ISSN 1870-4069

Revisión del estado del arte en técnicas de procesamiento de lenguaje natural para análisis de malware

Horacio Rodriguez-Bazan, Grigori Sidorov,
Ponciano Jorge Escamilla-Ambrosio

Instituto Politécnico Nacional,
Centro de Investigación en Computación,
México

b160524@sagitario.cic.ipn.mx,
sidorov, pescamilla}@cic.ipn.mx

Resumen. Hoy en día, casi todas las personas en el mundo tienen al menos un dispositivo móvil, la compañía Google Inc. creó el sistema operativo Android en 2008 que se convertiría en el sistema operativo más utilizado en los dispositivos móviles de todo el mundo, sin embargo, ya que cubre más del 80 % del dispositivo móviles en el mundo se convierte también en el objetivo principal para los ciberdelincuentes, aprovechando el riesgo que existe en las aplicaciones que instalamos en nuestros dispositivos mismas que podría contener malware (del inglés, software malicioso), analizar todas las aplicaciones almacenadas en una tienda de aplicaciones antes de ponerlas a disposición de las personas sería una tarea difícil o prácticamente imposible ya que día a día salen nuevas aplicaciones al mercado. Existen investigaciones que han trabajado con este tipo de aplicaciones maliciosas para analizarlas, aplicando técnicas como Procesamiento de Lenguaje Natural PNL (del inglés, Natural Language Processing, abreviado NLP), en este artículo se presentan aquellos trabajos que aplicaron PNL, para la detención de código malicioso en las aplicaciones, analizando dichos trabajos y las técnicas utilizadas, los resultados y el trabajo futuro.

Palabras clave: Análisis de malware, PLN, Word2Vec, BoW, TF-IDF.

State of the Art Review of Natural Language Processing Techniques for Malware Analysis

Abstract. Today, almost everyone in the world has at least one mobile device, the company Google Inc. created the Android operating system in 2008 that would become the most widely used operating system on mobile devices worldwide, however, since it covers more than 80% of the mobile devices in the world, it also becomes the main target for cyber

criminals, taking advantage of the risk that exists in the applications that we install on our devices that could contain malware, analyzing all the applications stored in an application store before making them available to people would be a difficult or practically impossible task, since new applications come onto the market every day. There are investigations that have worked with this type of malicious applications to analyze them, applying techniques such as Natural Language Processing NLP. This article presents those works that applied NLP, to stop code malicious in applications, analyzing those jobs and the techniques used, the results and future work.

Keywords: Malware analysis, PLN, Word2Vec, BoW, TF-IDF.

1. Introducción

El creciente desarrollo en las comunicaciones ha hecho que la humanidad busque nuevas formas para comunicarse más rápida y efectivamente, siendo la telefonía móvil la más usada alrededor del mundo para dicho propósito, esta necesidad ha llevado a tener la mayoría de los recursos en un solo dispositivo móvil. Siendo así que la compañía Google Inc. en 2008 creó la primera versión del sistema operativo móvil, el cual se convertiría en el sistema operativo móvil más utilizado hoy en día cubriendo más del 80% del mercado de los dispositivos móviles alrededor del mundo.

Estas nuevas formas de comunicación vía Internet han traído consigo riesgos en todos los sistemas conocidos, mismos los ciberdelincuentes han aprovechado para comprometer la seguridad de los usuarios, Android no ha sido la excepción, por el contrario, al dominar la gran mayoría del mercado de dispositivos móviles se ha convertido en el objetivo preferido de los ciberdelincuentes. Por esta razón es que día a día salen el mercado una cantidad importante de aplicaciones que contienen malware (del ingles, ***MAL****icious soft****WARE***) y sería una tarea complicada analizar cada una de las nuevas aplicaciones por una analista de malware.

Hoy en día las técnicas de Inteligencia Artificial (del ingles, Artificial Intelligence, AI) han sido aplicadas a diversas áreas del conocimiento para resolver problemas que no habían sido posible resolver mediante los métodos tradicionales, gracias a los avances y desarrollos de algoritmos supervisados, así como no supervisados (Machine Learning) ha sido posible emplear estás técnicas a diversos problemas.

