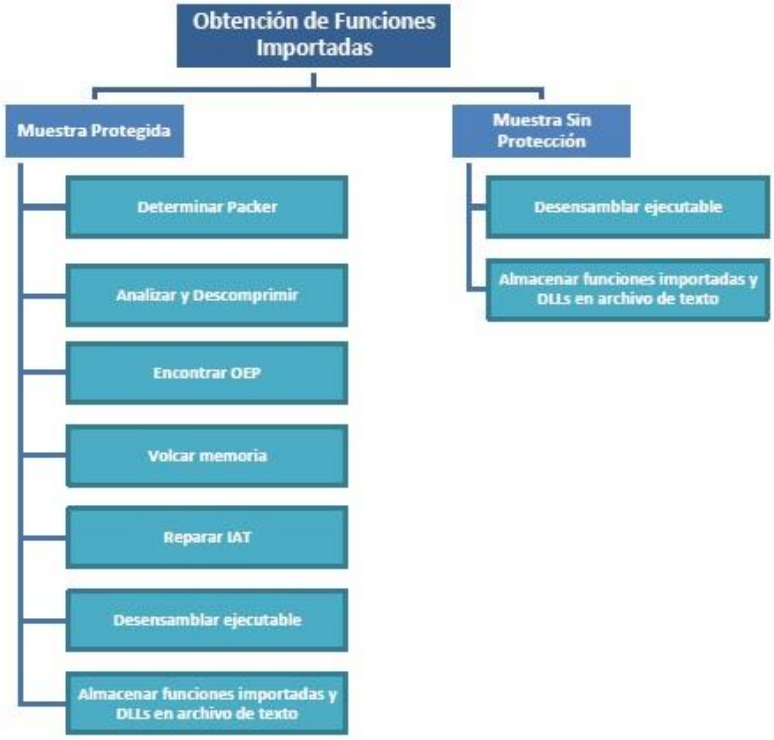
1. Utilizando el programa Maltrap, que es una herramienta cuyo propósito es llevar a cabo un análisis dinámico de programas maliciosos, se crearon reportes de las acciones que el malware realizó durante su ejecución.

2. El programa malicioso era ejecutado para observar su comportamiento.

Los resultados se compararon con las descripciones técnicas, realizadas por analistas de malware, encontradas en los sitios web de Securelist o del antivirus F-Secure.

Finalmente, en un archivo de texto se creó la primera base de datos de malware. Este archivo incluye el nombre del malware en la primera columna y la cantidad de funciones del sistema que se llaman de cada biblioteca de enlace dinámico en las columnas restantes. Fue importante crear este archivo para conocer qué tan viable iba a ser este método. A primera vista, es posible apreciar las semejanzas entre los vectores de las muestras que forman una familia y las diferencias con los vectores de familias diferentes.

Figura 3
Diagrama del procedimiento para obtener DLLs y funciones importadas

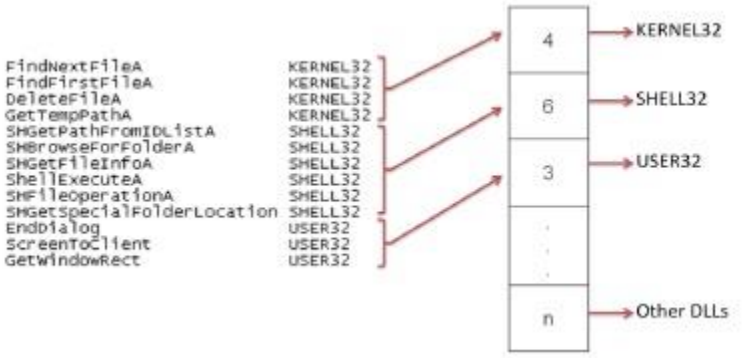


Elaboración propia

Después de finalizar el análisis estático de todos los ejecutables, los archivos de funciones importadas de cada muestra pasaban a través de una etapa de conteo. Para cada archivo se realiza el conteo de funciones por biblioteca, que básicamente consiste en contar el número de funciones que se usa de cada DLL, es decir, la cantidad de veces que se repite el nombre de una biblioteca en cada renglón de la tercera columna del archivo de texto, como se observa en la figura 4. El orden en el que se van guardando los nombres de las bibliotecas en esta base de datos, es denotado por un identificador o id y determina la posición del vector en la que se va a almacenar el total de funciones llamadas de cada biblioteca.

ISSN 1405-6690 impreso
ISSN 1665-8612 electrónico

Figura 4
Proceso del conteo de funciones para la construcción del vector de rasgos



Elaboración propia

Clasificación de los vectores de rasgos

En esta etapa se llevó a cabo la clasificación de muestras maliciosas, según sus rasgos o características, en las familias más similares. Como se mencionó anteriormente, el archivo generado en la etapa anterior contiene la representación vectorial de todas las muestras de la base de datos. En todos los experimentos, estas muestras se dividieron en dos grupos: el grupo de entrenamiento o de referencia, que se necesita para crear el vector de referencia de cada familia, y el grupo de prueba, que se utiliza para estimar la efectividad del método.

Una vez definidos ambos grupos, se llevaron a cabo los experimentos para evaluar el desempeño del método propuesto con diferentes técnicas de clasificación. En el caso particular de esta investigación, se decidió entrenar una red de Perceptrón Multicapa, enseñando a la red los patrones característicos de cada familia mediante las muestras del conjunto de entrenamiento, modificando los pesos para obtener la salida correcta y clasificando las muestras del conjunto de prueba.

Resultados experimentales

Para evaluar el desempeño de la metodología propuesta, se utilizó una base de datos compuesta de 245 muestras de gusanos y troyanos de 24 familias diferentes. De cada una de las muestras se extrajo un vector de características; donde cada característica representa una biblioteca de enlace dinámico encontrada en las muestras. El número total de características considerado para construir cada muestra fue de 39, donde el valor de cada

característica se obtuvo a partir de la cantidad de funciones importadas del sistema o APIs que el Ejecutable Portable llama de cada biblioteca.

Para evaluar el comportamiento general del método se realizó una comparativa entre una técnica de clasificación basada en la Distancia Euclidiana y una red de perceptrones multicapa. Para validar estadísticamente los resultados obtenidos, se realizaron dos series de treinta experimentos cada una. En la primera serie, se crearon el grupo de entrenamiento y el grupo de pruebas, cada uno con el 50% de las muestras de la base de datos. En la segunda serie, el 80% de las muestras de la base de datos fueron utilizadas para crear el grupo de entrenamiento, mientras que el 20% restante se utilizó para crear el grupo de pruebas. En cada uno de los 30 experimentos de las series, las muestras de ambos grupos de datos fueron seleccionadas de forma aleatoria.

Una serie se divide en dos fases:

1. Entrenamiento: En el caso de la distancia Euclidiana, se calculan las distancias entre cada muestra del grupo de entrenamiento y los vectores promedio de todas las familias. En el caso del perceptrón multicapa, se entrena la red para reconocer los patrones que corresponden a cada familia. Este paso es importante para tener una idea del porcentaje de precisión más alto que es posible obtener; puesto que se clasifican las mismas muestras que se utilizaron para obtener los vectores promedio o entrenar a la red, el porcentaje de clasificación correcta será mayor que en la fase de prueba.

2. Prueba: En el caso de la distancia Euclidiana, los vectores promedio se crean con el grupo de entrenamiento y se obtienen los resultados finales al calcular las distancias para cada muestra del grupo de prueba. En el caso del perceptrón multicapa, las muestras del grupo de prueba son clasificadas utilizando la información obtenida durante la etapa de entrenamiento.

El porcentaje promedio de clasificación obtenida y la desviación estándar, así como los porcentajes mínimos y máximos obtenidos en los experimentos utilizando la Distancia Euclidiana como clasificador, se muestran en la Tabla 1.

El mejor promedio de clasificación en la fase de prueba se obtuvo al utilizar el 50% de las muestras para generar los vectores de referencia. Esto puede deberse a que al utilizar el 80% de las muestras para crear los vectores promedio, se incluyen algunas cuyos vectores son significativamente diferentes a los de las otras muestras de la familia.

ISSN 1405-6690 impreso
ISSN 1665-8612 electrónico

Tabla 1
Resultados generales de los experimentos con Distancia Euclidiana

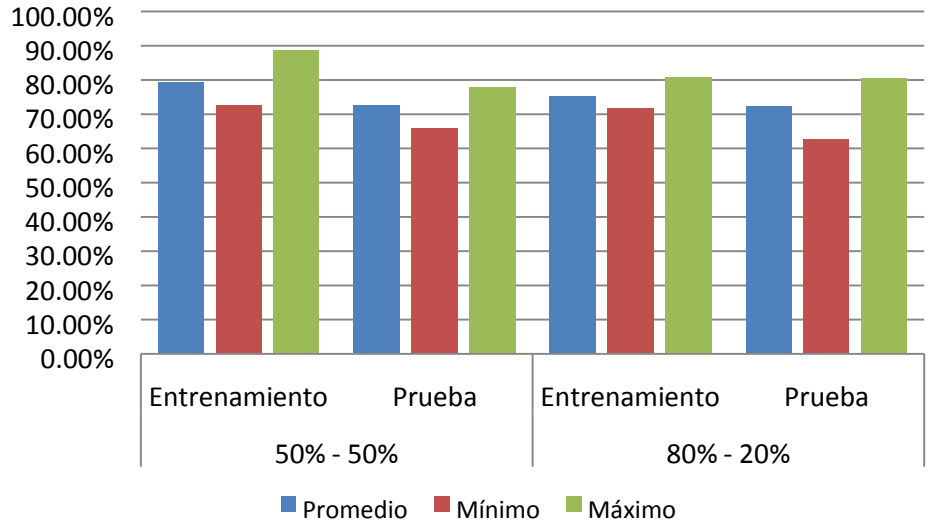
Distancia Euclidiana				
	50% - 50%		**80% - 20%**	
	Entrenamiento	**Prueba**	**Entrenamiento**	**Prueba**
% Promedio	79.26%	72.76%	75.35%	72.35%
% Mínimo	72.52%	65.81%	71.57%	62.75%
% Máximo	88.55%	77.78%	80.71%	80.39%
Desviación Estándar	0.03473	0.02890	0.02185	0.04410