Dichas técnicas han sido aplicadas al área de ciberseguridad, en especial para el análisis de malware, como es el caso de estudio de este artículo, la aplicación de las técnicas de Procesamiento de Lenguaje Natural, PLN (del inglés Natural Language Processing, NLP) para el análisis de malware.

El artículo está organizado con la siguiente estructura, en la Sección 1 se mencionan los aspectos relevantes de las aplicaciones Android, Sección 2 muestra la revisión del estado del arte analizando los trabajos en los que se han empleado técnicas del Procesamiento del Lenguaje Natural, en la sección 3 se presenta un

revisión de los resultados obtenidos con las diferentes técnicas aplicadas, en la última sección 4 una recopilación de las áreas no exploradas por los investigadores y que sirven como trabajo a futuro en el análisis de malware aplicando técnicas de Procesamiento de Lenguaje Natural.

2. Aplicaciones Android - APK

Un archivo *APK* es una aplicación para el sistema operativo Android, el sistema operativo móvil de Google Inc. Algunas aplicaciones vienen preinstaladas en los dispositivos, mientras que otras aplicaciones se pueden descargar desde Google Play sitio oficial de Google o inclusive de otras tiendas de aplicaciones. Las aplicaciones descargadas de Google Play se instalan automáticamente en su dispositivo, mientras que las descargadas de otras fuentes deben instalarse manualmente.

Por lo general, para los usuarios es transparente la instalción ya que nunca ven los archivos *APK* porque Android maneja la instalación de aplicaciones en segundo plano a través de Google Play u otra plataforma de distribución de aplicaciones. Sin embargo, hay muchos sitios web que ofrecen descarga directa de archivos *APK* para usuarios de Android que desean instalar aplicaciones manualmente. En este caso, debe tener cuidado de confiar en la fuente del archivo *APK*, ya que el malware se puede distribuir en archivos *APK*.

Los paquetes o aplicaciones de Android contienen todos los archivos necesarios para una sola aplicación. A continuación, se muestra una lista de los archivos y carpetas más destacados:

- ***META-INF /***: contiene el archivo manifiesto, la firma y una lista de recursos en el archivo.
- ***lib /***: bibliotecas nativas que se ejecutan en arquitecturas de dispositivos específicos (armeabi-v7a, x86, etc.)
- ***res /***: recursos, como imágenes, que no se compilaron en resources.arsc
- ***assets /***: archivos de recursos sin formato que los desarrolladores agrupan con la aplicación
- ***AndroidManifest.xml***: describe el nombre, la versión y el contenido del archivo APK.
- ***classes.dex***: las clases compiladas en lenguaje Java que se ejecutarán en el dispositivo (archivo classes.dex)
- ***resources.arsc***: los recursos compilados, como cadenas, utilizados por la aplicación (archivo .arsc)

Los archivos *APK* se guardan en formato comprimido .zip y se pueden abrir con cualquier herramienta de descompresión. Por lo tanto, si desea explorar el contenido de un archivo *APK*, puede cambiar el nombre de la extensión del archivo a ".zip" y abrir el archivo, o puede abrir el archivo directamente a través del cuadro de diálogo abierto de una aplicación .zip [3].

3. Técnicas de procesamiento de lenguaje natural para el análisis de malware

En esta sección se presenta el análisis de los trabajos relacionados al análisis de malware aplicando las técnicas de Procesamiento de Lenguaje Natural.

Algunas técnicas para el análisis de malware proponen el uso de Sandbox (ambiente de pruebas controlado) para analizar las muestras, sin embargo, existen muestras que detectan si están siendo ejecutadas en un ambiente controlado pueden cambiar su comportamiento durante la ejecución. Es así que los investigadores proponen otro método para el análisis porque utilizar sandbox requiere de tiempo para cada una de las muestras y recolectar los resultados sin perde de vista el punto anterior que si la muestra detecta que esta siendo analizada podría cambiar su comportamiento por lo que los datos recopilados pueden no ser los correctos. Las cadenas ASCII extraídas de archivos ejecutables son útiles para analizar malware a este método se le conoce como análisis estático. Con el reciente desarrollo de técnicas de procesamiento de lenguaje natural (PLN), es posible usar estas cadenas como método de detección de malware.