Elaboración propia

Por otro lado, los valores de la desviación estándar nos indican que los porcentajes de clasificación obtenidos en los experimentos presentan un menor grado de dispersión con respecto al porcentaje promedio, es decir, los resultados obtenidos en cada experimento no varían mucho entre sí y se acercan al valor promedio.

Aunque el valor máximo en la fase de prueba se obtuvo utilizando el 80% de las muestras en el entrenamiento. Como se observa en la figura 5, el valor máximo obtenido durante la etapa de entrenamiento fue menor a 90%; esto significa que, sin importar qué muestras se seleccionen para el conjunto de datos, no va a ser posible obtener resultados satisfactorios utilizando el enfoque de vectores propuesto con este método de clasificación.

Figura 5
Resultados obtenidos utilizando distancia Euclidiana como método de clasificación.



Elaboración propia

Incluso se puede afirmar que la probabilidad de obtener un porcentaje de clasificación muy bajo aumenta al incluir una mayor cantidad de muestras en el conjunto de entrenamiento, tal como se observa en la figura 5, donde el porcentaje de clasificación más bajo fue obtenido empleando 80% de las muestras en el entrenamiento. En general, los resultados obtenidos con la distancia Euclidiana no son satisfactorios, considerando además que comparar una muestra con todos los vectores promedio de una base de datos, con millones de muestras, sería imposible de llevar a cabo en un tiempo razonable. Es por esto que se decidió experimentar con otro método de clasificación.

El segundo método de clasificación utilizado para comprobar la eficacia del enfoque propuesto, vectores que utilizan como características la cantidad de llamadas API de cada DLL, es una Red Neuronal Artificial de tipo Perceptrón Multicapa. Para que sea posible comparar estos resultados con los alcanzados en los experimentos con la distancia euclidiana, se utilizan los mismos porcentajes para formar los grupos de entrenamiento y prueba, es decir, primero se evalúa con 50% - 50% y posteriormente con 80% - 20%.

Al emplear el esquema “winner-takes-all” sólo puede haber una neurona ganadora, por lo que cada patrón de entrada es clasificado a una sola familia de malware.

ISSN 1405-6690 impreso
ISSN 1665-8612 electrónico

El número de neuronas de entrada es 39 y el número de neuronas de salida es 24. Los algoritmos de Retropropagación y Levenberg-Marquardt se emplearon para entrenar a la red. Para determinar los parámetros para entrenar las redes neuronales, se realizaron varios experimentos con diferentes configuraciones, de los cuales seleccionamos aquellos parámetros que otorgaron los mejores resultados. El factor de aprendizaje es 0.01, la cantidad máxima de épocas para todos los experimentos es de 2000 y se utilizaron dos arquitecturas para evaluar cada algoritmo de aprendizaje:

- 1 Capa oculta con 31 neuronas
- 2 Capas ocultas, la primera con 19 neuronas y la segunda con 12.

Tabla 2
Resultados obtenidos utilizando Retropropagación como algoritmo de aprendizaje

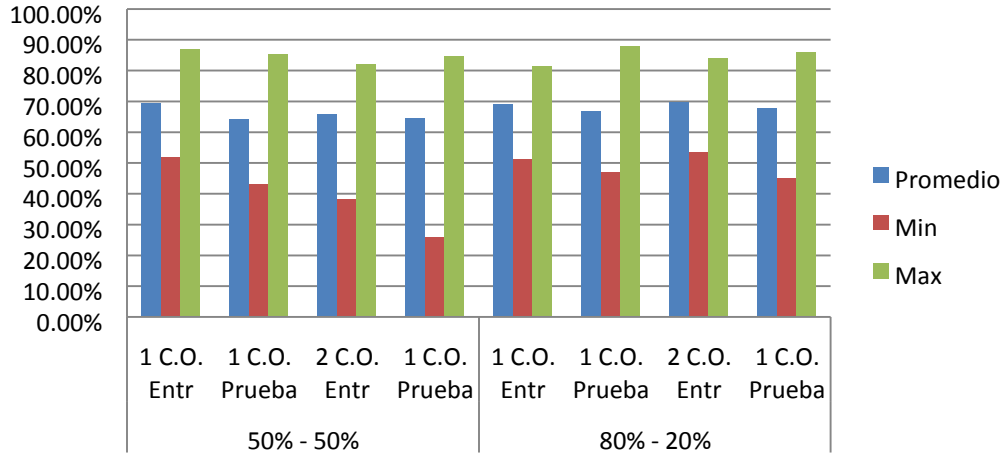
Retropropagación								
	50% - 50%				**80% - 20%**			
	1 C.O. Entr	**1 C.O. Prueba**	**2 C.O. Entr**	**1 C.O. Prueba**	**1 C.O. Entr**	**1 C.O. Prueba**	**2 C.O. Entr**	**1 C.O. Prueba**
% Promedio	69.30%	64.28%	65.88%	64.44%	68.98%	66.73%	69.61%	67.69%
% Mínimo	52.03%	43.09%	38.21%	26.02%	51.27%	46.94%	53.30%	44.90%
% Máximo	86.99%	85.37%	82.11%	84.55%	81.22%	87.76%	83.76%	85.71%
Desviación Estándar	0.0975	0.1143	0.0909	0.1229	0.0836	0.1137	0.0783	0.1088
C.O. = Capa Oculta								**Entr = Entrenamiento**

Elaboración propia

La Tabla 2 presenta los resultados obtenidos en los experimentos con el Perceptrón Multicapa, empleando la Retropropagación como algoritmo de aprendizaje. El porcentaje de clasificación más bajo en la fase de prueba fue 64.28% y se obtuvo utilizando el 50% de las muestras para entrenar a la red con una capa oculta. El mejor resultado conseguido fue de 67.69%, utilizando el 80% de las muestras en el grupo de entrenamiento y configurando la red con dos capas ocultas.

Los valores de desviación estándar obtenidos son más altos que los de la distancia Euclidiana, esto significa que al emplear este método con el enfoque existe un mayor grado de dispersión entre los porcentajes de clasificación de los experimentos, lo cual se puede apreciar claramente en la figura 6, donde los valores mínimos y máximos se alejan considerablemente de los valores promedio. Un buen resultado de clasificación, utilizando este método, depende en gran parte de la selección de las muestras para el conjunto de entrenamiento.

Figura 6
Resultados obtenidos con el perceptrón multicapa y el algoritmo de Retropropagación.



Elaboración propia

Debido a los resultados tan bajos, es evidente que el perceptrón multicapa entrenado con el algoritmo de retropropagación no es la mejor opción para el enfoque propuesto en esta investigación, por lo que el siguiente paso fue experimentar con un algoritmo de aprendizaje distinto: Levenberg-Marquardt.

Los resultados de los experimentos realizados con el Perceptrón Multicapa y el algoritmo Levenberg-Marquardt se presentan en la Tabla 3. Considerando únicamente los resultados de las fases de prueba, el porcentaje de clasificación más bajo fue de 34.96%, utilizando el 50% de las muestras en el grupo de entrenamiento; el porcentaje de clasificación más alto fue de 97.96%, entrenando la red con el 80% de las muestras; mientras que el porcentaje promedio más alto se consiguió al entrenar la red con el 80% de las muestras. Los tres porcentajes obtenidos con una arquitectura de una capa oculta.

ISSN 1405-6690 impreso
ISSN 1665-8612 electrónico

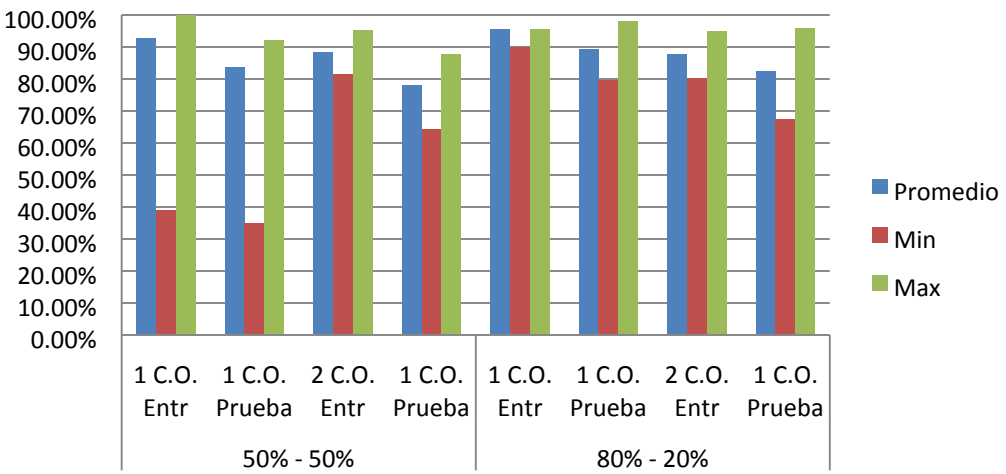
Tabla 3
Resultados obtenidos utilizando Levenberg - Marquardt como algoritmo de aprendizaje