Investigadores proponen un método de detección de malware que utiliza cadenas ASCII con técnicas de PLN. Dicho método divide estas cadenas en palabras y distingue la diferencia de las palabras entre archivos ejecutables benignos y maliciosos. La construcción de un modelo de lenguaje, construye un modelo de lenguaje usando técnicas de PLN. La tecnología de PLN utilizada en este tipo de métodos es LSI (del inglés, Latent Semantic Analysis) y Doc2Vec como lo proponen en esta investigación por Mohammadinodooshan et al. [4].

Doc2Vec, realiza un modelo de lenguaje para convertir las palabras en vectores de características utilizando el modelo PV-DM (del inglés, Paragraph Vector Distributed Memory) del párrafo vectorial, el entrenamiento se lleva mediante Maquinas de Soporte Vectorial (del inglés, Support Vector Machine ,SVM) con los datos de entrenamiento. Este enfoque muestra buenos resultados siempre y cuando las muestras no estén ofuscadas en ese caso este método de detección puede no funcionar como se espera, ya que depende de las cadenas para generar el modelo de espacio vectorial, al ser cadenas ofuscadas estas puede que no se detecten, en las condiciones ideales de trabajo este método tiene una efectividad Doc2Vec de 89% y LSI de 78% [4].

Las llamadas a API como entradas para clasificadores han sido exploradas por varios investigadores que emplean estos datos para sus modelos propuestos. Además, aprovechan el desarrollo en el Procesamiento del Lenguaje Natural (PLN), algunos métodos como n-gramas, doc2vec (Vectores de párrafo), TF-IDF (del inglés, Term Frequency – Inverse Document Frequency, abreviado TF-IDF) para convertir esas secuencias de API a modelo de espacio vectorial antes de alimentar a los clasificadores.

La selección de características generalmente se emplea n-gramas, para generar el vector de características para este fin se pueden emplear TF-IDF, vectores de párrafos (PV-DM y PV-DBOW). Dado el vector de características se emplean cuatro clasificadores comunes: Máquina de vectores de soporte (del inglés, Support Vector Machine, abreviado SVM), vecino K más cercano (del inglés, k-Nearest

Neighbors, abreviado K-NN), bosque aleatorio (del inglés, Random Forest, abreviado RF) y perceptrón multicapa (del inglés, Multi-Layer Perceptron, abreviado MLP).

Los resultados que se obtienen cuando se aplican estos clasificadores muestra la precisión de tres métodos usando cuatro diferentes clasificadores en caso de clasificar dos grupos de malware SVM adquiere la precisión más alta, 99.06% utilizando el método TF-IDF para vectorizar las características de malware. En caso de usando KNN (k = 3) como clasificador, el modelo de bolsa distribuida de palabras nos da la mayor precisión (98.70%) en comparación con Memoria distribuida y TF-IDF con 97.55% y 98.51%, respectivamente como muesta Tran et al. en está investigación [7].

Otro enfoque que se ha aplicado al análisis de malware es el uso de los n-gramas, con este enfoque los investigadores en lugar de utilizar secuencias de código de operación (del inglés, Operation Code, abreviado opcode), con esta técnica solo se analizan las llamadas a API extraídas de secuencias de operandos. Sin embargo, el número de llamadas API únicas es excesivo y las llamadas a API varían a medida que evoluciona el nivel API de Android. Una solución es utilizar el paquete n-gramas que se extraen desde secuencias de paquetes para representar el comportamiento de la aplicación y combinarlos con algoritmos de aprendizaje automático para construir modelos de predicción de malware.