Levenberg - Marquardt								
	50% - 50%				80% - 20%			
	1 C.O. Entr	1 C.O. Prueba	2 C.O. Entr	1 C.O. Prueba	1 C.O. Entr	1 C.O. Prueba	2 C.O. Entr	1 C.O. Prueba
% Promedio	92.76%	83.60%	88.32%	78.02%	95.36%	89.25%	87.58%	82.45%
% Mínimo	39.02%	34.96%	81.30%	64.32%	89.85%	79.59%	80.20%	67.35%
% Máximo	100.00%	91.87%	95.12%	87.80%	95.36%	97.96%	94.92%	95.92%
Desviación Estándar	0.1242	0.1137	0.0375	0.0561	0.0256	0.0491	0.0371	0.0768
C.O. = Capa Oculta					Entr = Entrenamiento			

Elaboración propia

Como se observa en la figura 7, los porcentajes promedio obtenidos al emplear una arquitectura de una capa oculta son más altos que los obtenidos con una arquitectura de dos capas ocultas; esto probablemente se debe a que 2000 épocas no son suficientes para ajustar los parámetros en las neuronas ocultas cuando el perceptrón multicapa es diseñado con un mayor número de capas.

Figura 7
Resultados obtenidos con el perceptrón multicapa y el algoritmo Levenberg-Marquardt



Elaboración propia

En cuanto a la desviación estándar, se presenta un menor grado de dispersión con respecto al porcentaje promedio cuando la red es entrenada con el 80% de las muestras, obteniendo los resultados más precisos al entrenar y tener un diseño de una capa oculta. Por el contrario, entrenar la red con tan sólo 50% hace que la red sea poco fiable.

A simple vista, se puede afirmar que el mejor método de clasificación para utilizar con el enfoque propuesto, es el perceptrón multicapa con el algoritmo de aprendizaje Levenberg-Marquardt, con el que se obtuvieron los mejores resultados tanto en porcentaje promedio como en porcentajes mínimo y máximo.

Comparación de los resultados obtenidos entre los distintos métodos de clasificación

La Tabla 4, muestra la comparación entre los porcentajes de clasificación correcta obtenidos mediante la distancia Euclidiana y el perceptrón multicapa, con una y dos capas ocultas, usando los algoritmos de aprendizaje de Retropropagación y LevenbergMarquardt. Como se puede observar, el mejor resultado es de 89.25% y se obtuvo con el perceptrón multicapa, con una arquitectura de una capa oculta, entrenando con el algoritmo Levenberg-Marquardt y empleando el 80% de las muestras para el entrenamiento.

Este porcentaje promedio demuestra que el enfoque puede ser utilizado en la clasificación de malware, aunque es importante considerar que es posible alcanzar mejores resultados.

ISSN 1405-6690 impreso
ISSN 1665-8612 electrónico

Contrario a lo que se esperaba, el método con los porcentajes de clasificación más bajos fue el perceptrón multicapa con Retropropagación, en vez del método con Distancia Euclidiana. La figura 8 muestra los resultados de forma gráfica para facilitar su compresión. La precisión del enfoque propuesto aumenta al incrementar la cantidad de muestras utilizadas para el entrenamiento; excepto por los resultados obtenidos con la distancia Euclidiana, donde no hubo una diferencia significativa en los resultados al cambiar la cantidad de muestras en el conjunto de entrenamiento.

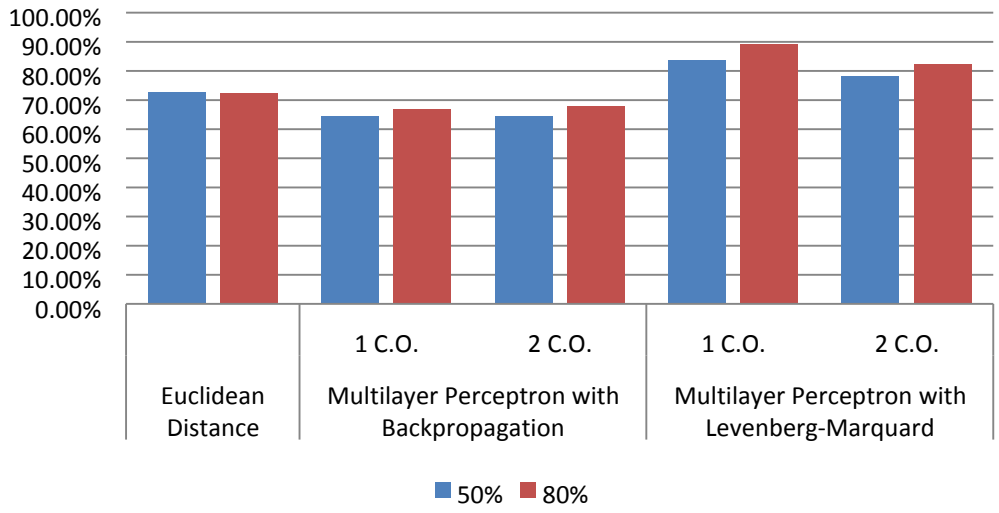
Tabla 4
Comparación general de la precisión promedio durante las etapas de prueba