A través de los datas decompilados se genera el código *Smali* que es usado para formar los n-gramas, calculando la frecuencia de paquetes únicos de n-gramas en todos los bloques divididos de cada aplicación, y la firma de n-gramas de una secuencia es un conjunto de todas sus subsecuencias con una longitud n. Dado que el número de n-gramas de paquetes únicos aumenta exponencialmente a medida que n aumenta, se recomendable experimentar variando n de 1 a 5.

Para comparar el rendimiento del paquete Zhang et al. emplean n-gramas con opcode n-gramas en diferentes clasificadores, se probaron cuatro diferentes algoritmos supervisados de aprendizaje automático, por ejemplo, Naive Bayes (NB), Árbol de decisión (DT), vecino k-más cercano (KNN) y aleatorio Bosque (RF), para construir modelos de clasificación.

Los resultados presentados con dichos clasificadores muestran que con n-gramas usando n entre 1 y 5, con cada uno de los clasificadores se tiene una precisión entre 95% y 98%, lo cual hace indicar que para estos datos no hay una diferencia marcada haciendo variaciones de los n-gramas [9].

Uno de los trabajos desarrollados por ElMouatez et al. es *MalDy* (mal die), en el cual se utilizan técnicas supervisadas de aprendizaje automático (Machine Learning), la idea principal de MalDy es el modelado de los informes de comportamiento en una secuencia de palabras, junto con el procesamiento avanzado del lenguaje natural (PLN) y las técnicas de aprendizaje automático (ML) para la ingeniería automática de características de seguridad relevantes para detectar y atribuir malware sin la supervisión de un analista de malware.

MalDy realiza una representación de características (palabras) haciendo ingeniería en características automáticamente sin la intervención de un experto en seguridad, MalDy propone emplear el modelo de PLN bolsa de palabras (del

inglés, Bag of Word, BoW), usando la Frecuencia de Documento de Frecuencia Inversa (TF-IDF) o Función Hashing tricking (FH) MalDy tiene dos variantes basadas en la técnica BoW elegida, ya sea TF-IDF o FH. Estas técnicas generan vectores de longitud fija a partir de informes de comportamiento utilizando frecuencias de palabras.

La evaluación de MalDy muestra la efectividad y la portabilidad de MalDy en todo el espectro de los análisis y de la configuración, las técnicas de PNL y ML para construir conjuntos de aprendizaje automático discriminativos. MalDy logra más del 94% de puntaje f1 en la tarea de detección de Android en los conjuntos de datos Malgenome, Drebin y MalDozer y más del 90% en la tarea de atribución [5].

Otro trabajo realizado en el que se propone un método resistente a la ofuscación es el propuesto por Aghamohammadi et al., llamado ORDroid, que puede detectar malware mutados y transformados, empleando redes neuronales (RNN) y procesamiento de lenguaje natural (PLN) para lograr este propósito. Adicionalmente proponen un método ligero de detección de malware, llamado LightDroid. La idea principal de este método es seleccionar un número mínimo de características del archivo Android Manifest, junto con una serie de características basadas en imágenes del archivo ejecutable Dalvik de manera que la precisión del modelo resultante esté cerca del estado del arte, mientras que su complejidad computacional sea lo más baja posible.

Los resultados muestran que LightDroid es el más ligero, con un 97,49% de precisión en los datos de prueba, mientras que ORDroid muestra que, teniendo en cuenta la precisión general de ambas pruebas y los datos transformados, dicho modelo es el mejor en comparación con los métodos más relacionados con una precisión del 98.07% en lo normal y del 93.00% en los datos transformados[1].

DroidVecDeep, un método de detección de malware de Android que utiliza una técnica de aprendizaje profundo.

Primero extraen características y las clasifican usando la impureza de disminución media. Después, transforman las características en vectores compactos basados en word2vec. Finalmente, entrenan el clasificador basado en el modelo de aprendizaje profundo.

Los datos de las aplicaciones se obtienen a través de la ingeniería inversa del APK y se extrae el archivo AndroidManifest.xml y la carpeta que contiene el código fuente smali usando Apktool. Los “Permission” e “Intent Action” se obtiene del archivo AndroidManifest.xml. Los permisos describen los permisos requeridos por la aplicación, aunque algunos de ellos serían potencialmente riesgosos, del código “smali” se obtienen algunas cadenas sensitivas que se utilizan como características.

Las características extraídas las representan como un vector de características basado en la codificación “one-hot”, con base en las investigaciones en Procesamiento de Lenguaje Natural (PLN), la codificación “one-hot” no contiene ninguna información de corpus, y la distancia entre todas las palabras es la misma. Word2vec define el vector de palabras de acuerdo con el contexto, y las palabras con alta relevancia tienen distancias más cercanas.

Word2vec utiliza dos modelos para producir una representación distribuida de palabras: el modelo continuo de bolsa de palabras (CBOW) y el modelo Skip-Gram. En este trabajo se trata la falta de extracción y caracterización de algunas características en trabajos del estado del arte, para ello evaluaron 3,000 aplicaciones benignas y 12,000 malware. Los resultados muestran que DroidVecDeep funciona bien en precisión y eficiencia de ejecución y es superior a algunas herramientas de detección maliciosas [2].

Una novedosa combinación de arquitecturas de visualización y aprendizaje profundo para un análisis híbrido entre el análisis estático y dinámico basado en el procesamiento de imágenes aplicado en un entorno de Big-Data. Es el primero de su tipo para lograr una detección inteligente de malware de día cero.

Para esta investigación, los autores utilizaron el conjunto de datos Ember con un subconjunto que contiene 70,140 archivos benignos y 69,860 maliciosos. El conjunto de datos se dividió aleatoriamente en un 60% de entrenamiento y un 40% de pruebas usando Scikit-Learn.

Los resultados se presentaron en tres enfoques para el análisis estático, dinámico y para procesamiento de imágenes. Los autores probaron el modelo de tres maneras diferentes para el análisis estático, logrando una precisión máxima del 98.9% usando el algoritmo DNN, para el análisis dinámico la precisión máxima fue del 93.6% usando el algoritmo CNN y para el procesamiento de imágenes fue hasta 96.3% usando "CNN 2 + LSTM".

En dicha investigación se tuvieron cuatro contribuciones, la primera es una nueva propuesta de un marco escalable e híbrido, ScaleMalNet facilita la recolección de muestras de malware de diferentes fuentes de forma distribuida y la aplicación de preprocesamiento de manera distribuida.

El segundo es una nueva técnica de procesamiento de imágenes para la clasificación de malware. El tercero ScaleMalNet sigue un enfoque de dos etapas, en la primera etapa el archivo ejecutable se clasifica en malware o legítimo utilizando análisis estático y dinámico y en la segunda etapa el archivo ejecutable de malware se clasifica en la familia de malware correspondiente, y la contribución de cuatro es independiente evaluación del desempeño de las arquitecturas clásicas de MLA y aprendizaje profundo, comparando varios modelos de análisis de malware [8].

Otra técnica implementando un enfoque secuencial, que incluye agrupamiento, clasificación y Blockchain. El aprendizaje automático (ML) extrae la información automáticamente del malware utilizando técnicas de agrupamiento y clasificación y almacena la información en la cadena de bloques.

La investigación tuvo dos contribuciones, la primera es que la técnica propuesta consiste tanto en tecnología Blockchain como en técnicas de aprendizaje automático para la detección de malware de dispositivos IoT (del inglés, Internet of Things), y la segunda es que el aprendizaje automático extrae automáticamente la información del malware mediante la técnica de agrupamiento y clasificación, y almacena la información en la Blockchain. Los autores propusieron el algoritmo Naive Bayes de funciones múltiples para la clasificación basada en árboles de

decisión para extraer características más importantes para proporcionar clasificación y regresión para lograr una alta precisión y robustez.