%	Distancia Euclidiana	Perceptrón Multicapa con Retropropagación		Perceptrón Multicapa con Levenberg-Marquardt	
		1 C.O.	2 C.O.	1 C.O.	2 C.O.
50%	72.76%	64.28%	64.44%	83.60%	78.02%
80%	72.35%	66.73%	67.69%	89.25%	82.45%
C.O. = Capa Oculta					

Elaboración propia

Dado que el enfoque, utilizado junto con una red neuronal, analiza la similitud entre un programa desconocido y un conjunto de programas previamente identificados como maliciosos, es posible considerarlo como un método de heurística genérica, ya que si el vector numérico de un ejecutable es suficientemente similar a alguno de los vectores de la base de datos de malware, éste será identificado como una variante de esa familia. En comparación, el funcionamiento del enfoque, empleado como entrada para calcular la Distancia Euclidiana entre una muestra y todas las familias, es más parecido a un método basado en definiciones; ya que, aunque no busca un vector en la base de datos que sea exactamente igual a la muestra que se quiere clasificar, si está comparando el vector con cada uno de los vectores promedio que representan las familias de malware.

Figura 8
Comparación de los resultados promedio obtenidos con todos los métodos de clasificación.



Elaboración propia

Por último, en la Tabla 5 se presenta una comparación entre los porcentajes de clasificación de las marcas antivirus AVAST, Microsoft Security Essentials, Panda y Norton. Para obtener los porcentajes de los antivirus se analizó cada una de las muestras de la base de datos; es importante recordar que no existe un estándar para nombrar y clasificar programas maliciosos, por lo que algunos antivirus pueden identificar la misma familia con nombres distintos. Cabe aclarar que los resultados mostrados son subjetivos, es decir, no indican la eficacia del antivirus para distinguir entre un programa malicioso y uno no dañino, se refieren únicamente a su capacidad para identificar la familia a la que pertenece una muestra.

Los criterios que se siguieron para considerar una clasificación correcta fueron los siguientes:

- El nombre de la familia debía ser idéntico o similar al de la muestra en la base de datos.

- Lo único que se tomó en cuenta fue el nombre de la familia. El tipo, la plataforma, el índice de la variante y los datos adicionales fueron ignorados.

ISSN 1405-6690 impreso
ISSN 1665-8612 electrónico

- Si el nombre no coincidía, se revisaba en alguna enciclopedia virtual para comprobar que se tratara de un alias de la familia en cuestión. La clasificación se consideró correcta en los casos donde la mayoría de las muestras de una familia fueron identificadas de la misma forma, aún si no se encontró un registro de relación entre ambos nombres.

La clasificación se consideró incorrecta cuando se dio alguna de estas situaciones:

- El antivirus no detectó el programa malicioso.
- La familia de la muestra no era la adecuada.
- El resultado del análisis era un nombre genérico, como por ejemplo: Suspicious File, Win32:Worm Generic, Backdoor Program, Generic Malware, Trojan-gen, Generic Trojan, Infostealer, Generic Backdoor, Trj/Genetic.gen, Trojan Horse, Trojan.ADH, Trojan.Gen.2, Win32:Malware-gen, Trojan.Gen, Trojan.Malcol, etc.

Tabla 5
Porcentaje de clasificación correcta del método propuesto y de algunos antivirus comerciales.

	Método Propuesto	AVAST	MS Security Essentials	Panda	Norton by Symantec
% Total	89.25%	77.14%	92.24%	56.33%	71.43%

Elaboración propia

Es importante mencionar que todos los programas de antivirus comerciales fueron instalados y configurados usando las características y propiedades base de cada antivirus. No fue necesario configurar ningún parámetro adicional.

Como se puede observar en la Tabla 5, el método propuesto parece ser más efectivo en la clasificación de variantes de malware a la familia correcta, que las técnicas empleadas por los antivirus AVAST, Panda y Norton. No es suficiente detectar la muestra y marcarla como maliciosa. Es importante definir a qué familia pertenece para tener un registro de sus características, su comportamiento y los daños que causa; así, en caso de infección, los usuarios sabrán cómo removerla sin tener que recurrir a un técnico. Es por esta razón que los términos genéricos pueden resultar inútiles e inconvenientes.

Conclusiones

Actualmente existen diversos enfoques de detección y clasificación de programas maliciosos como los troyanos y los gusanos; a pesar de eso, la cantidad de dispositivos electrónicos infectados, a nivel mundial, sigue siendo muy alta. Los problemas a los que se enfrentan las compañías antivirus también van en aumento, algunos de los principales son:

- Las miles, o quizá millones, de muestras que reciben diariamente sus laboratorios de análisis.

- Las técnicas empleadas por los creadores de código malicioso para dificultar la detección de sus programas, tales como: modificaciones al código, uso de compresores o packers, ofuscación, algoritmos de cifrado, etc.