Para esta investigación, los autores utilizaron un conjunto de datos que incluye 6,192 aplicaciones de malware benignas y 5,560 recopiladas de Google Play Store y Chinese App store con este conjunto de datos, logran una precisión del 98.0% utilizando el algoritmo Naive Bayes [6].

4. Mejores resultados

Investigadores han tratado de resolver el problema del análisis de malware, como se vio en la sección anterior este malware puede ser para PC (con sistema operativo Microsoft Windows) o para Móviles (con sistema operativo Android), estos dos tipos arquitecturas son las más atacadas por malware, lo que ha representado una tarea difícil de resolver cuando se presentan muestras nuevas o de día cero (aquellas de las que no se conoce nada).

La aplicación de algoritmos supervisados y no supervisados traen consigo que se puedan clasificar en familias y conocer de qué tipo de muestra se está tratando, los resultados mostrados en esta investigación sugieren el uso se técnicas de Procesamiento de Lenguaje Natural (PLN), por ejemplo el uso de n-gramas para generar la selección de características, sin embargo, no existe una guía que indique que características son las que se deben tomar en cuenta o de qué modo, es decir, si las características se toman con base en los resultados que se obtienen al ejecutar la muestra (análisis dinámico) o mediante el código (análisis estático).

Sabemos que hoy en día con los controles de seguridad que existen los atacantes están invocando la forma de pasar estos controles de seguridad a fin de hacer llegar al usuario estas aplicaciones con malware, los dos enfoques tanto el análisis dinámico como el estático han dado buenos resultados al introducirlos en los clasificadores más comunes Naive Bayes, KNN, RF, DT y RNN por citar algunos, en la mayoría de ellos se alcanza una precisión por arriba del 95% en la mayoría de los casos. Lo cual hace pensar que se tiene el problema resuelto para el análisis de malware, sin embargo, estos algoritmos se han probado en corpus conocidos.

Para nuevos desarrollos de métodos de análisis de malware al menos se deben considerar estos algoritmos como base para el desarrollo de alguna nueva forma de detección, ya que se ha observado que son los más usados por los investigadores gracias a los resultados que dan mediante la combinación de técnicas de Procesamiento de Lenguaje Natural con algoritmos de Machine Learning.

Table 1 muestra en resumen los trabajos que se revisaron como estado del arte, donde ellos muestran los algoritmos y técnicas de Procesamiento de Lenguaje Natural aplicados, como se observa, en todos los casos no se aplican las mismas técnicas, sin embargo obtienen resultados alentadores en cada uno de ellos, los resultados dependen de las características que están empleando en los modelos, lo que hace alusión es que no hay una forma única de elegir las características, pero sí que existen métodos para elegir las caracterizas más optimas.

Table 1. Tabla comparativa de algorítmos y resultados.

Herramienta	Algoritmos PLN	Clasificadores	Plataforma	Resultados
A [4]	LSI	SVM	PC	LSI 78.0%
	Doc2Vec			Doc2Vec 89.0%
B [7]	n-gramas	SVM	PC	SVM - TF-IDF 99.06%
	Doc2Vec	KNN		KNN - PV-DBOW 98.70%
	TF-IDF	RF		KNN - PV-DM 97.55 %
	TF-IDF	MLP		KNN - TF-IDF 98.51%
C [9]	n-gramas	NB	PC	n-gramas[1-5] 95% a 98%
		DT		
		KNN		
		RF		
D [5]	BoW	KNN	Android	KNN 59.2%
	TF-IDF	RF	PC	RF 87.00%
	FH	SVM		SVM 87.71%
E [1]	Word2Vec	RNN	Android	98.07%
F [2]	Word2Vec	DBN	Android	97.29%
	CBoW	SVM		96.40%
	Skip-Gram	NB		92.30%
G [8]	Scikit-Learn	DNN	Android	98.9%
		CNN		93.6%
		CNN2 + LSTM		96.3%
H [6]	Word2Vec	KNN	Android	92.0%
		BN		98.0%
		DBN		87.0%

5. Trabajo a futuro

Durante la revisión del estado del arte, se identificaron algunas áreas que no han sido del todo exploradas, aplicando las técnicas de Procesamiento de Lenguaje Natural para el análisis de malware en Android.