- Los programas maliciosos complejos desarrollados por los gobiernos con fines de espionaje y sabotaje.

Debido a que ninguna técnica de detección y clasificación es completamente efectiva, en esta investigación se propuso un nuevo enfoque, basado en el hecho de que la mayoría de los programas maliciosos nuevos son variaciones de muestras que ya han sido identificadas. Al conjunto de variantes de un programa malicioso se le conoce como familia y las muestras pertenecientes a la misma familia tienen estructuras internas similares. Una de esas estructuras internas es la IAT (Import Address Table o Tabla de Dirección de Importaciones), que contiene las funciones del sistema que los ejecutables importan de las bibliotecas de enlace dinámico (DLL).

El enfoque propuesto en esta investigación está basado en el reconocimiento de patrones y es capaz de clasificar muestras nuevas de programas maliciosos, utilizando vectores numéricos con la cantidad de funciones llamadas de cada DLL.

Al principio, se evaluó el desempeño de este enfoque con un clasificador linear, la distancia Euclidiana, con el que no se obtuvieron resultados aceptables.

En segundo lugar, se utilizó una red de perceptrón multicapa, entrenado con el algoritmo de retropropagación, para clasificar los patrones de las muestras a sus respectivas familias; los resultados obtenidos fueron sorpresivamente bajos a comparación de los resultados de la distancia Euclidiana.

ISSN 1405-6690 impreso
ISSN 1665-8612 electrónico

Por último, se utilizó el perceptrón multicapa, con el algoritmo de aprendizaje Levenberg-Marquardt y fue con este método de clasificación con el que se alcanzaron los mejores resultados, donde el promedio de clasificación correcta más alto fue de 89.25%, aunque el mejor porcentaje logrado fue de 97.96%. Aunque la arquitectura utilizada para validar el enfoque es una de las más conocidas y utilizadas en la resolución de problemas de reconocimiento de patrones y de que los resultados obtenidos fueron aceptables, no significa que el perceptrón multicapa sea el mejor o el único método que se puede utilizar con el enfoque propuesto; se puede usar con distintos tipos de clasificadores y redes neuronales, con la posibilidad de obtener mejores resultados.

A pesar de las complicaciones causadas por el análisis estático de las muestras, los resultados demuestran que este enfoque podría ser implementado, probablemente no como un único método de clasificación, pero sí como filtro o en combinación con otros métodos ya existentes, para reducir la cantidad de muestras que deben ser analizadas detalladamente, aumentar las detecciones y disminuir considerablemente los problemas causados por software malicioso. Las ventajas de este enfoque son las siguientes:

- El vector de características que se utiliza para representar una muestra de malware es creado mediante un procedimiento muy simple, que puede ser fácilmente implementado con cualquier lenguaje de programación, puesto que sólo consiste en contar la cantidad de funciones que se llama de cada biblioteca del sistema.

- Al tener características numéricas, el vector puede ser utilizado con una gran variedad de clasificadores y con diferentes tipos y arquitecturas de redes neuronales artificiales, por lo que resulta sencillo experimentar con diferentes métodos para encontrar aquel con el que se puedan obtener los mejores resultados.

- Este enfoque es independiente de las secuencias de llamadas API, dado que únicamente requiere la cantidad de funciones por biblioteca, así que no se ve afectado por las técnicas que reacomodan las secuencias de funciones, como las que utilizan los programas maliciosos polimórficos o metamórficos para evadir la detección y dificultar su análisis.

Sin embargo, es necesario considerar lo siguiente:

- Para obtener de un programa las características necesarias para crear su vector representativo, es necesario llevar a cabo un análisis estático de las muestras, que involucra una cantidad considerable de software externo, como por ejemplo: un

debugger, un desensamblador, un detector de packers, un reconstructor de la IAT, etc. Esto hace que el proceso de extracción de funciones importadas resulte ineficiente, por lo que es necesario automatizarlo y hacerlo, en la medida de lo posible, independiente de otro software. Tomando en cuenta que los programas maliciosos se vuelven cada vez más complejos y por lo tanto resulta más complicado obtener las funciones importadas, por lo que es necesario actualizar constantemente la información relativa a los packers y a las técnicas diseñadas para dificultar la Ingeniería Inversa.

- Una desventaja importante de este enfoque, utilizando el perceptrón multicapa, es que es que cada vez que un programa malicioso llame una función de una biblioteca nueva, se tiene que volver a entrenar la red. Y el proceso de aprendizaje se vuelve cada vez más lento a mayor cantidad de muestras en la base de datos.

- Se deben incluir en los experimentos muestras de programas no maliciosos para poder determinar si este enfoque puede realmente detectar posibles amenazas sin generar tantos falsos positivos, es decir, sin afectar el funcionamiento de los programas no dañinos.