– Análisis estático, aplicando procesamiento de lenguaje natural en los archivos clave de las aplicaciones como lo son *AndroidManifest.xml*, *classes.dex*, *resources.arsc*, ya ellos pueden ser procesados como texto para generar un modelo de espacio vectorial que permita conocer que tan similares son unas aplicaciones de otras o que caracteristicas se comparten, se debe tener en cuenta los métodos de ofuscación de Android dado que el texto para analizar no sería el mismo.

– Análisis dinámico, con base en los datos generados mediante la ejecución de la aplicacion en un ambiente controlado recolectar dichos datos y aplicar técnicas de procesamiento de lenguaje natural a fin de tener un modelo de espacio vectorial que incluya otras características adicionales a las llamadas a sistema API.
– Análisis hibrido, con base en los datos generados en los dos puntos anteriores (análisis estático y dinámico), generar un nuevo modelo de espacio vectorial que permita incluir ambos resultados a fin de tener las dos formas del análisis y que se puedan aplicar los algoritmos de Machine Learning con base en los dos criterios.
– Probar dichos modelos con otros corpus a fin de medir que tan eficiente son estos modelos cuando se enfrentan a datos desconocidos.
– Desarrollo de un método basado en aplicación de técnicas de lenguaje natural y aprendizaje automático para detectar autoría de códigos maliciosos ofuscados en aplicaciones Android, a fin de tener otra característica que se pueda incluir en un modelo de análisis de malware para robustecer los modelos de detección.
– Evaluación del método (experimentos con aplicación de este método con aplicaciones Android de autoría diversa).

En resumen, este articulo mostro las técnicas de procesamiento de leguaje natural aplicadas al análisis de malware, como se mostró en esta revisión aún existen técnicas que se pueden explorar para analizar dichas muestras para general modelos de detección que permitan trabajar con diferentes corpus, que puedan ser entrenados y probados con cualquier corpus, es decir un modelo que pueda detectar cualquier malware y que además obtenga una alta precisión.

Agradecimientos. Agradecemos a CONACYT, SNI, IPN (SIP, COFAA), apoyo de proyectos SIP 20200859 y 20200797 y Conacyt A1-S-47854.

References

1. Aghamohammadi, A., Faghih, F.: Lightweight versus obfuscation-resilient malware detection in android applications. Journal of Computer Virology and Hacking Techniques pp. 125–139 (2019)
2. Chen, T., Mao, Q., Lv, M., Cheng, H., Li, Y.: Droidvecdeep: Android malware detection based on word2vec and deep belief network. KSII Transactions on Internet and Information Systems 13(4), 2180–2197 (2019)
3. Hoog, A.: Android Forensics: Investigation, Analysis and Mobile Security for Google Android. Syngress Publishing, 1st edn. (2011)
4. Ito, R., Mimura, M.: Detecting unknown malware from ascii strings with natural language processing techniques. In: 2019 14th Asia Joint Conference on Information Security (AsiaJCIS). pp. 1–8 (2019)
5. Karbab, E.B., Debbabi, M.: Maldy: Portable, data-driven malware detection using natural language processing and machine learning techniques on behavioral analysis reports. Digital Investigation 28, S77–S87 (2019)

6. Kumar, R., Zhang, X., Wang, W., Khan, R.U., Kumar, J., Sharif, A.: A multimodal malware detection technique for android iot devices using various features. IEEE Access 7, 64411–64430 (2019)
7. Tran, T.K., Sato, H.: Nlp-based approaches for malware classification from api sequences. In: 2017 21st Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES). pp. 101–105 (2017)
8. Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Venkatraman, S.: Robust intelligent malware detection using deep learning. IEEE Access 7, 46717–46738 (2019)
9. Zhang, P., Cheng, S., Lou, S., Jiang, F.: A novel android malware detection approach using operand sequences. In: 2018 Third International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC). pp. 1–5 (2018)