Aunque los resultados presentados en este artículo fueron estadísticamente validados, todavía es necesario aplicar otro tipo de análisis donde se incluya un análisis sobre los falsos positivos del sistema. Esto es muy importante para trabajos futuros, ya que la emisión de falsas alertas es uno de los mayores problemas en los sistemas de detección actuales.

Bibliografía

Abdulla, S. M. (2012). Minimizing Errors in Identifying Malicious API to Detect PE Malwares Using Artificial Costimulation. *International Conference on Emerging Trends in Computer and Electronics Engineering*, (pp. 49-54).

Aberdeen's Robert Gordon University (2005). *An Introduction to Neural Networks.* Retrieved from http://www4.rgu.ac.uk/files/chapter2%20-

Alazab, M. L. (2010). Malware Detection Based on Structural and Behavioral Features of API Calls. *Proceedings of the 1st International Cyber Resilience Conference*, (pp. 1-10).

Bayer, U., Comparetti, P. M., Hlauschek, C., Kruegel, C., & Kirda, E. (2009). Behavior-Based Malware Clustering. *Network and Distributed System Security Symposium.*

ISSN 1405-6690 impreso
ISSN 1665-8612 electrónico

Carrera, E. y Erdélyi (2004). Digital genome mapping – advanced binary malware analysis. *Virus Bulletin*.

Chávez, G. (2012, Abril 26). *México es un país muy vulnerable al malware.* Retrieved from Excélsior: http://www.excelsior.com.mx/2012/04/26/dinero/829582

ESET. (2011). *ESET Security Report Mexico.* Retrieved from http://www.esetla.com/pdf/prensa/informe/eset-report-security-latinoamerica-2012.pdf

ESET. (2011). *Heurística Antivirus: Detección Proactiva.* Retrieved from http://www.esetla.com/pdf/prensa/informe/heuristica_antivirus_deteccion_proactiva_malware.pdf

Freeman, J. & Skapura, D. M (1993). *Redes Neuronales. Algoritmos, aplicaciones y técnicas de propagación.* Addison-Wesley.

George E. Dahl, J. W. (2013). LARGE-SCALE MALWARE CLASSIFICATION USING RANDOM PROJECTIONS AND NEURAL NETWORKS. *Proceedings IEEE Conference on Acoustics, Speech, and Signal Processing.* IEEE SPS.

Gheorghescu, M. (2006). An Automated Virus Classification System. *Virus Bulletin* .

Global Research & Analysis Team, Kaspersky Lab. (2012, Mayo). *Securelist: Monthly Malware Statistics: April 2012.* Retrieved from http://www.securelist.com/en/analysis/204792228/Monthly_Malware_Statistics_April_2012

Kaspersky Lab. (2012, Febrero 23). *Amenazas Informáticas (Malware) | Kaspersky Lab España.* Retrieved from http://www.kaspersky.com/sp/threats

Kinable, J. & Kostakis, O. (2011). Malware classification based on call graph clustering. *Journal in Computer Virology*. Vol. 7, pp. 233-245.

Lee, T. &. Mody, J. J. (2006). Behavioral Classification. *Proceedings of Annual Conference of the European Institute for Computer Antivirus Research (EICAR).* Hamburgo.

Mamoun Alazab, S. V. (2011). Zero-day Malware Detection based on Supervised Learning Algorithms of API call Signatures. *Proceedings of the 9-th Australasian Data Mining Conference (AusDM'11)* (pp. 171-181). Ballarat, Australia: Australian Computer Society, Inc.

Norton by Symantec. (2013, 23 Febrero). *Cybercrime – Computer Crimes Security – Online Fraud – Email Phishing Scams.* Retrieved from http://us.norton.com/cybercrimeindex/

Panda Security. (2012, 23 Febrero). *Classic Malware.* Retrieved from http://www.pandasecurity.com/homeusers/security-info/classic-malware/

Quist, D. (2005). *Offensive Computing, LLC.* Retrieved from http://www.offensivecomputing.net/

Rieck, K. H. (2008). Learning and Classification of Malware Behavior. *Proceedings of the 5th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment.*

Shevchenko, A. (2007, Noviembre). *Securelist: The Evolution of technologies used to detect malicious code.* Retrieved from http://www.securelist.com/en/analysis/204791972/The_evolution_of_technologies_used_to_detect_malicious_code?print_mode=1

Veeramani, R. & Rai, N. (2012). Windows API based Malware Detection and Framework Analysis. *International Journal of Scientific & Engineering Research*, Vol. 3, Num. 3, pp. 1-6.

VX Heavens. (2012). *VX Heavens.* Retrieved from http://vx.netlux.org/

Walenstein, A. V. (2007). Exploiting similarity between variants to defeat malware: "Vilo" method for comparing and searching binary programs. *Proceedings of BlackHat DC 2007.*

Zhang, Q. & Reeves, D. S. (2007). Metaaware: Identifying metamorphic malware. *ACSAC*, (pp. 411-420).

ISSN 1405-6690 impreso
ISSN 1665-8612 electrónico